IMPUTATION OF MISSING VALUES AND THE APPLICATION OF TRANSFER MACHINE LEARNING TO PREDICT WATER QUALITY IN ACID MINE DRAINAGE TREATMENT PLANTS

WITS
UNIVERSITY

A dissertation submitted to the Faculty of Science,
University of the Witwatersrand, Johannesburg,
in fulfilment of the requirements for the degree of
Master of Science

by

Taskeen Hasrod

Johannesburg, May 2024

DECLARATION

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Science at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other University.

Signature of Candidate : 

Date : 26/05/2024

ABSTRACT

Access to clean water is one of the most difficult challenges of the 21st century. Natural unpolluted water bodies are becoming one of the most dramatically declining resources due to environmental pollution. In countries like South Africa which has a mining-centred economy, toxic pollution from mine tailing dumps and unused mines leach into the underground water table and contaminate it. This is known as Acid Mine Drainage (AMD) and poses a grave threat to humans, animals and the environment due to its toxic element and acidic content. It is, therefore, imperative that sustainable wastewater treatment procedures be put in place in order to decrease the toxicity of the AMD such that clean water may be recovered. An efficient circular economy is created in the process since original wastewater can be recycled to not only provide clean water, but also valuable byproducts such as sulphur (from the elevated sulphate content) and other important minerals. Traditional analytical chemistry methods used to measure sulphate are usually time-consuming, expensive and inefficient, thereby, leading to incomplete analytical results being reported.

To address this, this study aimed at imputing missing values for sulphate concentrations in one AMD treatment plant dataset and then using that to conduct transfer learning to predict concentrations in two other AMD treatment plants datasets.

The approach involved using historical water data and applying geochemical modelling as a thermodynamical tool to assess the water chemistry and conduct preliminary data cleaning. Based on this, Machine Learning (ML) was then used to predict the sulphate concentrations, thus, addressing limited data on this parameter in the datasets. With complete and accurate sulphate concentrations, it is possible to conduct further modelling and experimental work aimed at recovering important minerals such as octathiocane, S_8 (a commercial form of sulphur), gypsum and metals.

Historical data obtained from the three AMD treatment plants in Johannesburg, South Africa (viz., Central Rand, East Rand and West Rand) were obtained and the larger Central Rand dataset was split into smaller untreated AMD (Pump A and Pump B) subsets. Thermodynamic and solution equilibria aspects of the water were assessed using the PHREEQC geochemical modelling code. This served as a preliminary data cleanup step. Eight baseline as well as three ensemble machine learning regression models were trained on the Central Rand subsets and compared to each other to find the best performing model that was then used to conduct Transfer Learning (TL) onto the East Rand and West Rand datasets to predict their sulphate levels.

The findings pointed to a high correlation of sulphate to temperature (°C), Total Dissolved Solids (mg/L) and most importantly, iron (mg/L). The linear correlation between iron and sulphate substantiated pyrite (FeS_2) as their source following weathering. Water quality parameters were found to be dependent on factors such as weather and geography this was evident in the treated water that had quite different chemistry to that of the untreated AMD. Neutralisation agents used were based on those parameters, thus, further delineating the chemistry of the treated and untreated water.

The best performing ML model was the Stacking Ensemble (SE) regressor trained on Pump B's data and combined the best performing models namely, Linear Regressor (LR), Ridge Regressor (RD), K-Nearest Neighbours Regressor (KNNR), Decision Tree Regressor (DT), Extreme Gradient Boosting Regressor (XG), Random Forest Regressor (RF) and Multi-Layer Perceptron Artificial Neural Network Regressor (MLP) as the level 0 models and LR as the level 1 model. Level 0 consisted of training heterogenous base models to obtain the crucial features from the dataset. These individual predictions and features were then fed to a single meta-learner model in in the next layer (level 1) to generate a final prediction. The stacking ensemble model performed well and achieved Mean Squared Error (MSE) of 0.000011, Mean Absolute Error (MAE) of 0.002617 and R^2 of 0.999737 in under 2 minutes. This model was selected to be used for TL to the East Rand and West Rand datasets.

Ensemble methods (bagging, boosting and stacking) outperformed individual baseline models. However, when comparing stacking ensemble ML that combined all the baseline models with stacking ensemble ML that only combined the best performing models, it was found that there was no significant improvement in excluding bad models from the stack as long as the good models were included. In one case, it was actually beneficial to include the bad performing models. All models were trained in under 2 minutes which proved the benefit of using ML approaches compared to traditional approaches. The treated water data was highly uncorrelated such that model training was unsuccessful with the highest achievable R^2 value being 0.14, thus, no treated water model was available for TL.

TL was successfully conducted on the cleaned and modelled East Rand AMD dataset using the Central Rand (Pump B) stacking regressor and a high level of accuracy with respect to Mean Square Error (MSE), Mean Absolute Error (MAE) and R^2 (MSE:0.00124, MAE:0.0290 and R^2:0.963) between the predicted and true sulphate values was achieved. This was achieved despite a marked difference in the distributions between the Central Rand and East Rand datasets which further proved the power of utilizing ML for water data. TL was successful in imputing missing values in the West Rand dataset following prediction of sulphate levels in the cleaned and modelled West Rand AMD and treated water datasets. No true values for

sulphate levels in the West Rand dataset were given, as such, accuracy comparisons could not be made. However, a general baseline idea of the amount of sulphate present in the West Rand treatment plant could now be understood.

The sulphate levels in all three treatment plants (Central Rand, East Rand and West Rand) were found to greatly differ from each other with the Central Rand having the most normal distribution, the East Rand having the most precise distribution and the West Rand having the most variable distribution. Whilst the sulphate levels in the treated effluent waters could not be reliably predicted due to inherent issues (e.g., analytical inaccuracies and inconsistences) and poor correlations within the treated water datasets, sulphate levels in all three of the untreated AMD datasets were successfully predicted with a high degree of accuracy. This underpinned the observation made previously about the discrepancies between treated and untreated water.

The study has shown that it is possible to impute missing values in one water dataset and use transfer learning to complete and consolidate another similar, but scarce, dataset(s). This approach has been lacking in the water industry, resulting in the reliance and use of traditional methods that are expensive and inadequate. This has caused water practitioners to abandon scarce datasets, thus, losing potentially valuable information that could be useful for water remediation and recovery of valuable resources from the water.

As a spin off from the study, it has been indicated that automation of such data analysis is possible. This was achieved by developing a Graphical User Interface (GUI) for ease of use of the SE-ML model by those with little to no programming background nor ML knowledge e.g., the laboratory staff at the AMD treatment plants. This can also be used for teaching purposes in academia.

DEDICATION

To my parents, Nasreen Hasrod and Ebrahim Ahmed Hasrod,

Thank you for all your sacrifices and support.

I could never have accomplished anything without you.

I love you

ACKNOWLEDGEMENTS

I would firstly like to thank God for His fortunes, favours and knowledge that He has blessed me with.

To my parents, Nasreen Hasrod and Ebrahim Ahmed Hasrod, thank you for all your sacrifices and support. You are undoubtedly my pillar of strength. Thank you for your patience in bringing me up when I was little, I could never have accomplished anything without you.

To my family and friends, thank you for all your prayers and words of encouragement. It was indeed those prayers that got me though it all.

To my supervisor, Prof. Hlanganani Tutu, thank you for all your words of wisdom and guidance. I am especially thankful for Prof's insightful discussions and patience when teaching me. Prof always availed himself whenever I was in need, and I am forever grateful for that. I learnt both academic and life lessons with Prof, and I will carry that with me for the rest of my life. It has been an honour being Prof's student. Thank you, Prof, for taking me as your student.

To my co-supervisor, Dr. Yannick Nuapia, thank you for your guidance, support and ideas.

To Prof. Helder Marques, thank you for always taking the time out to advise and assist me. From my very first lecture on my very first day of university in 1st year till now, Prof has always taken a personal interest in me, thank you Prof for all your advice and wise words.

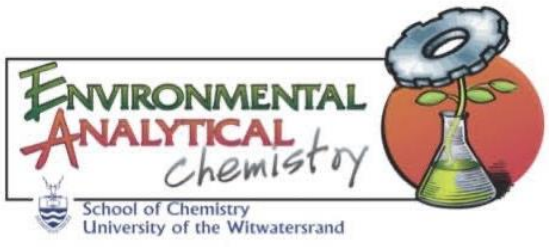
To Prof. Neil Coville, thank you for being one of my biggest supporters who has always encouraged me to give it my best.

To the Environmental Analytical Chemistry group, thank you for welcoming me with open arms. Your encouragement well wishes and support has indeed allowed me to flourish in your group. Thank you for always bringing a smile to my face whenever I needed it.

I am thankful to the Trans Caledon Tunnel Authority (TCTA) for providing us with data.

I am thankful to the University of the Witwatersrand, School of Chemistry for allowing me to demonstrate to the 1st-year, 2nd-year and 3rd-year undergraduate students and co-supervise an honours student. It has taught me patience and empathy and I have met many wonderful people with whom I shared many laughs with.

It is not the end that matters, but rather it's the journey. I am thankful that to have met so many wonderful new people, gone on new adventures, made new memories, did things I ordinarily would have never done and I've seen some wonderful sunsets along the way…



وَجَعَلْنَا مِنَ ٱلْمَآءِ كُلَّ شَىْءٍ حَىٍّ

"And We created every living thing from water."(Quraan, 21:30)

PUBLICATIONS

This dissertation is based on the following publications:

1. **T. Hasrod**, Y. Nuapia and H. Tutu (2024): "ChatGPT Helped Me Build a Chemistry App, and Here's How You Can Make One Also". *Journal of Chemical Education.* https://doi.org/10.1021/acs.jchemed.3c01170. Impact Factor: 3.208.
   Contribution of candidate: conducted the app development, developed the codes, wrote the draft article.

2. **T. Hasrod**, Y. Nuapia and H. Tutu (2024): "Comparison of individual and ensemble machine learning models for prediction of sulphate levels in untreated and treated Acid Mine Drainage". *Environmental Monitoring and Assessment*. https://doi.org/10.1007/s10661-024-12467-8. Impact Factor: 3.307.
   Contribution of candidate: conducted the data analysis and modelling, developed the codes and wrote the draft article.

3. **T. Hasrod**, Y. Nuapia and H. Tutu: "The application of transfer machine learning to predict and impute missing sulphate levels in different Acid Mine Drainage treatment plants". *Cleaner Water*. (Submission ID CLWAT-D-24-00049, under review).
   Contribution of candidate: conducted the data analysis and modelling, developed the codes and wrote the draft article.

PRESENTATIONS AND AWARDS

The work has been presented at the following symposia and conferences:

1. FameLab 2023 – University of the Witwatersrand internal heat (**Flash Talk**), Johannesburg (Gauteng), South Africa, 5 May 2023.

2. FameLab 2023 – National South African Semi-Finals (**Flash Talk**), Johannesburg (Gauteng), South Africa, 14 September 2023.

3. FameLab 2023 – National South African Finals (**Flash Talk**), Pretoria (Gauteng), South Africa, 21 September 2023.

4. AI Expo Africa 2023 (**Poster**), “Prediction of sulphate levels in Acid Mine Drainage treatment plants using individual, ensemble and transfer machine learning algorithms”, Sandton (Gauteng), South Africa, 1-3 November 2023.

5. FameLab 2023 – International Finals (**Flash Talk**), Cheltenham (England), United Kingdom, 24 November 2023.

This work has been awarded the following:

1. **Winner** of 2023 FameLab University of the Witwatersrand internal heat (**Flash Talk**), chosen to represent the University of the Witwatersrand at the national (South African) FameLab finals, Johannesburg (Gauteng), 5 May 2023.

2. **Advanced to National South African Finals** of 2023 FameLab South Africa (**Flash Talk**), Johannesburg (Gauteng), 14 September 2023.

3. **Winner** of 2023 FameLab National South African Finals (**Flash Talk**), chosen to represent South Africa at the International FameLab finals, Pretoria (Gauteng), 21 September 2023.

4. **Finalist** of 2023 FameLab International Finals (**Flash Talk**), Cheltenham (England), United Kingdom, 24 November 2023.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMNS

AI	Artificial Intelligence
AMD	Acid Mine Drainage
DL	Deep Learning
DT	Decision Tree Regressor
EC	Electrical Conductivity (mS/m)
EDA	Exploratory Data Analysis
EN	Elastic Net
ER	East Rand
EU	European Union
FE	Iron (mg/L)
GUI	Graphical User Interface
IC	Ion Chromatography
ICP-AAS	Inductively Coupled Plasma Atomic Absorption Spectroscopy
ICP-AES	Inductively Coupled Plasma Atomic Emission Spectroscopy
IQR	Inter-Quartile Range
KNNR	K-Nearest Neighbours Regressor
LASSO	Least Absolute Shrinkage and Selection Operator
LR	Linear Regression
MAE	Mean Absolute Error
ML	Machine Learning
MLP	Multi Layer Perceptron Artificial Neural Network Regressor
MN	Manganese (mg/L)
MSE	Mean Squared Error
NaN	Not a Number
NMSE	Negative Mean Squared Error
PCA	Principal Component Analysis
R^2	Coefficient of Determination
RD	Ridge Regressor
RF	Random Forest Regressor
RFE	Recursive Feature Elimination
SDG	Sustainable Development Goal
SE-ML	Stacking Ensemble Machine Learning
SSM	Squared Sum error of the Mean line
SSR	Squared Sum error of the Regression line

SUL	Sulphate (mg/L)
SVR	Support Vector Regression
TCTA	Trans Caledon Tunnel Authority
TDS	Total Dissolved Solids (mg/L)
TEMP	Temperature (°C)
TL	Transfer Learning
TSS	Total Suspended Solids
TW	Treated Water
UN	United Nations
WCCS	Within-Cluster Sum of Squares
XG	Extreme Gradient Boosting Regressor

CHAPTER 1 : BACKGROUND

1.1.1 INTRODUCTION AND LITERATURE REVIEW

1.1.1 Water

1.1.1.1 The Essence of Life

Water is a basic necessity that every living creature on Earth depends on.[1] Whilst globally society and the economy are evolving at alarming rates with the advent of new technologies, the basic necessity of having access to clean water will always be a fundamental pillar in the survival of generations.[2] It is, therefore, imperative that clean water be available to communities in order to ensure that basic human rights such as cleanliness and good health are fulfilled to the highest of standards.[3,4]

The United Nations (UN) has a specific Sustainable Development Goal (SDG) – SDG 6 – dedicated to "Ensure availability and sustainable management of water and sanitation for all".[5] As seen in Figure 1.1a, current forecasts predict that by 2030 approximately 1.6 billion people will lack safely managed drinking water, 2.8 billion people will lack safely managed sanitation and 1.9 billion people will lack basic hand hygiene facilities.[6] Access to clean water also affects other aspects of life such as the environment and, therefore, has a great impact on climate change. Also seen in Figure 1.1b, is SDG 13 which addresses the importance of combating climate change in order to prevent natural disasters, prevent droughts and rising sea levels as well as save marine life by limiting the rise in sea temperatures.[7]

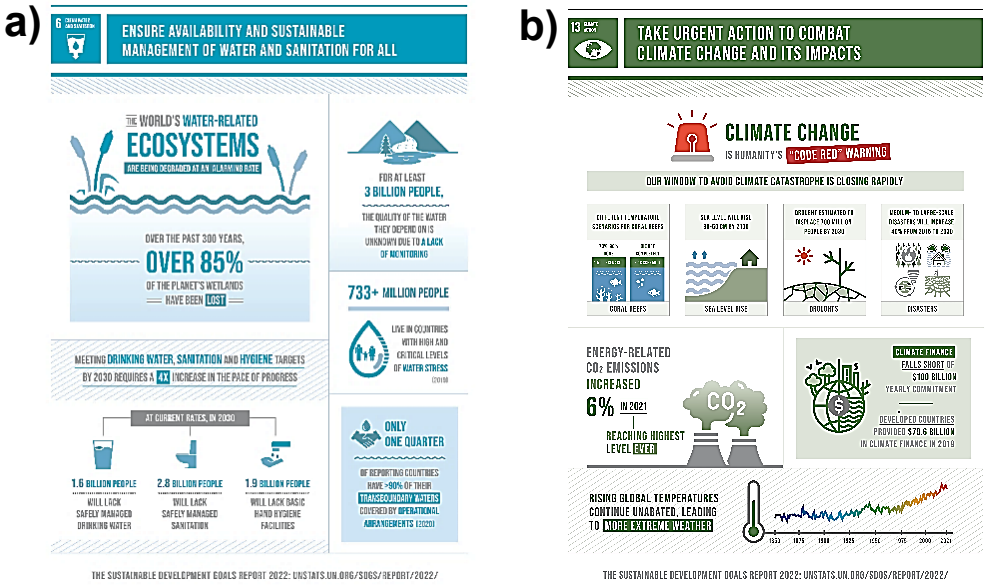


Figure 1.1 : a) UN SDG 6 showing the importance of water, b) SDG 13 showing the necessity to take action to prevent drastic consequences of climate change.[6,7]

1.1.1.2 Sustainable living - The need for Wastewater Treatment Plants and Circular Economy.

Natural sources of clean drinking water such as springs, rivers, dams, etc. will not be sufficient to meet the future demands of water due to its special and temporal distribution in conjunction with the ever-increasing population and rapid technological advancements.[8,9] It is predicted that by 2050 clean water scarcity is due to increase by 22-34 % as a result of population growth, economic development and changes in consumption patterns.[9] It is, therefore, imperative to create a sustainable living environment by reviewing the ways in which one recycles and disposes of by-products.[8] One of the main methods of doing such is to treat wastewater in order to reclaim clean treated water and recover nutrients and minerals from water-based waste.[10] As seen in Figure 1.2, when such a reliable and efficient circular economy is created, an initial waste product can be repurposed and used as a cheap feedstock for another process which creates other benefits such as the recovery of valuable by-products and the access to clean water.[11]

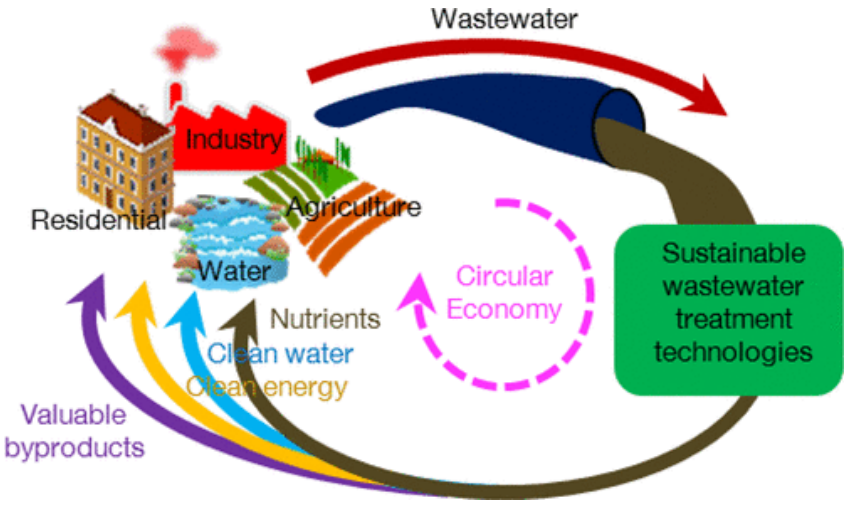


Figure 1.2 : An example of a circular economy created from the implementation of sustainable wastewater treatment technologies.[11]

Such circular economy will allow for the generation of not only clean drinkable water, but also has several economic benefits such as the creation of new jobs due to the need for manpower to operate treatment plants, research and development and manufacturing in order to produce the technology and equipment needed.[12] It was predicted that a 1 % rise in the growth of the wastewater industry in the European Union (EU) would result in the creation of twenty thousand new jobs.[10] Implementation of a circular economy has several economic and environmental benefits that include reduced unemployment rates, a higher skilled workforce, a decrease in environmental pollution and the generation of valuable products.[13] Creation of such a circular economy would contribute towards accomplishing UN SDG 12 “Ensure sustainable consumption and production patterns” by recycling and repurposing waste.[14,15]

1.1.1.3 South Africa – a Mining Centred Economy and its implications on the Environment and Human Rights.

Historically, South Africa had a mining centred economy and as such, most of the country's current economic growth comes from its abundance of mineral resources such as gold, platinum,etc.[16] A result, thereof, is the abundance of pollution and in particular wastewater produced by mining activities which plays a pivotal role in environmental pollution due to its increased heavy metal concentration and acidic nature.[17,18] The appearance of toxic ponds (as shown in Figure 1.3) due to daylighting (the restoration of the above-ground presence of buried rivers and streams)[19] results in serious environmental concerns that infringe on human rights since these ponds often contaminate clean water sources.[20,21] Such toxic ponds mainly affect the poorer and underdeveloped communities, such as informal settlements that live near mines, on top of mine tailing dumps or on reclaimed land after a mine has closed.[20] The negative effects of these toxic ponds include posing as a major human health risk, poisoning of healthy soil used for agriculture, destruction of wildlife and ecosystems and the weaking of building foundations which results in structural collapses of houses or shacks.[20] These toxic ponds are known as Acid Mine Drainage (AMD) and is discussed in further detail in the following section.

Figure 1.3 : AMD pond daylighting in South Africa (note the dark red colour that indicates its acidity and heavy metal content).[22]

1.1.2 Acid Mine Drainage (AMD)

One of the main polluting factors related to mining activities is the production of Acid Mine Drainage (AMD). As depicted in Figure 1.4, AMD is formed as a result of natural runoffs from a mine dump after rainfall, disposal of water used during operations by the mining company or the eventual decanting and above ground flooding of abandoned and disused mine shafts that fill up with water.[20] It could also be as a result of water leaching through the tailing dump fractures into the voids and aquifers of the water table below.[23] This results in toxic leachates that are rich in heavy metals (Fe, Al, Mn, etc), metalloids (e.g. arsenic) and sulphates which

acidify the pH of the water table by adjusting it to a pH of about 3-5.[24] This water which has an acidic pH, high heavy metal and high sulphate concentration is what is known as AMD.

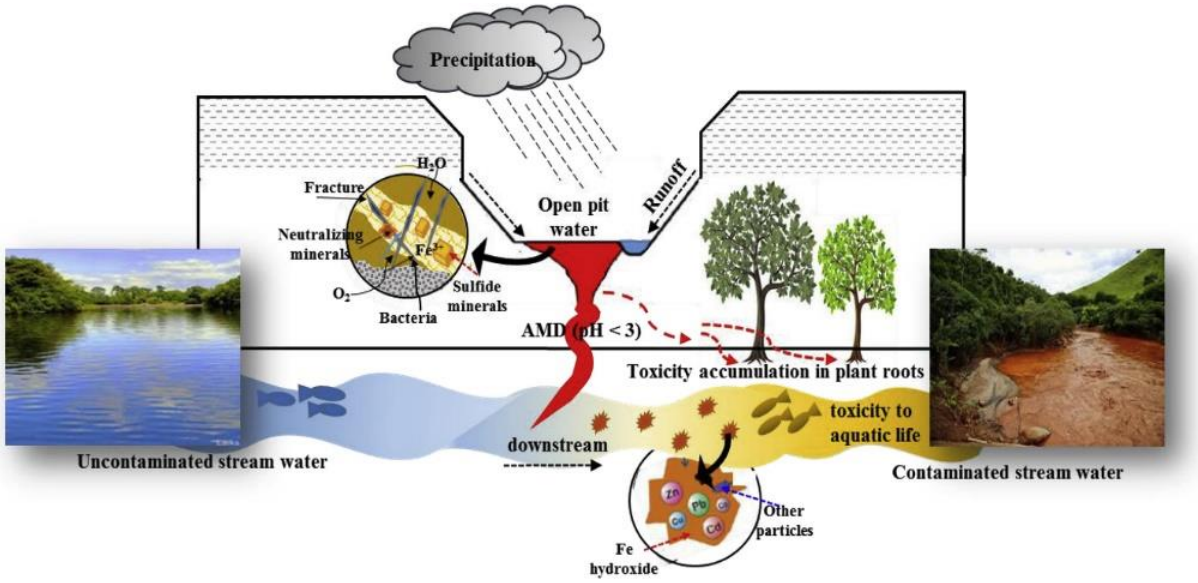


Figure 1.4 : Formation of Acid Mine Drainage (AMD) and how it contaminates the water table below.[23]

On a molecular level, AMD is formed through a series of incredibly complex chemical reactions that involve sulphide minerals (examples of which are given in Figure 1.5) present in ore deposits that upon exposure to oxygen present in air and water undergo oxidation.[17] This exposure usually happens during mining, excavations or other earth moving processes that result in the ore deposits being exposed to the external environment.[17] Of these sulphide minerals, iron sulphide (FeS_2) which has two polymorphs, namely, pyrite and marcasite, has the most potential to produce AMD upon oxidation.[25]

Figure 1.5 : Sulphide minerals and their chemical formulae.[26]

The chemical reactions of iron sulphide, and in particular, pyrite to form AMD can be broken down into three main steps as indicated in Table 1.1. These reactions are dependent on

numerous factors such as environmental conditions and can involve multiple pathways such as chemical, biological (which involves the use of bacteria) and electrochemical.[27]

Table 1.1 : Chemical reactions of iron sulphide (pyrite) to form AMD.[17,18]

Step	Chemical Reaction	Chemical Equation
1	- Oxidation of iron sulphide - Enhanced oxidation of other sulphide minerals using the ferric ion (Fe^{3+})	$2FeS_2 + 7O_2 + 2H_2O \rightarrow 2Fe^{2+} + 4SO_4^{2-} + 4H^+$ (1.1) $FeS_2 + 14Fe^{3+} + 8H_2O \rightarrow 15Fe^{2+} + 2SO_4^{2-} + 16H^+$ (1.2)
2	Oxidation of ferrous iron (Fe^{2+})	$4Fe^{2+} + O_2 + 4H^+ \leftrightarrow 4Fe^{3+} + 2H_2O$ (1.3)
3	Hydrolysis and precipitation of ferric iron and minerals	$Fe^{3+} + 3H_2O \leftrightarrow Fe(OH)_3\downarrow + 3H^+$ (1.4)

As shown in Table 1.1, each of the reactions produce hydrogen ions (H^+) which contribute to significantly lowering the pH of the water, thereby, turning it acidic. The quantities of iron sulphide present in mining waste determine the characteristics of AMD since these are what ultimately lead to the heavy metal content and acidic nature.[28] Other sulphide minerals (e.g., chalcopyrite and sphalerite) are also capable of undergoing oxidation and these would further contribute to the heavy metal content of the AMD.[29,30]

It is of utmost importance that AMD treatment processes must be implemented to reduce its toxicity (heavy metal content and acidity) such that the treated water may be suitable to be discharged into nearby streams, rivers and wetlands. It would also be economically worthwhile to extract and recover any valuable metals, minerals and compounds for commercial use from the AMD through the wastewater treatment process since these may have precipitated out during the process.

1.1.2.1 AMD treatment processes

AMD treatment processes can be classified into two different categories *viz*, active (which requires the continuous feeding of inputs to sustain the process) and passive (which is mainly self-sustaining and requires very little input once in operations), and a schematic of each class and sub-division can be found in Figure 1.6.[24] Active treatment of AMD involves the addition alkaline materials such as lime (CaO), calcium carbonate ($CaCO_3$), sodium carbonate ($NaCO_3$), sodium hydroxide (NaOH) and magnesium hydroxide ($MgOH_2$) in order to raise the pH of the water (i.e. neutralize the waters), encourage oxidation of ferrous iron and precipitate metal hydroxides and carbonates.[31] Passive treatment involves the use of natural and constructed wetland ecosystems which requires low maintenance costs, however, it is very expensive to initially set up.[24] As such, this project will focus on the abiotic active treatment of AMD.

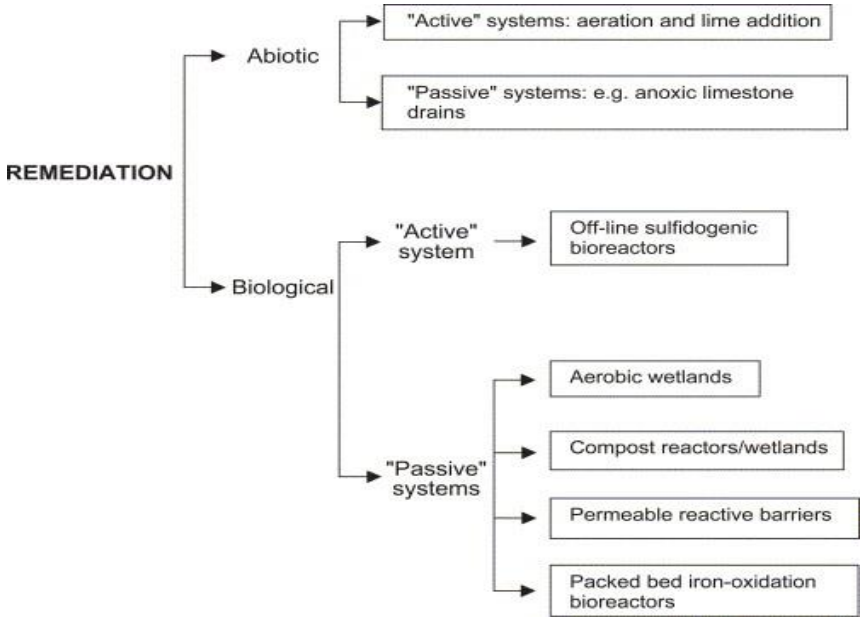


Figure 1.6 : Remediation (treatment) strategies of AMD and its sub-divisions.[24]

A schematic of an active process can be found in Figure 1.7 where AMD and lime slurry are fed to the process as inputs and the products are the “Treated AMD” and the “Waste Sludge”. Active treatments are the most widespread and whilst there is high cost involved regarding the continuous inputs of alkaline neutralizing agents, the waste sludge produced is iron-rich and can contains various other metals.[24] The sludge contains high metal concentrations of iron (Fe), cobalt (Co), manganese (Mn) and aluminium (Al) and can be repurposed in many ways such as extracting pigments which can be used for paintings.[32] This project will further investigate the constituents of the untreated AMD and treated water which still contains high concentrations of calcium (Ca), magnesium (Mg) and in particular, sulphates (SO_4^{2-}).

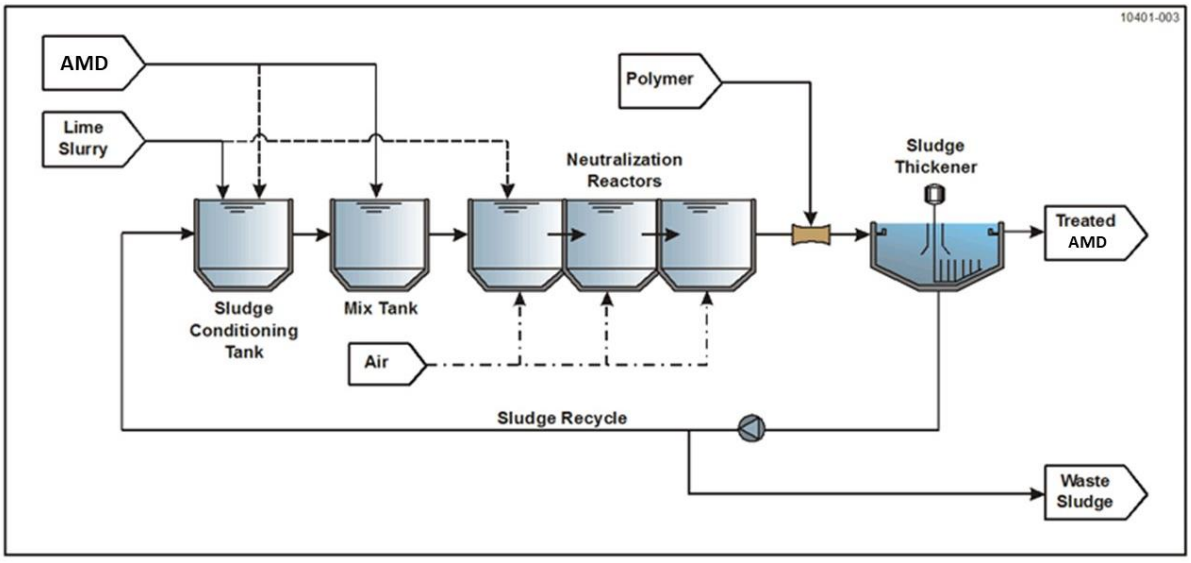


Figure 1.7 : Schematic diagram of an active AMD treatment process which uses limestone.[33]

1.1.2.2 Sulphates and valuable octathiocane present in treated waters

Treated water is discharged into streams, wetlands and rivers and has high sulphate (SO_4^{2-}), calcium (Ca) and magnesium (Mg) concentrations.[34] As seen in Figure 1.8, in reducing environments, such as wetlands, the presence of sulphur reducing bacteria reduce the SO_4^{2-} to hydrogen sulphide (H_2S).[35] Hydrogen sulphide can then be further oxidised by a variety of complex reactions involving transition metals and radicals to form crystalline elemental sulphur which is known as octathiocane (S_8).[36] This S_8 is what gives rise to the yellowish powder seen in some wetland areas since it precipitates out as a yellow powder (Figure 1.9a).

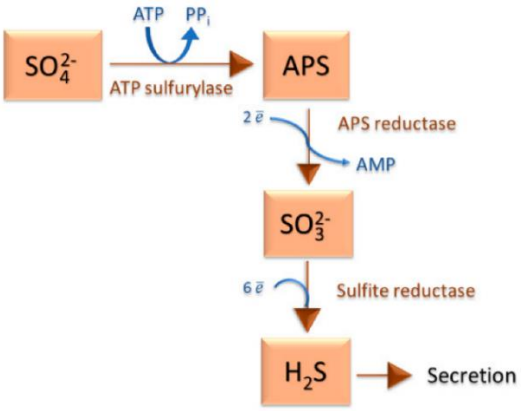


Figure 1.8 : Reduction of sulphate (SO_4^{2-}) to hydrogen sulphide (H_2S) via the presence of sulphur reducing bacteria.[35]

As seen in Figure 1.9b, octathiocane is comprised of eight sulphur atoms bonded together in a ring conformer and as seen in Figure 1.10, it adopts the monoclinic unit cell with a P 2/n space group.[37] It is extremely beneficial to extract octathiocane since it is a commercial product the most commonly used form of pharmaceutical sulphur which has uses as bacterial metabolite and fungal metabolite, antiseptic and disinfectant due to its potency.[38,39] Octathiocane acts as a keratolytic agent and is used in various cosmetics such as lotions, soaps and bath additives for the treatment of acne and dermatitis since it can kill fungi, mites and other parasites.[40]

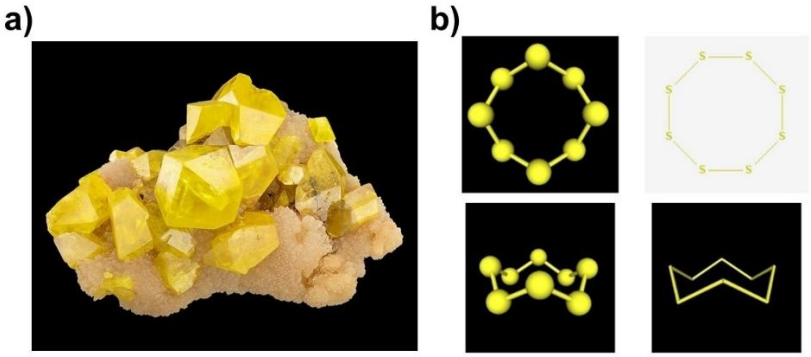


Figure 1.9 : a) Mineralogical form of octathiocane[41] *and b) Molecular structure and conformers of octathiocane (S_8)*[37]

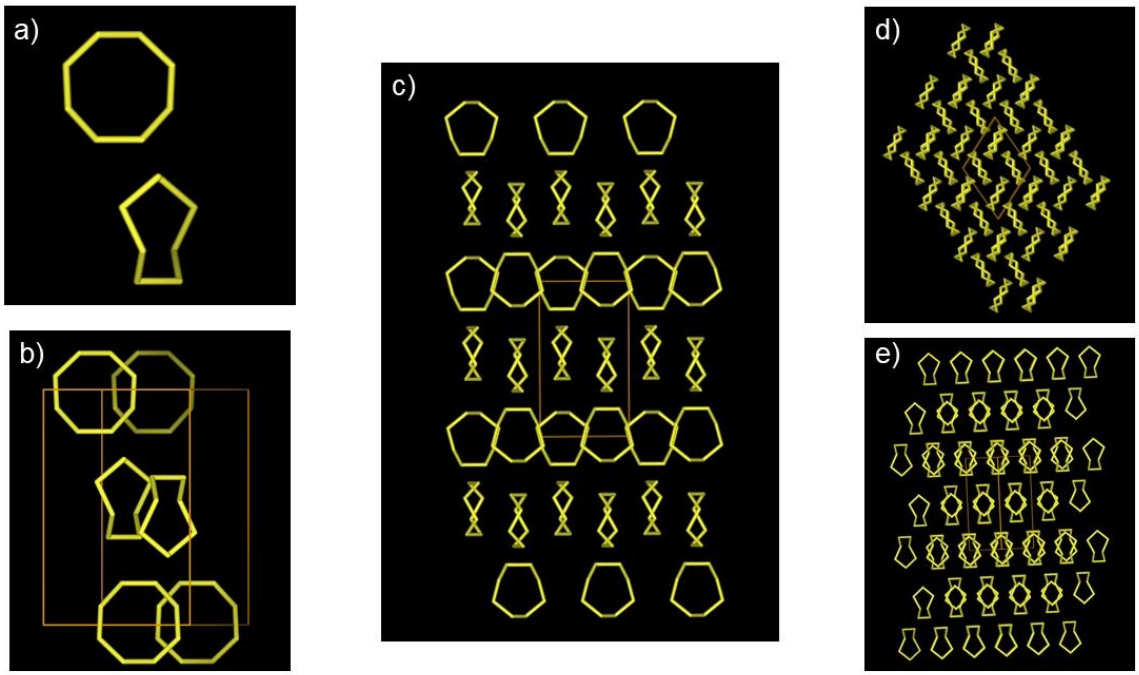


Figure 1.10 : a) Crystal structure, b) unit cell and c)-e) packing of octathiocane.[37]

It is, therefore, extremely valuable create a circular economy that extracts octathiocane from untreated and treated AMD (Figure 1.11) due to its wide variety of uses and its low cost since it is essentially derived from wastewater.

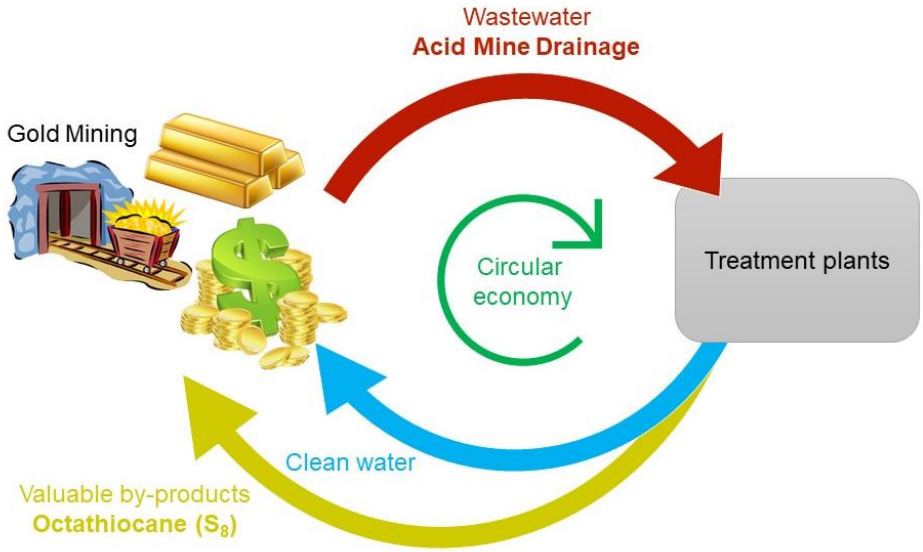


Figure 1.11: Creation of circular economy by extracting octathiocane from untreated and treated AMD.

1.1.2.3 Comparison of traditional analytical methods used to determine sulphate concentrations in AMD and their associated challenges

In order to determine the possible quantities of sulphate in the AMD effluent water and, thus, the approximate amount of recoverable octathiocane (S_8) present, accurate and reliable methods of analysis must be carried out. Traditional analytical methods used to determine sulphate concentrations include the turbidity, Ion-Chromatography (IC), Inductively Coupled Plasma Atomic Emission Spectroscopy (ICP-AES), spectrophotometry, gravimetry and titrimetry.[42,43] An in-depth review of these methods as discussed by Reisman *et* al.[42] and Roy *et al*.[43] with their associated advantages and disadvantages can be found in Table 1.2.

Table 1.2 : Comparison of traditional analytical methods used to determine sulphate concentrations in mining water.[42,43]

Analytical Method	Description	Advantages	Disadvantages
Turbidity	This method utilizes the precipitation of sulphate ions out as insoluble barium sulphide.	- Suitable for field work since testing kit is portable. - Easily available commercial kit.	- Requires the addition of stabilizing agents such as thymol, gelatine and polyvinyl alcohol. - Barium complexation is prone to interferences from other metal ions and this may require a complexation agent such as EDTA to be used. - Filtration may be required for highly coloured and turbid samples in order to remove interferences. - Testing kit precision and accuracy has not been validated. - Limited range of quantification and requires several dilutions.
IC	The separation of anions occurs on the basis of their relative affinities and the measured by detecting the suppressed conductivity.	- Allows for direct determination of sulphate concentrations.	- Requires validation by other analytical methods. - Prone to acidification interferences. - Limited range of quantification and requires several dilutions.

Analytical Method	Description	Advantages	Disadvantages
ICP-AES	This spectral method can determine the sulphate concentration in a sample by measuring the emitted colour (indicative of element identity) and intensity of colour (indicative of element concentration) during its burning in an inert gas (usually argon) plasma.[44]	- Allows for analysis of high sulphate concentrations in wastewater. - Robust method and capable of determining sulphate concentrations even in the presence of other interfering ions at low pHs.	- Assumes negligible sulphur species with low oxidation states. - Indirect way of measuring sulphate concentrations since the output is total sulphur concentration which must then be converted to sulphate concentration using a conversion factor of 2.996 mg/L. - Prone to interferences from calcium and hydrochloric acid. - Limited range of quantification and requires several dilutions.
Spectrophotometry	Barium chloroanilate is added to react with sulphate ions and liberate them as chlorianilic acid. The UV absorbance of chlorianilic acid at specific wavelengths is measured and related to sulphate concentrations via the Beer-Lambert law.	- Allows for rapid sample throughput. - Achieves a reasonable accuracy.	- Prone to interferences by cations which from insoluble chloroanilates. - Trace amounts of phosphates and arsenates can cause interferences due to their high optical densities. - To remove interfering cation species, a cation-exchange column must be used in conjunction with a long high-temperature evaporation

Analytical Method	Description	Advantages	Disadvantages
			procedure. This is time consuming and requires a lot of heat.
Gravimetry	This method relies on the precipitation of sulphate ions out as insoluble barium sulphate.	- Simple method which does not require any specialized equipment.	- Requires a lot of skill and time in determining the end-point. - Time consuming. - Prone to co-precipitation of calcium sulphate and magnesium sulphate.
Titrimetry	This is the traditional titrimetric approach whereby concentrations of added reagents are measured. In the case of sulphur, an ethanolic solution with thorin as the indicator is used.	- Simple and easy to use method. - Does not require any specialized equipment.	- Requires strict experimental conditions for accurate and precise measurements. - Requires the prior removal of cations from solution by IC which is time consuming.

Overall, these methods are extremely time-consuming, expensive, make use of specialized equipment and utilize several expensive and hazardous chemicals (e.g. hydrochloric acid and hydrofluoric acid).[45] Gravimetry, ion chromatography, spectrophotometry and potentiometry are some of the most common methods used to determine sulphate values ,however, they require a large amount of sample preparation, require costly lab equipment and are extremely laborious.[46] Due to inefficient sulphur removal,[47] high sulphate values present in effluent waters may also be a reason for inaccurate result since these experimental values do not normally take into account the thermodynamic equilibrium of charged species.

Therefore, sulphate values are usually missing from large datasets that contain several other water quality parameters related to the untreated AMD and treated water of the AMD treatment plant. This poses a huge problem since the missing sulphate values are normally imputed by averaging the few values that were experimentally determined without taking into consideration the rest of the data associated with that specific missing value. This brings the accuracy and reliability of the dataset into question and, therefore, any conclusions regarding the water quality are brought into dispute since the validity of the individual values are questionable.

Therefore, there exists a need to accurately predict the missing sulphate values in the large datasets obtained from the AMD treatment plants in order to determine the potential economic value that lies in its AMD and effluent waters. Artificial Intelligence (AI) and in particular, Machine Learning (ML) holds the power of doing such in a cost effective, timely and accurate manner and as such was explored in this project. Further, transfer learning was conducted to generalise the results to other similar AMD treatment plants where data is sparse.

1.1.3 Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL)

Basically put, **Artificial Intelligence (AI)** is the science and engineering of creating intelligent machines and programs in order to understand human intelligence.[48] In a nutshell, AI combines computer science and large, complete datasets in order to enable problem-solving.[49] Such problem solving consists of using subsets of AI such as Machine Learning (ML) and Deep Learning (DL) to create accurate systems/models to make predictions or classifications based on input data.[49] The United Nations says that “Artificial Intelligence ensuring human rights is at the heart of the Sustainable Development Goals,”[50] due to its big data processing capabilities and makes human rights analysis, programming and planning much easier for governments.[51] At the UN 2023 Water Conference which took place in March 2023, a great deal of focus was placed on using AI in water data analytics in order to achieve the other SDGs such as energy, climate, cities, environment, etc.[52]

Machine Learning (ML) is a subfield of AI that uses data and algorithms to "imitate the way humans learn, gradually improving its accuracy".[53] In the field of data science, ML is critical to identify hidden features and deliver key insights into data mining projects (that uses structured data) via the use of statistical methods and mathematical algorithms.[53–56] Feature extraction and data pre-processing are done manually by a human in order to identify salient features of the data-set.[56] These key insights are absolutely vital in order to understand the system one is working with and, thus, make appropriate decisions as to what should be the next step or how much value there is in pursuing a certain idea.

Deep learning (DL) is a further sub-field of Machine Learning and is capable of dealing with unstructured data in its unedited (i.e., raw) form e.g. texts or images and automatically (and rapidly) identifies distinguishing features in order to classify such data.[53] Deep-learning eliminates a lot of data pre-processing and feature extraction as opposed to ML which requires a great deal of human input.[56]

A typical schematic representing AI and its subfields can be found in Figure 1.12 in order to better understand the nesting of each field.

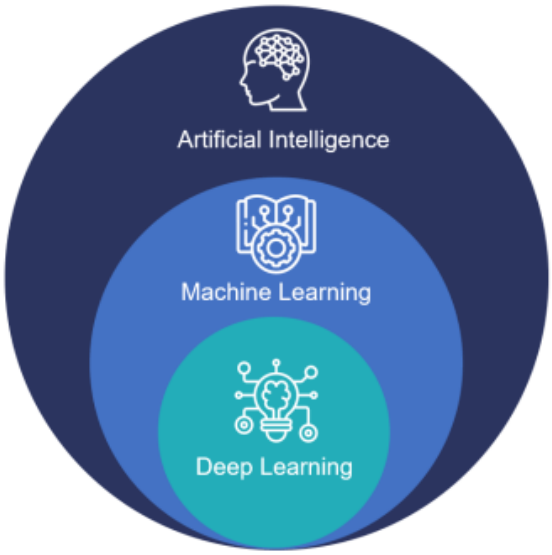


Figure 1.12 : Schematic diagram depicting Artificial Intelligence (AI) and its subfields, namely Machine Learning (ML) and Deep Learning (DL).[57]

When looking at the applications of ML in environmental assessment, ML has successfully been used to predict ammonia levels in groundwater to understand the nitrogen reduction pathways and nitrogen conservation and loss.[58] An adaptive neuro-fuzzy inference system was used to predict the reaeration coefficient of streams in India in order to understand to sources of organic loading such that waste and water resource management schemes could be developed.[59] ML was used to model soil moisture and its effect on slope stability in order to identify triggers of shallow slope landslides.[60] A forward feeding dynamic neural network

was used to develop an early warning system for reservoir water management and maintenance.[61] It was shown that an electrochemical system when used in conjunction with ML was more efficient at removing antibiotics from wastewater.[62] A double machine learning model and time series clustering was used to understand climate change and lake response.[63] In order to assess the determinants of environmental sustainability, ML was used to determine the factors associated with CO_2 emissions and their related correlations.[64] ML was used to provide a cost-effective alternative to traditional modelling procedures in predicting daily groundwater levels for sustainable crop production.[65] ML was also used to predict nitrate levels cost-effectively, reliably and quickly in groundwater.[66]

1.1.3.1 Supervised and Unsupervised machine learning

Within ML there are two approaches that can be used to carry out data science methods, viz. Supervised and Unsupervised ML. **Supervised ML** comprises of using labelled datasets to train (i.e. supervise) algorithms to classify or predict outcomes.[67] The use of labels allows models to accurately learn trends in the data by correcting for any errors made since a true value is known[67], and this can be used in two ways during data mining:

1. **Regression** – the model learns the relationship between dependant and independent data in order to **predict** numerical values based on different data points.[67]
2. **Classification** – an algorithm is used to assign test data into specific categories.[67]

Unsupervised ML uses algorithms to analyse and cluster unlabelled data sets without any human intervention in order to identify hidden features and relationships.[67] Inherent structures in the data can be identified by using one these three main categories:

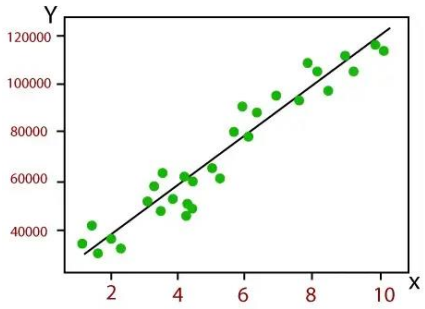
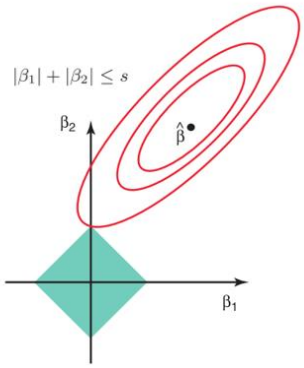
1. **Clustering** – this technique groups unlabelled data based on their similarities or even their differences.[67]
2. **Association** – this method finds relationships between values in a dataset by applying different rules and recommendations.[67]
3. **Dimensionality reduction** – when the number of features (i.e., variables) in a given dataset is extremely high, this is used to reduce the number of dimensions to a more manageable size whilst still maintaining a high level of data integrity.[67]

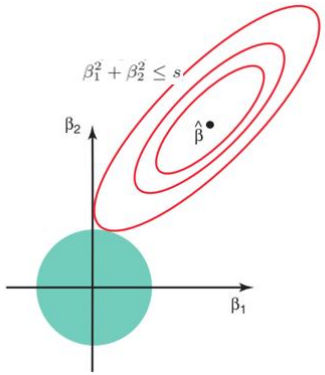
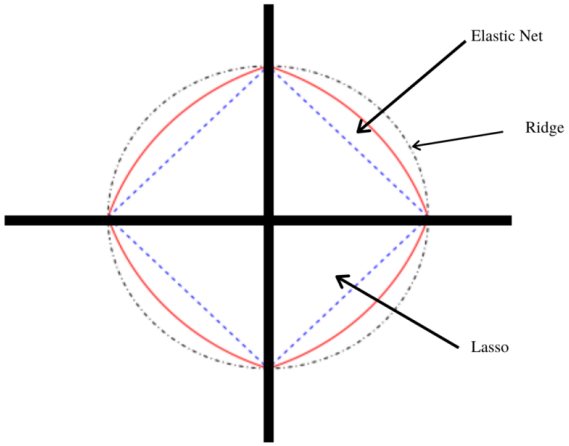
The main aim of this project was to accurately predict the sulphate levels present in untreated and treated Acid Mine Drainage given the known, labelled water quality parameters of the influent and effluent water. Thus, supervised machine learning and regression were used in order to achieve the desired goal with a high level of accuracy.

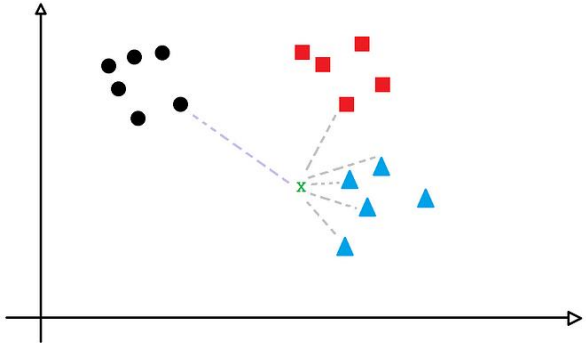
1.1.3.2 Regression models, their algorithms and explanation

There are a variety of regression models that can be used based on the number of input variables, the type of output variable and the shape of the regression line.[68] Regression models are used to determine the relationship between the input variables (the predictors) and the output variable (the target) and are summarized in Table 1.3. All of these models were used to predict sulphate levels and compared to each other in order to determine the best performing model.

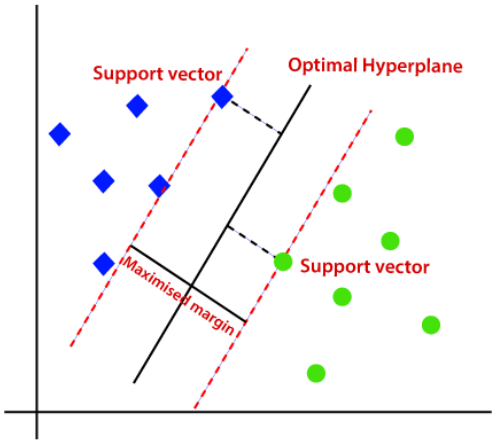
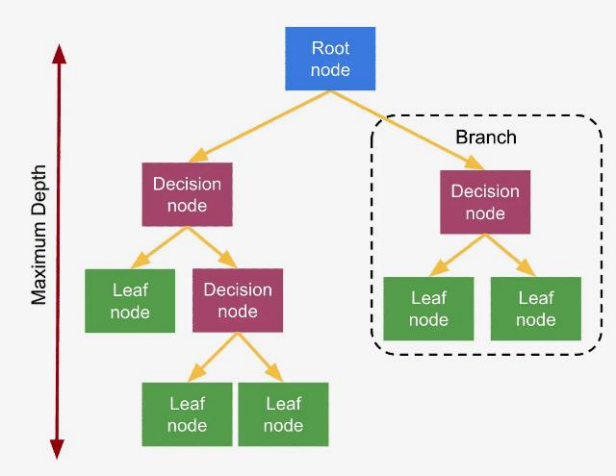
Table 1.3 : Regression models, their explanation, graphical depictions and when to use them.

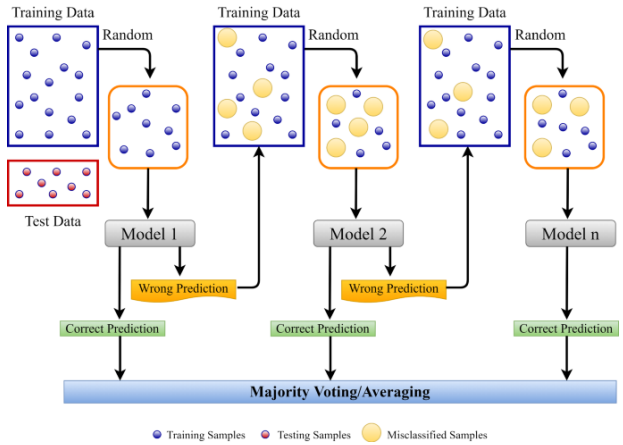
Regression model, graphical depiction and references	Explanation and equation	When / when not to use
Linear (LR)[68–70] 	This model is used when the relationship between the predictor and the target is linear. It can be further broken down into two main types based on the number of predictors, namely, simple linear regression where there is only one predictor, and multiple regression analysis where there are multiple predictors. This model simply uses the equation of a straight line in the form $$y = \beta_0 + \beta_1 x + \varepsilon$$ Where *y* is the target, *x* is the predictor, β_1 is the gradient, β_0 is the y-intercept or bias and ε is the error. The line of best fit is determined by varying the gradient and y intercept in order to minimize the error between the observed and predicted values.	This model is sensitive to outliers in the dataset and, therefore, should not be used on large data sets.
LASSO[68,69,71,72] 	This is also known as L1 and performs feature extraction along with regularization. This is due to the prohibition of the absolute size of the regression coefficient, thus, the value of the coefficient gets nearer to zero. This allows for feature extraction from the dataset in order to build the model. "LASSO" stands for "**L**east **A**bsolute **S**hrinkage and **S**election **O**perator" and shrinks data values towards a central point e.g., the mean.[73] $$\hat{\beta}_{LASSO} = argmin \left\{ \frac{1}{2} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \lvert\beta_j\rvert \right\}$$	Over-fitting is reduced since only required features are used and other features are set as zero. For highly collinear variables, only one variable is picked and the rest are shrunk to zero. This model works well for data that has few variables.[73]

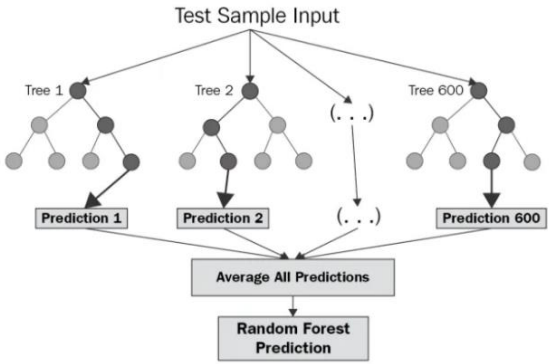
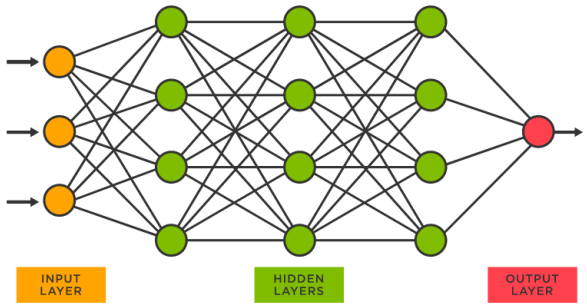
	It is worth noting that this is the same as minimizing the sum of squares of errors with a tuning parameter (λ) to control the shrinkage.[73]	
Ridge (RD)[68,69,71,72]  	This is also known as L2. When there is high bias in the data set due to multicollinearity this technique is used and is also known as Regularization. The λ (lambda) in the equation below reduces the multicollinearity: $$\hat{\beta}_{ridge} = argmin \left\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} {\beta_j}^2 \right\}$$ While both LASSO and Ridge regression constrain coefficients using a penalty factor, LASSO takes the magnitude of the coefficients whilst Ridge takes the square of the coefficients.	Used when there is high correlation between variables when bias and over-fitting is high. A bias matrix is introduced in order to limit over-fitting. This can also be used when the number of predictors greatly exceeds the number of observations.[74]
Elastic Net (EN)[75–77]  	This model performs regularization and feature selection simultaneously by combining the penalties from LASSO and Ridge and learning from their shortcomings to improve its regularization. It improves on LASSO's limitation of only taking one feature when there is multicollinearity. The EN penalty is, $$\hat{\beta}_{EN} = \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^{p} {\beta_j}^2 + \alpha \sum_{j=1}^{p} \lvert\beta_j\rvert \right)$$	This is a powerful technique to be used when the number of predictors (*p*) is greater than the number of samples used. The double shrinkage of coefficients can cause low efficiency in predictability and high bias.

	Where α is a hyperparameter between 0 and 1 (Ridge=0 and LASSO=1) and is used to adjust the weighting of the contribution of each penalty to the loss function.	
k-Nearest Neighbours Regression (KNNR)[78–80] 	This algorithm works by picking a certain point and evaluating its "k" nearest neighbours to find similarities and, thereby, grouping points together based on their similarities. The algorithm works on the bases of evaluating distances between points with the assumption that similar data points lie closely to each other. Thus, the distances between points that are deemed to be "similar" is minimized and the distance between cluster that are deemed to the "different" are maximized. By training a regression function (*f*), targets (y_i) are found using a set of predictors (x') from a set of *N* observations where $N_k(x')$ contains the indices of the k nearest neighbours of x'. $$f_{KNN}(x') = \frac{1}{k} \sum_{i \, \epsilon \, N_{k(x')}} y_i$$ There are different types of distance metrics that can be used and these form the tuneable hyperparameters of the model. The most common distance metric is the Euclidean distance (p=2) which uses real-valued vectors to measure the straight line between the point of interest and another point using the equation below, $$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$	This algorithm does not perform well on large datasets since it is heavily reliant on memory to store all of its data during training. It also does not perform well on high dimensionality data and can be prone to overfitting.

	Another distance that can be used is the Manhattan Distance (p=1) which utilizes the absolute distance between two points as shown in the following equation, $$Manhattan\ Distance = d(x,y) = \left(\sum_{i=1}^{m} |x_i - y_i|\right)$$ One other distance metric is the Minkowski Distance which essentially combines the Euclidean (p=2) and Manhattan Distances (p=1). $$Minkowski\ Distance = \left(\sum_{i=1}^{n} |x_i - y_i|\right)^{1/p}$$ One last distance metric that can be used is the Hamming Distance used for Boolean or string values to identify when lengths do not match. $$Hamming\ Distance = D_H = \left(\sum_{i=1}^{k} |x_i - y_i|\right) \begin{cases} x = y\ , D = 0 \\ x \neq y\ , D \neq 1 \end{cases}$$	
Support Vector Regression (SVR)[81–84]	A hyperplane (i.e., the straight line required to fit the data) is constructed such that data lies on either side of it. The data points that lie closest to the hyperplane are known as the support vectors and these influence the position and orientation of the hyperplane. This hyperplane is used to predict discrete values and find the line of best fit using a threshold value instead of minimizing the error	When there is non-linearity in the data, this model can be very useful. It is also tolerant towards outliers and can easily be updated. However, these models are not suitable

	between the real and predicted value. This threshold value is the distance between the hyperplane and the support vectors (*w*). SVR's are governed via the mathematical equations shown below which constitute the minimization problem: $$f(x) = \begin{bmatrix} w \\ b \end{bmatrix}^T \begin{bmatrix} x \\ 1 \end{bmatrix} = w^T x + b \;\; x, w \epsilon \mathbb{R}^{M+1}$$ $$min_w \frac{1}{2} \|w\|^2$$	for larger datasets and for datasets that contain a lot of noise.
Decision Tree Regressor (DT)[85–87] 	This is a non-linear regressor model that consists of a hierarchy of nodes that are easy to interpret. At each node a mathematical decision is and evaluated to "True" or "False" simply by determining whether a sample follows this rule. This is similar to an if-else ladder and leads to a sample trans versing the depth of the tree via a series of True-False decisions taken. The Tree starts at the root rode whereby only a single decision is taken. Thereafter, the sample is passed through a series of decision nodes and finally terminates at the leaf node where a final decision and, hence, a final value is given as the output. If Y_i is the initial split value and *n* is the number of a value at the current node, then the final value ($\overline{Y_t}$) is, $$\overline{Y_t} = \frac{1}{n} \sum_{i \epsilon n} Y_i$$ In order to determine split points (i.e. nodes) in a regression problem, the DT calculates the Reduction in Variance (RV). RV is	This model is extremely susceptible to small variations the training data and is prone to overfitting.

	also known as the Mean Squared Error (MSE) and is a measure of the estimated purity of the leaf nodes. If the number of "True" values is *x* and the number of "false" values is *y*, $$mean = \frac{(x \times 1) + (y \times 0)}{Total\ number\ of\ values}$$ $$RV = \frac{x(1 - mean)^2 + y(1 + mean)^2}{Total\ number\ of\ values}$$	
Extreme Gradient Boosting Regressor (XG)[88–90] 	This is a collection of **sequential** decision trees that are trained upon each previous tree's mistakes in order to try and correct them. The final learner is essentially a collection of each of the previous "weak" learners which were refined and able to form a "strong" learner, whereby, a final accurate prediction is given. XG also suppresses weights by applying advanced regularization techniques. The loss function calculated to minimize the error between the true and predicted values is given as follows, $$L^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$ Where *l* is the loss function, *t* is the number of iterations and *Ω* is the decision tree complexity penalty.	This algorithm reduces the overfitting present in decision trees and works well when there are imbalanced classes. However, it is a complex algorithm that requires a significant amount of hyperparameter tuning.
Random Forest Regressor (RF)[91–93]	This is essentially a collection of **parallel** decision trees that utilizes bagging and bootstrapping. Bagging is simply an ensemble learning approach whereby several decision trees are collected and trained in parallel at the same time using different sub-sample of the data in order to reduce overfitting. Bootstrapping randomly samples	This model works well with non-linear data, however the model is prone to over-fitting and is not easily interpretable.

	subsets over a given number of iterations and averages the results to obtain an even better result. Random forest regression is governed by the mathematical equation, $$Y = E_0(X, \theta_k)$$ Where *k* indicates the number of trees planted depending on the random variable (θ), *X* is the given inputs, *Y* is the desired outputs which is calculated from taking the average over all the trees.	
Multilayer Perceptron Artificial Neural Network (MLP)[94–96] 	These networks are inspired by the neurons in the brain and mimic the way neurons signal to each other. They are comprised of node layers where input nodes are connected to output nodes via a series of hidden layers. Nodes are connected to each other via a series of weights and thresholds. Node activation is responsible for sending data to the next layer of the network and occurs when the output of the individual node is above the threshold value. If $\bar{X}$ is the sum of input features, *d* is the number of nodes, $\bar{W}$ is the sum of weights and $\bar{W} \cdot \bar{X}$ is the linear function (which is computed at the output node), then the predicted value $\hat{y}$ can be calculated as follows, $$\hat{y} = sign\{\bar{W} \cdot \bar{X}\} = sign\left\{\sum_{j=1}^{d} w_j x_j\right\}$$ Errors are minimized via the updating of weights and node activation occurs when the *sign* is positive.	These models are able to store large volumes of data and work with incomplete data. They also have a high fault tolerance and have parallel processing capabilities. However, these models require high computing power due to their parallel processing and are sometimes hard to explain since their behaviour is sometimes unknown.

When focussing specifically on AMD assessment, ML techniques have gathered much attention due to their robustness and generalizability in various countries.[97] ML image recognition techniques were applied to monitor mining environmental problems using aerial drone technology in Ziyang County, China using SVR, RF and U-Net Neural Network.[98] ML techniques such as Bayesian machine learning and Functional Data Analysis (FDA) were used in Fabero, Spain in order to detect anomalous river flow activities that would serve as early flood warning signs.[99] In northern Tunisia RF, SVM and MLP were used to predict toxic heavy metal content and the behaviour of mine waste in its relation to the dissolution of iron bearing minerals.[100] RF, XG and neural network were used in to predict electrical conductivity and pH in Johannesburg, South Africa in order to improve forecasting procedures that ensure the smooth operation of AMD treatment plants.[101] In the Philippines, a Neural Network with particle swarm optimization was used to improve mapping prediction capabilities for spatial maps to predict ground water quality.[102] The extent of AMD pollution was mapped using aerial drone technology and RF in the Tintillo River, Spain in combination with hyperspectral technology in order to better evaluate contaminated water bodies using remote sensing.[103] A neural network was used to predict mine seepage flow rate in Canada to improve flood control, mine site management and contaminant treatment strategies that rely on weather and rock dynamics as well as climate change.[104] Neural networks and SVM using a probability bound analysis were used in Canada to quantify and identify uncertainties relating to AMD in order to mitigate risks and improve AMD management strategies.[105] A neural network, SVMs, DT and KNN were used to predict copper concentrations in an AMD plant in Canada to develop cost-effective and sustainable AMD remediation strategies.[106]

1.1.3.3 Stacking Ensemble machine learning

Stacking Ensemble machine learning (SE-ML) involves the combination of predictions from multiple heterogeneous machine learning models on the same dataset.[107] A single model learns how to best combine the predictions from each of the individual models, thereby, using the “wisdom of the group”.[107] In this way one hopes to remove the errors associated with each individual model and obtain a much more accurate representation of the true value.[92]

As depicted in Figure 1.13, SE-ML trains heterogenous (i.e. different) base models with a training set in the first layer (level 0) and then combines their predictions to train a meta-learner (level 1) in the next layer in order to give the final prediction.[108] SE-ML was proven to have outperformed other models in binary classification of water quality parameters[109], combined the merits of individual models and improved its accuracy for total suspended solids (TSS) predictions[110] and had a higher chance of detecting contamination levels in water distribution systems[111].

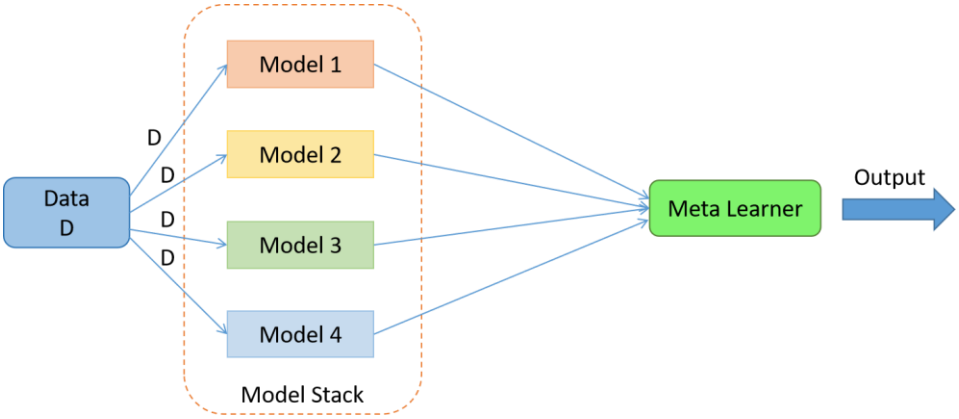


Figure 1.13 : Stacking ensemble machine learning (SE-ML) schematic.[112]

After a thorough literature review, it was found that there were no publications regarding the use of stacking ensemble machine learning regression algorithms for water quality prediction in AMD treatment plants. Also noted, was the lack of versatility in models used to predict AMD water quality parameters since the models mainly used were neural networks, SVM and RF. A novel approach to predicting water quality in AMD would be to use a range of diverse ML regression algorithms and also to combine heterogenous (i.e., different) regression models using a stacking ensemble regressor (SE-ML).

Therefore, it would be beneficial to use a variety of individual and ensemble ML regression algorithms (including SE-ML) in the prediction of sulphate values in order to obtain a significantly more accurate representation. This will allow for more comprehensive water quality datasets to be obtained, thus providing a benchmark for further modelling (e.g., time series forecasting and transfer learning) and basis for experimental work (e.g., remediation and recovery of valuable resources). In this study, one of the valuable minerals that has been identified as potentially recoverable, owing to accurate determination of sulphate in AMD, is elemental sulphur or octathiocane (S_8) that has applications in health care products and chemical production. Experiments can be designed better as a result of complete datasets.

1.1.3.4 Transfer Learning (TL)

Transfer learning (TL) is the transfer of knowledge that has been learnt from a previous task using a pre-trained model in order to improve and enhance learning on a related new task that does not share the same dataset.[113] This is done to take advantage of the accuracy of the pre-trained model since it was trained on a large labelled training set (known as the source domain) and can now be used to perform the same task on a similar dataset that has very little or sparse data (known as the target domain).[114] This is extremely useful in the chemical sciences since often fields are interrelated with similar data, however, the availability of data might be an issue when it comes to generating a large enough dataset in order to train ML models from

scratch.[115] Therefore, transfer learning can be used to complete smaller datasets after being trained on a large and accurate dataset.

Transfer learning allows for the prediction of values in a target domain that has a feature space of different dimensions or a feature space comprised of completely different attributes, thus, creating a completely different distribution.[114] This is incredibly important in real-world cases where data is collected from different domains and, thus, have different distributions.[114] Normally, statistical models have to be re-built due to the differing distributions which is expensive and difficult due to the labour involved in labelling training data in order to create a new model.[114] However, transfer learning is able to create a high-performance model that can achieve the same high level of prediction accuracy in a different domain.[114] This allows for new problems to be solved faster using existing knowledge that is shared amongst multiple domains.[116,117] Transductive machine learning is when the source domain and target domain are different datasets, however, the task to be accomplished in both the source domain and target domain are the same.[116]

In the field of environmental monitoring, TL was successfully used to create a residual neural network to forecast particulate matter 2.5 ($PM_{2.5}$) levels that served as an early warning system for indoor air pollution and its associated health risks in a subway station in South Korea.[118] TL was successfully used to classify a limited number minimally labelled Hyperspectral images in order to identify land cover classes that would have otherwise been impossible using traditional ML techniques.[119] For remote-sensing image classification, TL was used to attain maximum classification performance for applications in land resource management, environmental protection and precision agriculture.[120] TL was also used for automated real-time visual water surface oil sheen monitoring that assists with detecting hydrocarbon pollution in subaquatic environments.[121] In the field of marine environmental monitoring and exploration, an unsupervised transfer learning algorithm was exploited to reuse existing training data for object detection in new marine image datasets.[122]

In the context of this project, transductive transfer learning can be used to predict the sulphate values in two other AMD treatment plants operating using the same active treatment, but in different locations. This is possible due to the influent AMD water and process used to treat the water being similar, hence, the datasets produced are of similar dimensions and feature spaces such that the pre-trained model should be able to accurately predict the sulphate values in the other two plants. After a thorough literature review, it was found that there existed no publications on the use of TL in AMD and so this provided additional novelty to this work.

By harnessing the power of TL and SE-ML, this study seeks to predict sulphate levels in three AMD treatment plants to increase the potential feedstock piles (three treatment plants as

opposed to one) and availability of data needed to model the possible extraction of commercial octathiocane from AMD and treated AMD. This ultimately contributes towards creating a bigger and more efficient circular economy that promotes sustainable development, protects human rights and boosts the South African economy.

1.1.4 Generative AI

As discussed previously, SVMs, DT, MLP-ANN etc. were all examples of discriminative AI models and techniques that are mainly deployed to understand and learn patterns in large datasets and leverage this knowledge learnt in order to make predictions, classifications or cluster observations according to a set of similar features.[123] These models are used to differentiate between data and spot anomalies, assist with visualizations and carry out supervised and unsupervised learning.

Generative AI differs in the sense that its applications are more creative and seeks to understand how data is distributed in order to generate new content rather than learn patterns.[123] Generative AI has been attributed as the main catalyst that initiated the 5th paradigm of science due its ability to expedite repetitive and cumbersome work in order to open up new creative avenues for discovery.[124] Generative AI also has the ability to overcome several bottlenecks currently faced by discriminative AI models which are discussed below.[125]

1. **Data availability bottleneck** – discriminative models have proven to perform inadequately on small datasets, however, generative AI can synthesis new data based on existing relationships and this can be used to enhance and enlarge datasets. Significant bias towards only reporting and publishing data pertaining to experiments that have succeeded and, therefore, concealing data pertaining to failed experiments contributes to a heavy positive class imbalance that affects the performance of discriminative AI models. Generative AI models have the ability to generate data based on experiments that have both worked and failed which provides much more robust dataset to learn form.
2. **Cognition bottleneck** – traditional ways of disseminating scientific knowledge in the form of publications leads to high bias when interpreting the work when extracting knowledge the traditional way, i.e., by human reading. Such bottlenecks lead to the fragmentation of scientific domains into highly specialized fields which further impacts the unification of data as well as hinders the progression of new inventive ideas. Generative AI has the ability to overcome this by creating new, different and more creative data quickly with limited human bias which can be shared easily.
3. **Actionability bottleneck** – due to the nature of the statistics that underpin discriminative models, these models may be unsuited to datasets that contain a great

deal of variety between observations which make them related to each other, but not unified. Generative AI has the ability to generate synthetic data which could greatly enhance the unification of fragmented datasets.

In summary, while discriminative AI seeks to understand data, generative AI seeks to augment and generate new data.

1.1.4.1 Generative text models – the case of ChatGPT

Generative AI tools such as generative text models which can create poems, essays, code, emails etc. have significantly gained popularity after the launch of the Large Language Model (LLM) "ChatGPT"[126] in November of 2022[127]. ChatGPT is a LLM chatbot trained by OpenAI that interacts in a conversational way to human input prompts in order to generate a text output.[126]

Since its launch, ChatGPT has greatly contributed to enhanced research efficiency and allows for a more tailored and curated learning experience for students.[128] It has gained much criticism due to its potential to be misused as well as the lack of accuracy associated with naïvely trusting everything it says.[129–131] It also has the potential to generate false information (i.e., it "hallucinates"), fabricate references, fails to recall factual information, struggles to perform chemistry tasks and makes mathematical errors.[132,133] However, its popularity and usage has grown exponentially since its release in November 2022[127] and it is now a matter of "how" it should be integrated into chemistry curricula rather than "if" it should be. Despite the risks and potential harm ChatGPT might cause, its potential to be used for beneficial teaching and learning is big since students and staff can now experience an efficient and tailored learning experience that shifts the focus from simple recall to more application based learning and creative thinking.

With the increase in popularity and accessibility of ChatGPT, significant attention has been drawn to its coding and debugging capabilities.[134,135] ChatGPT has the ability to write code segments in a variety of languages, with Python being of particular interest, and also provides guidance along with great technical support.[136] There exists a huge potential for this to be exploited in non-technical fields (such as humanities) and non-traditional fields (such as chemistry) in order to get more customized and tailored software that is suitable for use by those with no programming background.[137] Chemistry curricula traditionally does not teach students any programming skills[138], however, models such as ChatGPT could be used to relieve some of the hardcore programming demands, and as such, allow students to create customized software that is freely available and easily accessible.

Artificial Intelligence (AI) and its subset, Machine Learning (ML), have become very popular due to their increased efficiency, time-saving nature, cost-effectiveness and accuracy.[139] However, there exists a large technical gap in relation to the knowledge and programming skills required to implement ML models in fields such as chemistry.[139,140] Therefore, a large disparity exists between the development of accurately trained models and the deployment of these models for general public use, i.e., for use by people with little to no coding experience and knowledge of ML. The incredibly useful and ready-to-use ML models have remained reserved and largely unused due to their lack of accessibility. This is especially true in chemistry and related environmental fields since knowledge of how to access, open, use and collect the output from these models is normally only known by those that have created and trained ML models in the first place.

Therefore, it is beneficial in order to create a no/low code environment where people with no prior programming background nor ML knowledge can use the trained models. This would greatly improve the accessibility and user-friendliness of these ML models which allows for them to be used more frequently, thus, contributing to the enhancement of water quality datasets since, for instance, in this study, more sulphate values for different AMD treatment plants can be predicted using a single trained model by people such as laboratory analysts. Such a low/no code environment is termed a Graphical User Interface (GUI) and resembles an application ("app") like environment.

In the field of environmental sciences, GUIs have been used in order to increase the findability and accessibility of published marine related datasets by creating a custom search engine called "PangaVista"[141] that operated using the "PANGAEA" GUI which allows one to search for datasets related to a specific keyword or even using a specific set of geographic co-ordinates.[142] In another study, a GUI called ENFORMS (Environmental Information System) was developed for a regional watershed in Michigan, USA in order to query and manipulate multimedia (satellite images, documents, audio-files, charts, etc) and access its associated metadata (descriptions, data type, hierarchal classification).[143]

1.1.4.2 A Graphical User Interface (GUI) for discriminative AI models made using generative AI models such as ChatGPT

Programming knowledge needed to create GUI's is quite in-depth and requires a decent amount of experience. However, models such as ChatGPT when used in conjunction with human prompting can be used to create the scaffolding of the code needed to generate the GUI.[144] The code can then be modified to one's liking using ChatGPT which conducts debugging required to create a fully functioning and easily accessible GUI.[145]

This allows for the trained ML models to become more generalisable and expands the transfer learning capabilities of the ML models since they can be used to predict sulphate levels in more AMD treatment plants and mine-impacted water bodies by people with no coding nor ML knowledge. This would further contribute to acquiring more sulphate level data for design of experiments that are aimed at water remediation and recovery of value from AMD.

1.2 AIM AND OBJECTIVES

1.2.1 Aim

The aim of this project was to consolidate AMD treatment plant datasets by imputing missing sulphate concentrations and applying transfer machine learning on the sparse datasets.

This aim was achieved by pursuing the following objectives:

1.2.2 Objectives

1. To clean and pre-process the large dataset obtained from the Central Rand AMD treatment plant using traditional data cleaning processes and geochemical modelling.
2. To predict sulphate levels in the Central Rand AMD treatment plant using a variety of individual and ensemble machine learning regression models.
3. To determine the best performing model and to determine the effect of combining models *via* stacking ensemble machine learning.
4. To predict and impute the missing sulphate levels in two other similar AMD treatment plants (East Rand and West Rand) that have sparse datasets using transfer machine learning.
5. To create a GUI for the best performing model in order to improve its accessibility and generalizability for sulphate level prediction by those with little to no programming nor ML knowledge.

1.3 HYPOTHESIS AND QUESTIONS

1.3.1 Hypothesis

Machine learning regression models can be used to determine sulphate levels in the Central Rand AMD treatment plant. Ensemble machine learning model predictions are more accurate than individual models. Transfer machine learning can be used to predict the sulphate values in two other AMD treatment plants (East Rand and West Rand) with scanty datasets.

1.3.2 Questions

1. Does geochemical modelling pre-cleaning significantly alter the sulphate values?
2. What are the best machine learning regression models to use for predicting sulphate values and do ensemble models have better predictive accuracies?
3. Can transfer learning be used to determine the sulphate values in two other AMD treatment plants that have scanty datasets?

CHAPTER 2 : METHODOLOGY

This chapter presents the research approach and methodology protocols to pursue the objectives of the study. Aspects such as the acquisition of data, data preprocessing and cleaning, optimisation and application of the ML models are discussed.

2.1 LOCATION AND ACQUISITION OF DATA

Historical data pertaining to water quality parameters of analysed AMD water samples (untreated and treated) from three AMD treatment plants, namely, Central Rand, East Rand and West Rand were collected individually by the laboratory technicians at the respective plants over the years of 2015 - 2020. These datasets were then collated onto separate Excel spreadsheets and sent from the Trans Caledon Tunnel Authority (TCTA) to the University of the Witwatersrand, School of Chemistry and were used as the original master datasets. This was the only data employed for this analysis and no new data was generated. The historical data covered continuous monitoring over the years 2015–2020 with daily analysis conducted on each sample from each location. A modified map[146] indicating the location and proximity of the AMD treatment plants to each other in Johannesburg, Gauteng, South Africa can be found in Figure 2.1.

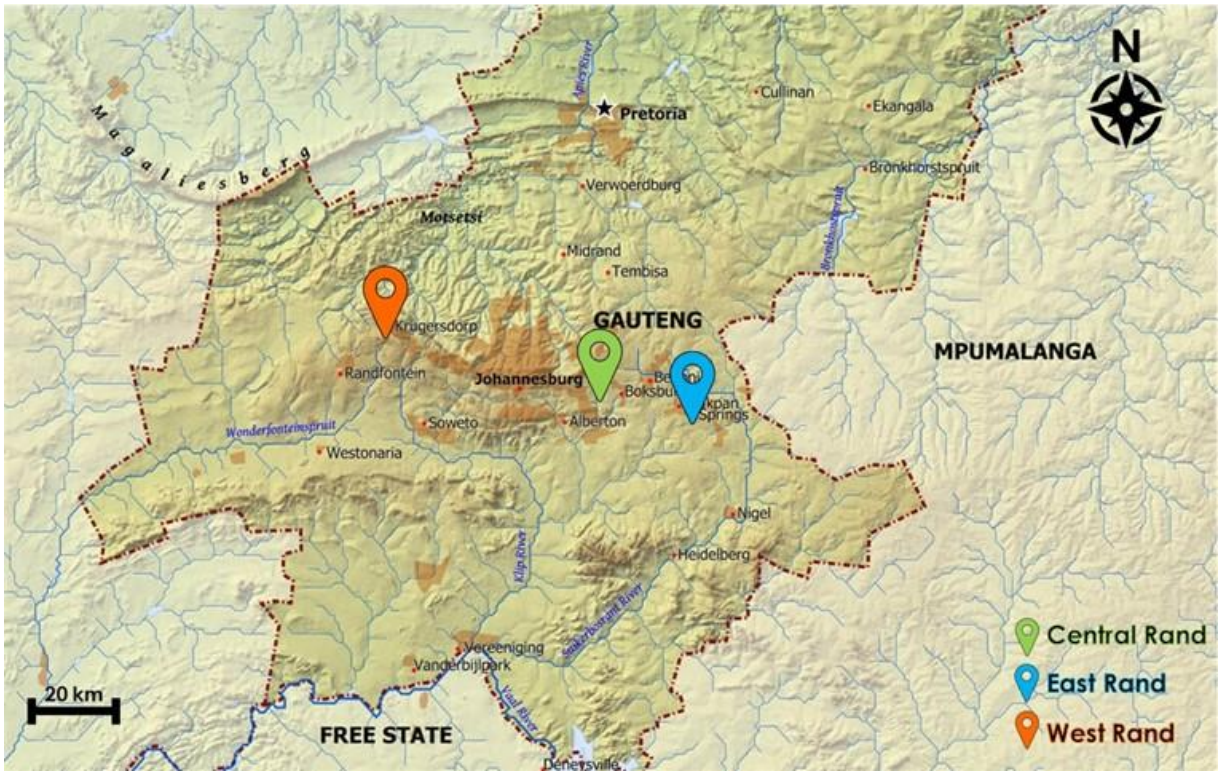


Figure 2.1 : Modified map[146] of Johannesburg, Gauteng, South Africa, indicating the location of the AMD treatment plants whose data was analysed and their proximity to each other.

2.2 METHOD

Geochemical modelling, data cleaning, model training and model comparison were conducted on the Central Rand dataset. The application of transfer learning was then conducted using the best model obtained from the Central Rand dataset to the West Rand and East Rand datasets. The software used was Python 3.11 programming language hosted in the Google Colaboratory environment using a Lenovo V110 laptop equipped with Intel® CoreTM i3-6006U CPU @ 2.00GHz.

The specific Python packages (with their version numbers in brackets) that were used are detailed as follows: Numpy (1.23.5) and Pandas (1.5.3) for data cleaning, processing, conversion and exploratory data analysis. Scikit-learn (1.2.2) for model training and testing. All graphing and visualizations were conducted in Matplotlib (3.7.1), Seaborn (0.12.2) and Plotly. (5.15.0).

The methodology pertaining to each dataset will be discussed individually in the following sections.

2.2.1 Central Rand dataset

As seen in Figure 2.2, the initial large dataset received from the Central Rand plant contained two major categories, namely, “AMD FEED” and “TREATED WATER” where the “AMD FEED” was further split into “PUMP A” and “PUMP B” signifying two input streams.

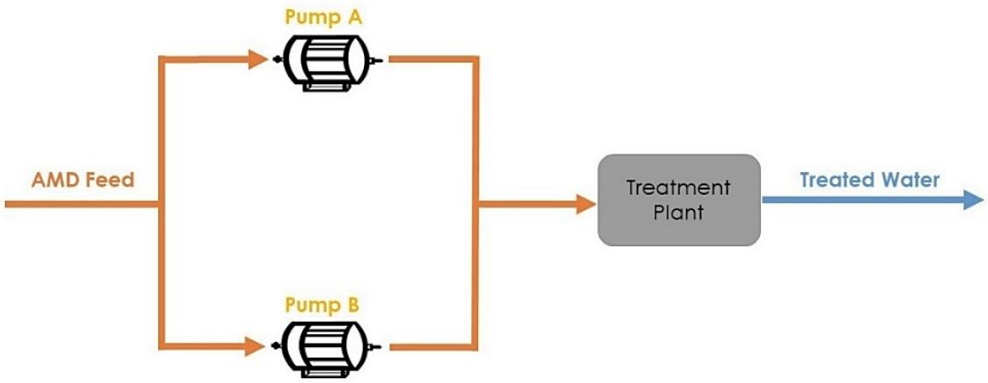


Figure 2.2 : Schematic overview of the Central Rand AMD treatment plant.

Each of the three categories (“PUMP A”, “PUMP B” and “TREATED WATER”) consisted of 10 water quality parameters that were monitored over the years of 2015 to 2020 and a screenshot of the initial data received can be found in Table 2.1. These water quality parameters were Temperature (°C), Electrical Conductivity (mS/m), Total Dissolved Solids (mg/L), pH, Turbidity (NTU), Total Suspended Solids (TSS) (mg/L), Sulphates (mg/L), Aluminium (mg/L), Iron (mg/L) and Manganese (mg/L).

Table 2.1: Screenshot of initial layout of received Central Rand dataset.

	AMD FEED																		
	PUMP A										PUMP B								
	Temperature	Electrical Conductivity	Total Dissolved Solids	pH	Turbidity	TSS	Sulphates	Aluminium	Iron	Manganese	Temperature	Electrical Conductivity	Total Dissolved Solids	pH	Turbidity	TSS	Sulphates	Aluminium	Iron
	°C	mS/m	mg/l	pH	NTU	mg/l	mg/l	mg/l	mg/l	mg/l	°C	mS/m	mg/l	pH	NTU	mg/l	mg/l	mg/l	mg/l
2015/03/01																			
2015/03/02	24.9	567	3160	5.56	1.13		3490		772	anganese rea	24.8	564	3100	5.41	1.61		3772		815
2015/03/03	25.5	551	3040	5.75	2.43		3755		1049	22	25.6	553	3070	5.54	2.34		3222		1220
2015/03/04	25.4	548	3.07	5.79	6.1		3706		888	29	25.2	567	3150	5.65	10.9		3879		844
2015/03/05	o distilled water		no distilled wat	distilled wat	distilled water		no distilled wat	distilled wat	distilled wat	distilled wat	distilled watre		no distilled wat	distilled wat	distilled watre		no distilled wat	distilled wat	distilled wat
2015/03/06	26.3	538	2940	5.81	2.81		4177		899	20	26.1	527	2930	5.71	7.45		4053		820
2015/03/07																			
2015/03/08																			
2015/03/09	26.1	535	2900	5.4	5.39		4039		845	21	25.8	538	2980	5.55	6.87		3891		864
2015/03/10	25.6	555	2990	5.75	11.5		3999		712	23	25.5	527	2950	5.87	5.53		3817		737
2015/03/11	27	553	3070	5.79	3.72		3775		811	18	26.7	568	3100	5.61	4.94		3993		1100

2.2.1.1 Data Preparation and Pre-Processing

As observed in Table 2.1, initial inspection of the dataset revealed that the Aluminium and TSS columns were incomplete and consisted of mainly blank values, thus, these columns were dropped from all three categories (i.e. “PUMP A”, “PUMP B” and “TREATED WATER”). The date column provided no significant value for machine learning and was, thus, also dropped from the entire dataset. The Turbidity column was related to how much light was able to penetrate through a sample, the higher the turbidity the higher the amount of suspended solids. Whilst this was an important parameter in determining water quality, it was not directly related to the chemistry between the dissolved cations and anions and was, therefore, unrelated to most of the other water quality parameters and subsequently dropped from the entire dataset.

However, the date column provided valuable information for human understanding and upon further inspection, several peculiarities were noticed specifically regarding the data for the Treated Water (TW). It was found that data for this category was recorded on weekends, public holidays and on days where the plant was assumed to be shut down or off, and this immediately brought the legitimacy and credibility of the TW data into contention.

Since AMD Feed water has a totally different chemical composition to its Treated Water, there was a need to separate the large dataset into three smaller more similar datasets in order to ensure that the models trained later on would be trained on relevant data. The retention time (i.e., how long the AMD feed took to reticulate through the plant) was also considered when splitting the large dataset since it was likely that this retention time was extremely long. Thus, the TW had no relation to the AMD water measured one the same day since they were completely different samples.

Thus, the large Central Rand dataset was then split into three individual smaller datasets according to the categories, namely, “Pump A”, “Pump B” and “Treated Water”. Each smaller

dataset was copied and re-created in a separate excel file and dealt with individually for pre-cleaning, modelling, cleaning, training and testing.

2.2.1.2 Data Pre-Cleaning

The datasets for “Pump A”, “Pump B” and “Treated Water” were first pre-cleaned separately in Excel by shortening each of the remaining headings to “TEMP”, “EC”, “TDS”, “pH”, “SUL”, “FE” and “MN” with the prefix “A-“, “B-“ or “T-“ indicating each of the three categories. The units row was also removed for each dataset. Any values that had a “±5%” associated with them were fixed by removing the “±5%” using Excel’s built in “Find and Replace” feature.

For each of the three datasets, data pre-cleaning was then conducted in Python in order to remove any blank rows (usually indicative of the weekends, maintenance or breakdowns), to convert any character data (indicating any specific issues, missing reagents or technical breakdowns) to NaN (Not-A-Number) values and to fix numerical typos (e.g. ‘6..19’ instead of ‘6.19’). For all parameters, all rows containing a NaN value were then removed from the dataset and all remaining numerical data was converted to a float. The code used to conduct this cleaning for Pump A can be found in Appendix A. Pump B and TW files are not appended due to them bearing a very high similarity to Pump A . The cleaned datasets were then saved as new separate excel files and were ready for geochemical modelling.

2.2.1.3 Data Pre-Cleaning using Geochemical Modelling

Upon inspection of the received sulphate concentrations, it was found that these values were extremely high, therefore, these values needed to be thermodynamically corrected for using geochemical modelling. The software used to conduct this was PHREEQC Interactive 3.7.3-15968[147] hosted on a Lenovo V110 laptop equipped with Intel® CoreTM i3-6006U CPU @ 2.00GHz. Using PHREEQC each of the smaller datasets (“Pump A”, “Pump B” and “Treated Water”) were modelled in order to thermodynamically correct for the extremely high sulphate values by performing a charge balance on the sulphate values using the TEMP, pH, FE and MN values. As seen in **Equation 2.1** and as described in PHREEQC, a charge balance adjusts the concentration of a selected element (with an ionic species, in the case of sulphur it is S_6) in relation to other elements in order to achieve a balance.[148]

$$\sum(C_i z_i) = \sum(A_j z_j)$$

Equation 2.1

Where C_i is the concentration of cations, z_i is the respective cationic charge, A_i is the concentration of anions and z_j is the respective anionic charge. In the case where there is a great charge imbalance, the problem is unsolvable if no adjustments can be made by removing

all of the element.[148] In this case, the specific samples that were deemed to be unsolvable were completely removed from the datasets.

Several outliers or incorrect numerical value were identified during geochemical modelling and this served as an extra data pre-cleaning step. Examples of outliers removed were those where the total anionic species concentration did not balance with the cationic species concentration or where the pH recorded would be unreasonable when looking at the measured concentrations of ionic species present.

Once the corrected sulphate values were obtained, they were used to replace the old sulphate values in order to create a final complete Excel file for each dataset (Pump A, Pump B and Treated Water) and were ready for regression. A screenshot of Pump A's cleaned and modelled Excel file ready to be fed into Python for regression can be seen in Table 2.2. Pump A's PHREEQC input and output code files can be found in Appendix B (Pump B and TW files are not appended due to them bearing a very high similarity to Pump A).

Table 2.2: Screenshot of Central Rand Pump A's Excel file that contained the cleaned and modelled values ready to be fed to Python for regression.

A - TEMP	A-EC	A-TDS	A-pH	A-SUL	A-FE	A-MN
25.5	551	3040	5.75	1.29E-02	1049.4	21.6
25.4	548	3.07	5.79	1.12E-02	888.07	28.5
26.3	538	2940	5.81	1.12E-02	898.82	20.1

2.2.1.4 Data Cleaning and Visualization

Each of the cleaned and modelled datasets were then fed to Python for Exploratory Data Analysis (EDA) which included visualizing the distribution of each dataset using histograms, density plots, box and whisker diagrams, scatter matrices and correlation matrices in order to better understand the distribution and correlation of each dataset. These visualizations and their analysis can be found in "CHAPTER 3: RESULTS AND DISCUSSION". From EDA, it was found that several outliers existed in each dataset and this heavily skewed the distributions of the respective datasets. These outliers were removed by creating a custom function that utilized the Inter Quartile Range (IQR) method[149] to remove the entire row in which the outlier was found.

The Inter Quartile Range method was used to determine which data points constituted the main dataset and which datapoints were outliers since anything above 1.5 times the upper quartile and 1.5 times the lower quartile fell outside the main dataset and became an outlier. It was necessary to remove outliers since they heavily skewed the distribution of the data and, therefore, contaminated the statistical representation of the datasets which could have led to poor model performance when conducting machine learning.

The new, smaller datasets were then normalized using a “MinMaxScaler” that removed the effect of different scales amongst variables (e.g. TEMP being in units of °C and FE being in mg/L). Normalization is especially necessary when dot products are used between predictors (such as KNNR and SVR), when penalties are applied or when distances are being measured in order to prevent dwarfing of smaller predictors by other predictors that have larger scales of measurement.[150,151] This “MinMaxScaler” transformed all the data to lie in the range of 0 to 1 by implementing **Equation 2.2**[152] to each datapoint.

$$y = \frac{x_i - min}{(max - min)}$$

Equation 2.2

2.2.1.5 Feature extraction and Dimensionality reduction

Feature extraction using Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE) was conducted to reduce the dimensionality (i.e., reduce the number of predictors) in the data. PCA essentially breaks down a large matrix *X* into the product of two smaller matrices *T* (Target) and *P’* (Predictors) that capture the essential features of *X*.[153] By plotting the columns of *T* against the rows of *P’* one is able to gain an understanding of the dominant “object patterns” and its resulting complimentary “variable patterns” and this can be used to separate out noise from the data.[153] RFE uses statistical measures in order to rank features based on their importance where features are labelled as 1 in descending order of importance with less important features being removed after each iteration.[154] This prevents overfitting when predictors are highly correlated.[154]

PCA was conducted using two principal components and its results were visualized using a bi-plot in order to determine the inter-relationships between variables by observing clustering of individual observations (scores) and how the water quality parameters were related to each other (i.e., loadings). A Scree plot was also used to determine the number of principal components that were necessary to explain at least 75% of the variance in the data. The number of optimal clusters was determined using the “Elbow Method” which deems the optimal number of clusters to be the point at which a plot of “Number of Clusters” against the “Within-Cluster Sum of Squares” bends and a “K-Means” clustering model was used to determine the actual clusters. The “Within-Cluster Sum of Squares“ (WCCS) can be mathematically defined according to **Equation 2.3**[155],

$$WCCS = \sum_{k=1}^{k} W(C_k) = \sum_{k=1}^{k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

Equation 2.3

Where for k clusters, x_i is the datapoint belonging to cluster C_k and μ_k is the mean value of points that lie in cluster C_k.

RFE was conducted using LR as the fitting model and the "number of features selected" was the number of principal components needed to explain at least 75% of the variance (which was found using the Scree plot). The highest ranking features, as determined from RFE, were then deemed to be the most important features and were selected to be the predictors and combined with the sulphate values (target) in order to create the final datasets ready for ML.

The Python code used for data cleaning and visualization for Pump A can be found in "Appendix C" (Pump B and TW files are not appended due to them bearing a very high similarity to Pump A). A summary schematic outlining the entire data processing procedure for the Central Rand treatment plant can be found in Figure 2.3.

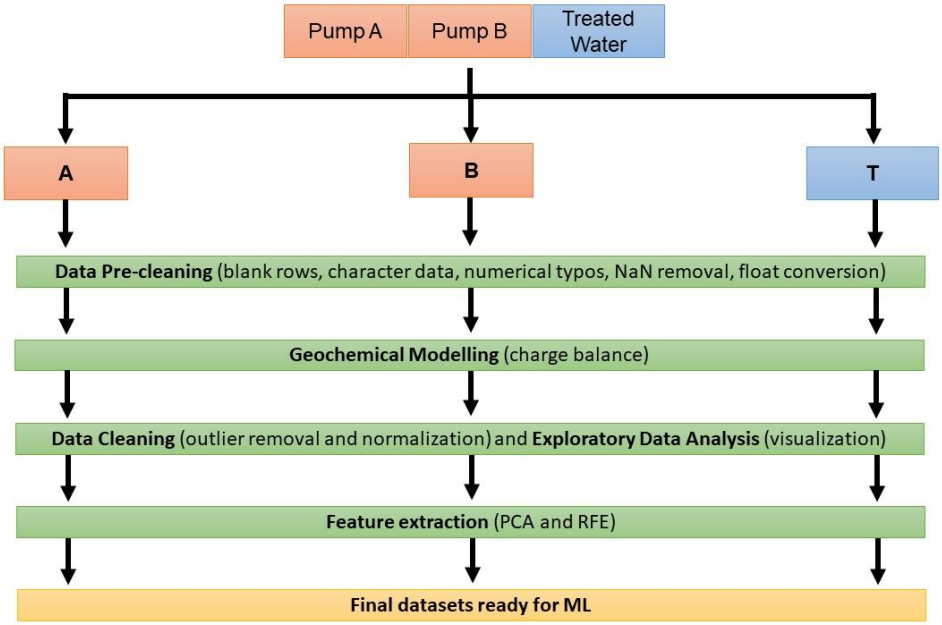


Figure 2.3 : Data processing procedure conducted for the Central Rand AMD treatment plant.

2.2.1.6 Machine Learning – Individual Regression Models

The final dataset was split into an 80% training set and 20% testing set with 10-fold cross validation being used as a resampling method. Ten individual regression models were spot checked using the Negative Mean Squared Error (NMSE) metric in order to compare models. NMSE was selected to be the metric of comparison since it provides the best manner in assessing the difference between the true and predicted sulphate values in an understandable manner since the best performing model is the one with the NMSE closest to zero. NMSE was chosen as opposed to the Mean Squared Error (MSE) since NMSE was easier to understand graphically, since the "highest" (i.e., closest to the top) model was the best performing model.

The ten models chosen were Linear Regression (LR), Ridge Regression (RD), LASSO (LASSO), Elastic Net (EN), K-Nearest Neighbours Regressor (KNNR), Decision Tree (DT), Support Vector Regression (SVR), Extreme Gradient Boosting Regressor (XG), Random Forest Regressor (RF) and Multi-Layer Perceptron (MLP).

2.2.1.7 Stacking Ensemble Machine Learning (SE-ML)

SE-ML was conducted in two forms. Firstly, SE-ML was conducted using a function inspired by Jason Brownlee[107] to combine all the individual regression models in level0 and using a Linear Regressor as the level1 model in order to predict sulphate levels in the training set.

Thereafter, stacking ensemble learning was repeated using only the best regression models (NMSE closest to zero) and still using Linear Regression as the level1 model to predict sulphate values in the training set. This was done to determine the effect of removing the worst performing models on the prediction of sulphate levels in the training set.

The experimental settings used for all models mentioned (individual and stacked) were the default settings of the models when imported from Python's Scikit-learn library. No modifications to these default setting were made. For the Random Forest (RF) regressor, the number of estimators used was 100 in all instances. This was done on purpose in order to fairly compare the predictive performance of all the models. Optimisation of the models was deemed to be unnecessary since the default settings provided accurate predictions.

The Python code used for machine learning regarding the individual regression models and SE-ML for Pump A can be found in "Appendix D" (Pump B and TW files are not appended due to them bearing a very high similarity to Pump A).

2.2.1.8 Model Testing

All models (individual, SE-ML using all models and SE-ML using only the best models) were then used to predict sulphate values on the testing set and their performance was compared using the Mean Squared Error (MSE), Mean Absolute Error (MAE) and the Coefficient of Determination (R^2).

The MSE measures the average of the squared differences between the predicted value and the true value via **Equation 2.4**, where y_i is the target value and $\hat{y}_i$ is the predicted value.[156]

$$MSE = \frac{1}{N}\sum_{i}^{N}(y_i - \hat{y}_i)^2$$

Equation 2.4

By squaring the difference this allows for the removal of the effects of a positive or negative residual, thereby, allowing for an accurate error to be calculated.[156] Large errors are ultimately

further emphasized and this has a “punishing” effect on model training since the lowest MSE (i.e., closest to zero) is the most favoured.

The MAE does not weight any errors, therefore, no inflation of errors occurs and so the metric increases linearly with an increase in error.[156] MAE is calculated by taking the average of the absolute error values as shown in **Equation 2.5**.[156] The best performing model is the one with MAE closest to zero since this is when the loss is minimized.

$$MAE = \frac{1}{N}\sum_{i}^{N}|y_i - \hat{y}_i|$$

Equation 2.5

The R^2 evaluates the performance of the model rather than the loss and is independent of the context of the problem.[157] In a nutshell, the R^2 metric calculates how much better the regression model is at predicting a value rather than just using the mean as a prediction of the unknown value.[157] It is calculated using the Squared Sum error of the Regression line (SSR) and the Squared Sum error of the Mean line (SSM) as shown in **Equation 2.6**.[157]

$$R^2 = 1 - \frac{SSR}{SSM}$$

Equation 2.6

When the R^2 value is closer to zero, the fraction tends to 1 and it means the SSR is almost equal to the SSM (since 1-1=0), and this is an indication of poor model performance since the regression model doesn’t perform any better than the mean.[157] When R^2 is closer to 1, the fraction tends to zero, and this means the regression model is far outperforming the mean value.[157] Therefore, models with R^2 values closer to 1 are the best performing models.

The Python code used for model testing for Pump A can be found in “Appendix E” (Pump B and TW files are not appended due to them bearing a very high similarity to Pump A). A summary schematic outlining the ML procedure conducted on the Central Rand dataset can be found in Figure 2.4.

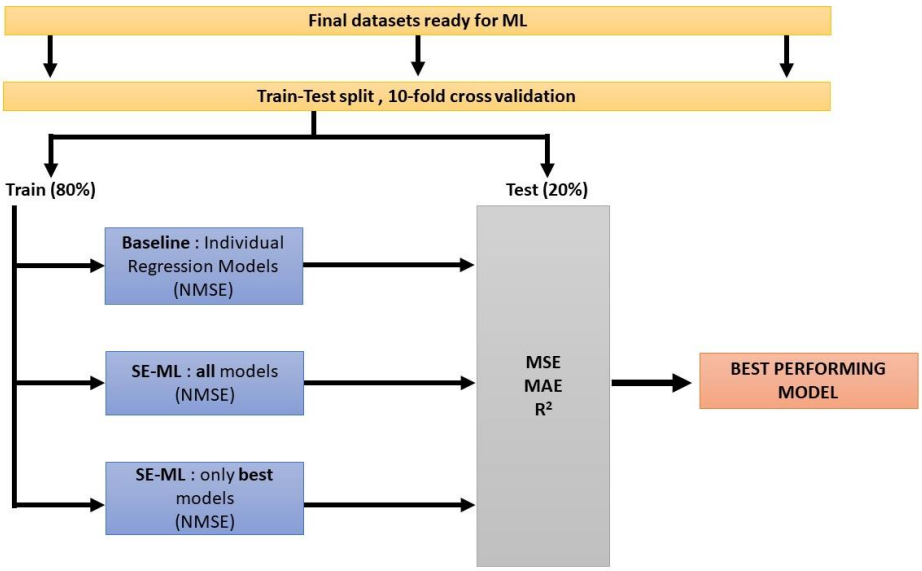


Figure 2.4 : ML procedure used to train and test regression models on the Central Rand dataset.

2.2.2 Transfer Learning (TL) onto East Rand (ER) dataset

The East Rand dataset bore a high similarity to the Central Rand dataset as seen in Table 2.3, with a single AMD feed and Treated Water subsets. The same data splitting, pre-cleaning, geochemical modelling, data cleaning and visualization process was followed as outlined in "2.2.1 Central Rand dataset" in order to prepare the data for TL. Additionally, there were two columns present in the East Rand dataset, viz, "AMD Pumped Volumes, m^3" and "Sludge disposed, m^3" which were also dropped for the Treated Water data.

Table 2.3: Screenshot of initial structure and layout of the East Rand AMD treatment plant.

Date	AMD Feed (composite)										TREATED WATER					
	Electrical Conductivity	Total Dissolved Solids	pH	Turbidity	TSS	Acidity	Sulphates	Aluminium	Iron	Manganese	Temperature	Electrical Conductivity	pH	Turbidity	Total Dissolved Solids	Total Suspended Solids
	mS/m	mg/l	pH	NTU	mg/l		mg/l	mg/l	mg/l	mg/l	°C	mS/m	pH	NTU	mg/l	mg/l
2016/08/31	310	2700	6.31	60.90	92	346	1680	0.016	107	7	22.2	266	8.6	12.9	2324	9.3
2016/09/01	312	2734	6.33	60.80	70	350	1617	0.098	109	8	20.3	265	8.62	12	2310	8.6
2016/09/02	314	2756	6.36	60.70	92	332	1693	0.034	110	8	21.1	264	8.67	10.7	2389	8.6
2016/09/03	315	2818	6.35	68.10	70	348	1582	0.021	102	9	21.3	265	8.54	2.93	2372	4
2016/09/04	312	2828	6.18	51.40	88	Not analysed	1502	Not analysed	114	Not analysed	21.9	264	8.71	13.1	2270	6.2
2016/09/05	305	2864	6.15	71.60	90	Not analysed	1709	Not analysed	117	Not analysed	22.6	260	8.6	11.1	2444	7.1
2016/09/06	308	2815	6.3	60.40	Not Analysed	338	1675	0.015	104	8	21.9	264	8.5	12.1	2423	Not analysed
2016/09/07	Not Analysed	Not Analysed	Not Analysed	Not Analysed	Not Analysed	Not Analysed	Not Analysed	Not Analysed	Not Analysed	Not Analysed	21.3	265	8.52	12.5	2338	6.5

However, during geochemical modelling of the East Rand Treated Water data, the simulation runs were halted due to many faulty data points and could not proceed to finish. This was experimental evidence for the inherent inaccuracies or highly unusual chemistry present in the East Rand Treated Water dataset and, therefore, no TL was conducted on the Treated Water.

Once the East Rand AMD feed data was cleaned, TL was conducted using the best performing model from the Central Rand dataset (Pump B stacking regressor using only the best models) and its predicted values were assessed for its accuracy by comparison to the true sulphate values using MSE, MAE and R^2 metrics. The predictors used were “ER AMD-TEMP” and “ER AMD-FE” and the target was “ER AMD-SUL”. These predictors were fed as a new testing dataset and the predicted sulphate levels were compared to the true sulphate levels to determine the accuracy of the stacking regressor.

2.2.3 Transfer Learning (TL) onto West Rand (WR) dataset

2.2.3.1 Initial inspection and structure of West Rand dataset

Five Excel workbooks each representing a year between 2015 and 2020 were received from the West Rand AMD treatment plant. As seen in Table 2.4, each workbook had separate sheets for each month of the year starting from 24th November of the previous year such that, for example, one working year runs from 24th November 2014 to 23rd January 2015. The measured values for each sampling point on each day were given along with an acceptable reference guideline for each water quality parameter.

Also seen in Table 2.4, each of the water quality parameters measured (e.g., pH) were measured for samples taken at several locations. However, further inspection revealed that not all water quality parameters were measured for all locations and this led to an incomplete dataset for several sampling points. This was different to the Central Rand dataset which measured all water quality parameters on three sampling points consistently over the five-year period.

Table 2.4: Screenshot of a portion (24 November 2014 – 23 December 2014) of the layout of the West Rand AMD treatment plant dataset.

24 November 2014 - 23 December 2014

Dec-14	Location	Ref	Units	Directive Guidelines	24	25	26	27	28	29	30	1	2
pH	CPS Pit Outlet Treated water	Q10a	pH units @25°C	6.5 - 9.5	8.79	8.59	8.56	8.51	8.55		8.79	8.71	8.73
	17Winze + Overflow from 18 Winze (Q0)	Q24 &Q25a			3.40	3.58	3.39	3.36	3.72		3.66	4.61	3.35
	Q1 (Pumped from Sump1)	Q15			3.45	3.59	3.73	3.87	4.46		3.98	4.74	4.08
	Q2 (Pumped from Sump2)	Q16			3.10	2.94	2.95	3.06	2.99		2.94	2.89	2.88
	Untreated Flow into KGR	Q17			2.86	2.85	2.85	2.93	2.88		2.81	2.85	2.82
	Treated Flow into KGR	Q10b			7.26	7.20	7.36	7.32	7.36		7.39	7.39	7.38
	KGR Inflow - Q5	Q14			3.86	4.04	5.87	3.63	3.47		3.97	3.50	3.58
	Hippo Pool	Q22							3.87				
	Aviary	Q23							6.42				
	18 Winze	Q25b			6.28	6.27	6.30	6.34	6.27		6.30	6.30	6.33

Dec-14	Location	Ref	Units	Directive Guidelines	24	25	26	27	28	29	30	1	2
Temperature	CPS Pit Outlet Treated water	Q10a	Temperature °C		20.50	20.60	21.00	19.50	19.80		19.20	20.60	21.10
	17Winze + Overflow from 18 Winze (Q0)	Q24 &Q25a			20.10	19.60	20.00	18.40	19.40		19.30	19.50	20.00
	Q1 (Pumped from Sump1)	Q15			19.80	19.10	19.30	17.80	19.50		19.50	19.30	20.10
	Q2 (Pumped from Sump2)	Q16			19.30	18.80	19.40	17.60	19.20		19.10	18.80	19.60
	Untreated Flow into KGR	Q17			19.40	18.80	19.70	17.70	19.80		19.60	19.10	20.10
	Treated Flow into KGR	Q10b			19.70	19.60	19.80	18.80	19.20		19.20	19.70	20.10
	KGR Inflow - Q5	Q14			19.60	19.20	19.70	18.00	19.00		19.00	19.90	20.10

Dec 2014 | Jan 2015 | Feb 2015 | Mar 2015 | April 2015 | May 2015 | June 2015 | July 201: ...

Based on the pH of the water samples taken at different locations, each of the locations that had a significant amount of data were then split into one of two categories *viz*. “Untreated AMD” and “Treated AMD” as shown in Table 2.5.

Table 2.5: Split of West Rand dataset based on pH (low pHs are acidic and, therefore, untreated and vice versa)

Untreated AMD (low pH)	Treated AMD (high pH)
17Winze + Overflow from 18 Winze (Q0)	CPS Pit Outlet Treated water
Q1 (Pumped from Sump1)	Treated Flow into KGR
Q2 (Pumped from Sump2)	18 Winze
Untreated Flow into KGR	
KGR Inflow - Q5	

Water quality parameters measured such as temperature (°C), electrical conductivity (mS/cm), pH, turbidity (NTU), iron (mg/L) and manganese (mg/L) matched those measured in the Central Rand treatment plant, however, even though sulphates (mg/L) were listed as a water quality parameter no actual values for any of the years were reported. Additional water quality parameters measured that did not feature in the Central Rand dataset were acidity (mg/L), alkalinity (mg/L) and volumes (ML) discharged.

Noticeably, the West Rand treatment plant did not measure nor report any Total Dissolved Solids (TDS) which was measured in the Central Rand treatment plant. The TDS was determined to be an important predicting feature for two out of three of the final models trained on the Central Rand dataset. Therefore, the final model used to conduct transfer leaning was the best model obtained from Pump B of the Central Rand dataset that only needed temperature and iron as predictors in order to determine the sulphate levels. All other models trained (Pump A and Treated Water for Central Rand) all relied on TDS which, unfortunately, not measured on the West Rand data.

Since only the temperature (°C) and iron (mg/L) values were needed as predictors, an initial visual inspection of the data revealed that only the “KGR Inflow - Q5” and “Treated Flow into KGR” locations had consistently measured these two parameters over the five years. Thus, these were the only two locations that were chosen to conduct TL on as the AMD feed and treated AMD, respectively, due to their completeness. A visual schematic of the locations can be found in Figure 2.5.

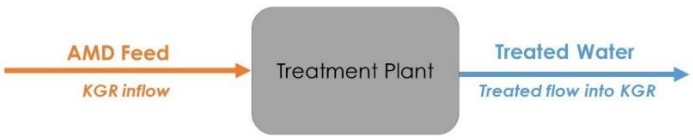


Figure 2.5 : Locations chosen to conduct TL on the West Rand plant.

2.2.3.2 “KGR inflow” (untreated AMD) of West Rand dataset

2.2.3.2.1 Data pre-cleaning

The temperature (°C) and iron (mg/L) values for KGR inflow were manually collected from the master spreadsheets and placed onto a new Excel spreadsheet as seen in Table 2.6, and this served as the main spreadsheet to be used later. Whilst curating the data, it was noticed that in many cases the last decimal place recorded by the lab technicians was always zero and this immediately brought the accuracy and reliability of the data into question.

Table 2.6: Screenshot of KGR inflow (untreated AMD from West Rand) curated spreadsheet.

TEMP - KGR inflow	FE - KGR inflow	SUL - KGR inflow
19.60	1.60	
19.20	0.80	
19.70	0.20	
18.00	4.00	
19.00	9.00	

The curated Excel spreadsheet was then fed to Python for data cleaning in order to remove any character data signifying no sample or no electricity and also to remove any NaN values. The code for this can be found in “Appendix F” and the final pre-cleaned dataframe was saved as a new Excel file.

2.2.3.2.2 Data cleaning and EDA

The pre-cleaned Excel file was then fed to Python for EDA which consisted of visualizing the temperature (TEMP) and iron (FE) distributions and understanding it descriptive statistics. This consisted of plotting its histograms, density plots and box and whisker diagrams.

Afterwards the data was further cleaned to remove any outliers as discovered in EDA and to normalize the values to remove the effects of differing scales (i.e., TEMP in °C and FE in mg/L). This was done using the same custom function from the Central Rand dataset and the “MinMaxScaler” was used as the normalizing function. The code for this cleaning and visualization can be found in “Appendix G”.

2.2.3.2.3 Transfer Learning (TL)

The best performing model from Central Rand Pump B was the Stacked Ensemble Regressor that utilized only the best performing models and a schematic of this model can be found in Figure 2.6. This model was then used to conduct TL on the KGR inflow data in order to predict its sulphate values.

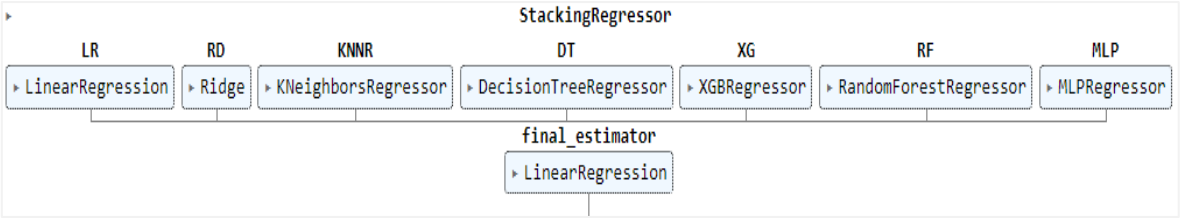


Figure 2.6 : Schematic representation of the stacking regressor trained on Central Rand Pump B that was used for transfer learning to predict the KGR inflow (West Rand) sulphate values.

TL was conducted by passing the cleaned KGR inflow dataframe as a new set of predictors ("X_test_new") to the already trained stacking regressor from Central Rand Pump B. Predictions for the sulphate values were made using the stacked model and the predicted sulphate values (SUL) were saved as a new dataframe.

Since normalized TEMP and FE values were used as predictors, the target SUL values were obtained in their normalized form as well. Thus, un-normalization of the sulphate values was conducted by using the minimum and maximum sulphate values from Central Rand Pump B and solving for x in **Equation 2.2** which results in **Equation 2.7**, i.e.,

$$x_i = y(max - min) + min$$

Equation 2.7

Central Rand Pump B's sulphate minimum and maximum values were used since these were the only unnormalized values available as a result of no sulphate values being recorded on the West Rand plant. Since the KGR inflow data followed a similar distribution to Central Rand Pump B, it was determined that Central Rand Pump B's sulphate minimum and maximum values could be used as a reasonable approximation for the minimum and maximum KGR inflow sulphate values.

The final KGR inflow sulphate values (both normalized and un-normalized) were visualized and their descriptive statistics were analysed before saving them as a new Excel file as seen in Table 2.7.

Table 2.7: Screenshot of Excel spreadsheet containing final KGR inflow sulphate values (both normalized and un-normalized) predicted using TL.

SUL - KGR inflow (normalized)	SUL - KGR inflow (un-normalized)
0.53324281	0.008761646
0.025139573	0.000414018
0.259952712	0.004271763
0.521233867	0.008564351
0.037610911	0.00061891
0.526905201	0.008657526
0.272868588	0.004483958

The code to conduct TL and its related visualizations for the KGR inflow can be found in "Appendix H".

2.2.3.3 “Treated flow into KGR” (Treated AMD) of West Rand dataset – Data precleaning, data cleaning and TL

The same data pre-cleaning, cleaning and TL process was followed as outlined in the previous section (“2.2.3.2 “KGR inflow” (untreated AMD) of West Rand dataset”) using the same stacking regressor trained on Central Rand Pump B’s dataset. The only difference was that the initial data fed was regarding the treated AMD from the West Rand plant, therefore, predictions of sulphate levels were now for treated AMD. Central Rand Pump B’s stacking regressor (i.e., model that was trained on untreated AMD) was used to predict sulphate levels of treated AMD since EDA revealed that they had similar statistical distributions of the predictors required (i.e., TEMP and FE). Therefore, it was concluded that Central Rand Pump B’s stacking regressor was suitable to be used to predict sulphate levels in treated water since similarities existed between the datasets.

Another complication that arose was that the Central Rand Treated Water trained models all relied on TDS as a predictor which, unfortunately, was not measured by laboratory technicians in the West Rand treatment plant.

The code used to pre-clean, clean and conduct TL on the Treated AMD from the West Rand plant are not appended since it bears an extremely high similarity to the code already appended for the KGR inflow.

2.2.4 GUI creation using ChatGPT prompt engineering

The accurately trained and tested Central Rand Pump B SE-ML model was used as the “Regression model” to create a GUI to predict sulphate levels in AMD. From PCA and RFE, only the temperature (°C) and iron (mg/L) levels were needed in order to predict sulphate levels. A total of six inputs were needed in order to provide an accurate sulphate level (mg/L) prediction. These were the minimum (“Min”), maximum (“Max”) and actual value (“Value”) pertaining to the temperature and iron concentrations in an AMD sample. The minimum and maximum values were needed in order to normalize the value such that an accurate sulphate value can be predicted. In cases where no minimum nor maximum values are available, estimates based on historical knowledge can be provided instead.

The website default “August 3, 2023” version of ChatGPT-3.5 was prompted through a series of seven conversations in order to create the GUI named “Sulfind” (i.e., to find sulphate levels) using the Python library “Tkinter”. Several of the conversations were used to troubleshoot any problems along the way as well as to learn how to use Jupyter Notebooks and Anaconda Navigator. This was achieved by combining the coding and debugging abilities of ChatGPT with a human negative error message feedback loop in order to generate an error-free Python

code needed for the GUI to function properly. The final code used to produce the GUI can be found in Appendix L.

An interesting discovery was made that GUIs cannot be created using the Google Colaboratory environment, which was where the original stacking ensemble regression model was trained. Since Google Colab is a cloud-based virtual environment, it does not have a graphical interface needed to import Tkinter library that creates the GUI.[158] GUIs could be created in the Jupyter Notebook environment (where Tkinter can be imported easily), however, this required a different setup and required that Python packages (such as Numpy, Pandas, Scikit-learn, etc.) be installed as opposed to the Google Colaboratory environment which barely needed any setup prior to conducting machine learning. ChatGPT proved to be incredibly useful in providing accurate and concise information pertaining to how to install these packages, indicated where challenges were and how to operate the Jupyter Notebook and the Anaconda environment. As such, the GUI was created using Anaconda Navigator in the Jupyter Notebook environment equipped with a Python3 kernel.

2.2.4.1 Deployment of GUI to the web guided by ChatGPT

In order to further improve the accessibility and “no code” nature of the developed GUI, ChatGPT was prompted as to how to convert the GUI from being based in a Jupyter Notebook to being available on the web through a simple URL link. This was achieved using the Python library “Flask” in order to create the website for the GUI, embed the regression model into it and take user input. The cloud-based services provider, “Heroku”, was used to host, build and deploy the simplified web version of the GUI for the general public to access and use.

This was done in order to avoid people becoming confused with the Jupyter Notebook interface since the GUI can now be easily accessed through a browser by simply clicking a link (https://sulfind-webapp-gui-c4ce20a6aee6.herokuapp.com/). A schematic of the “Sulfind” GUI creation and deployment guided by ChatGPT can be found in Figure 2.7 and the final “Flask” code for the web app can be found in Appendix M.

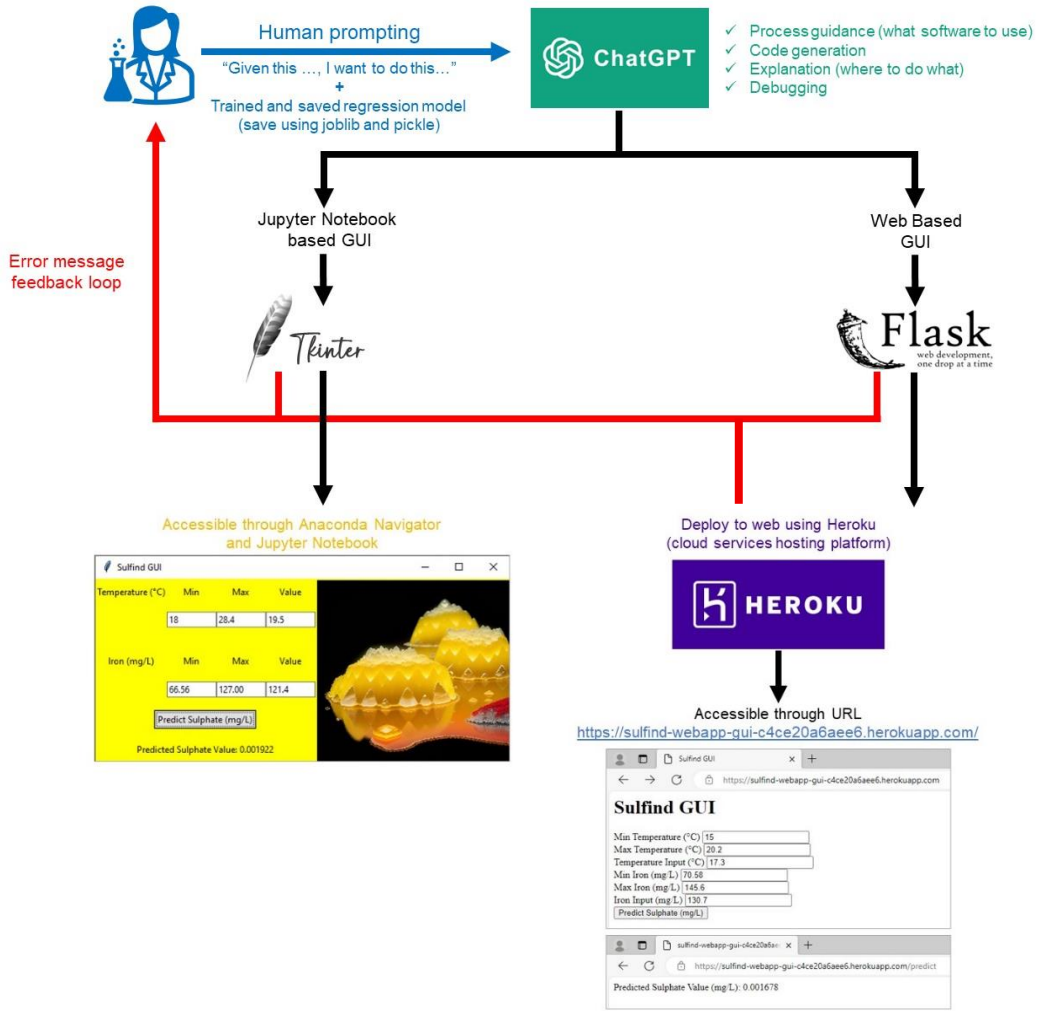


Figure 2.7: Schematic outline of ChatGPT assisted "Sulfind" GUI creation (both Jupyter Notebook and web based).

Screenshots of interesting portions of the conversations with ChatGPT can be found in Appendix N.

CHAPTER 3 : RESULTS AND DISCUSSION

This chapter presents the findings of the study based on the discussed research approach and methodology protocols. The results are presented and accompanied by their discussion, citing literature similarities and differences where possible.

3.1 CENTRAL RAND DATASET

Overall, as seen in Figure 3.1 and Table A1 (which can be found in "Appendix I"), EDA revealed that all water quality parameters changed after the treatment procedure. Quite noticeably, it was found that the temperature, iron and manganese levels decreased after the treatment process, whereas the pH and sulphate levels increased after the treatment process. This was in accordance with what was expected and indicted that the Central Rand treatment plant was functioning properly since most of the metals (FE and MN) would have been removed as sludge during the treatment process and would, thus, not be present in the treated water.[32]

The pH levels of the treated water were expected to increase since the treatment process effectively adds limestone which is an alkaline neutralizing agent that would react with any acidic species to neutralize them, thereby, increasing the pH of the treated water.[24,33]

Due to the low pH nature of AMD, minerals present such as Pyrite (FeS_2), Pyrrhotite ($Fe_{(1-x)}S$) and Chalcopyrite ($CuFeS_2$) dissolve releasing more sulphates and metals into solution. Upon neutralisation during the treatment process, the metals are removed as sludge (mainly co-precipitating with Fe hydroxides).

Sulphates also precipitate out mainly as gypsum ($CaSO_4.2H_2O$), although they largely remain in solution as the most dominant specie.[159] Temperature can be seen to decrease since energy is required to break apart the metal sulphate bonds and this would result in energy from neighbouring molecules being absorbed, thus, resulting in a decrease in temperature.[160]

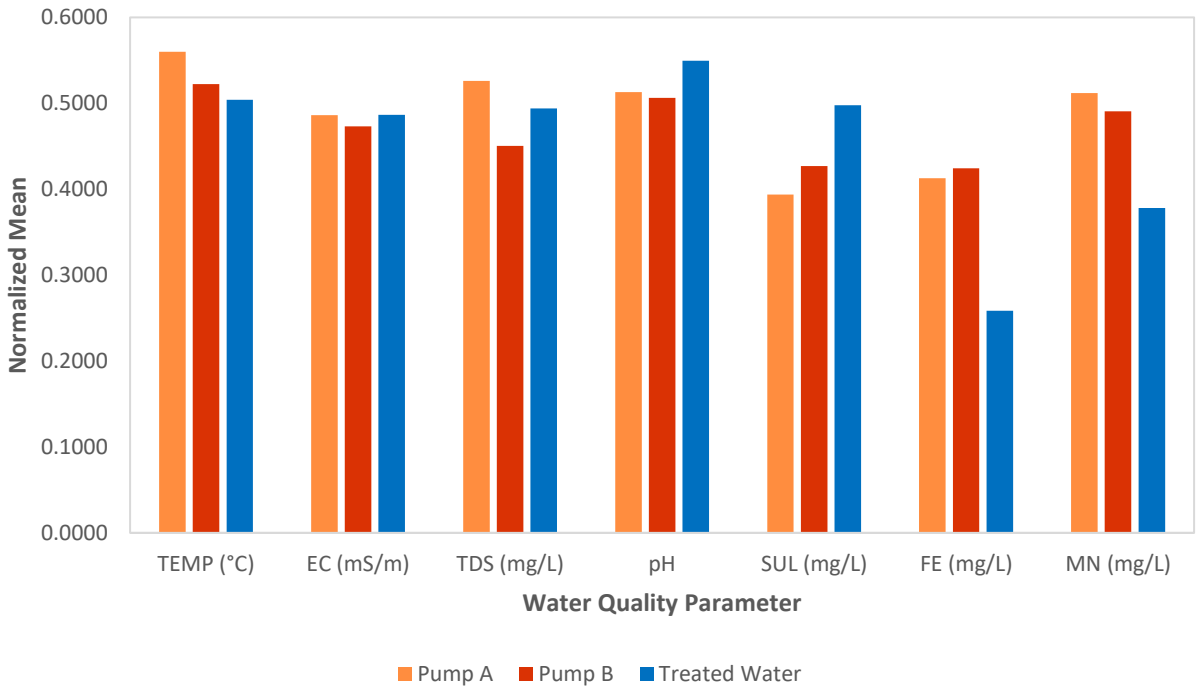


Figure 3.1 : Changes in water quality parameters in the Central Rand AMD treatment plant (Pump A and Pump B are AMD feed prior to any treatment procedures)

As seen in Figure 3.2, the original Pump A dataset had 2136 rows of data, however, after data-precleaning only 1272 rows of data remained which indicated that there were a lot of missing values or incomplete data present in the original dataset. A further reduction in the size of the dataset was noticed after geochemical modelling since several outliers and incorrect datapoints were identified during simulation runs and were, thus, removed from the dataset resulting in 1107 rows of data remaining. The same decreasing trend was observed for Pump B, with the data originally starting off with 2136 rows, after pre-cleaning and geochemical modelling 1396 rows remained and finally 1189 rows of data remained after outlier removal. For the Treated Water originally there were 2136 rows of data that were decreased to 1530 rows after pre-cleaning, further decreased to 1529 rows after geochemical modelling and lastly only 1197 rows remained after outlier removal.

Incorrect datapoints that were removed during each of the cleaning and outlier steps were immediate indicators of the inherent inaccuracies present in the dataset which provided evidence for the inconsistencies and errors present in traditional chemistry approaches.

Pump A, Pump B and Treated Water are further discussed individually in greater detail in the following sections. All units for water quality parameters are the same as those in Figure 3.2, and have been omitted from some figures in order to prevent cluttering and improve readability.

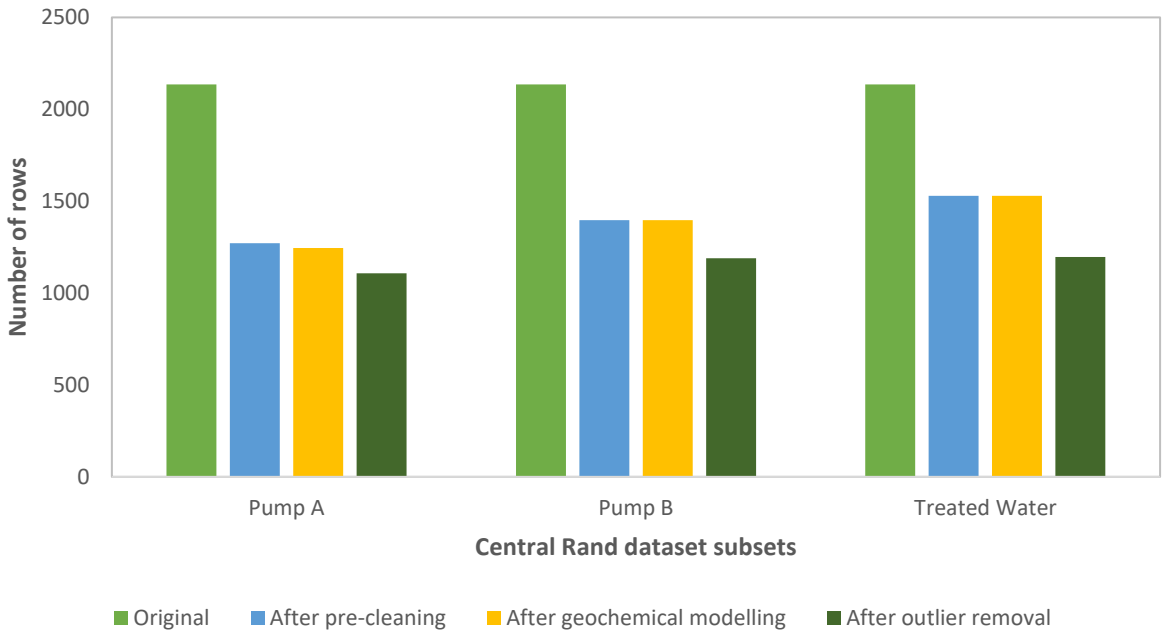


Figure 3.2 : Reduction in size of individual datasets pertaining to the Central Rand AMD treatment plant after data pre-cleaning, geochemical modelling and data cleaning (outlier removal).

3.1.2 AMD Feed - Pump A

3.1.2.1 Statistics, visualizations and correlations of Pump A

Upon inspection of the sulphate concentrations for Pump A, it was found that it originally lied in the range of 3000 – 4000 mg/L, however after geochemical modelling these values lied in the range of 0.002 – 0.0015 mg/L which was five orders of magnitude smaller than the original values. This dramatic decrease indicated that the original sulphate values recorded were extremely high and inaccurate since they did not thermodynamically balance with the other cationic water quality parameters measured (i.e., FE and MN). This highted the experimental inaccuracies associated with the traditional analytical procedures used to measure sulphate concentrations in AMD and further served as experimental support for the need to conduct geochemical modelling as a data pre-cleaning step.

As seen in Figure 3.3a and Figure 3.4a, initial data visualization revealed highly skewed distributions in the histograms, several sharp and pointed peaks in the density plots and the presence of several outliers in the box and whisker diagrams. This served as immediate evidence that the presence of outliers adversely affected the distributions of the data and, therefore, needed to be removed from the data. As observed in Figure 3.4a, A-pH was majorly skewed resulting in only a single bar being visible in its histogram. Upon inspection of its corresponding box and whisker plot in Figure 3.3a, it can clearly be seen that a recorded pH of above 5000 was responsible for this extreme irregularity. All statistical values used to plot the graphs can be found in Table A2 (in “Appendix I”).

From Figure 3.3b and Figure 3.4b, the distributions drastically improved after outlier removal and tended to more normal distributions centred around a mean value. The actual spread of values could much more easily be observed and interpreted. The density plot traces for Pump A showed several shoulders, sharp peaks and split peaks (bimodal distributions) which indicated that the spread of values for each variable was centred around one or two main values. This could be due to a variety of reasons such as the season and the weather whereby one value is favoured in hot, sunny conditions and other values in favoured in colder conditions.[159,160] As seen in Figure 3.3b, there were very few outliers remaining and Pump A's distribution was now normal. Normalizations had no effect on the distributions of data, rather it only scaled the values to lie within the range of 0 to 1 in order to reduce the effects of differing scales.

From Figure 3.3b, the mean temperature was 26.5 °C and the mean pH was 5.8 for the untreated AMD. This was in accordance with what was expected since the untreated AMD was expected to have a low pH signifying acidic conditions. From Figure 3.4b, SUL and FE have the same double hump shape which serves as experimental evidence for their extremely strong correlation and also indicates the presence of sulphide minerals such as pyrite.[159] MN, TEMP and EC showed similar depressed hump shapes and this indicated that these parameters were correlated to each other. TDS and pH had unique shapes which indicated that they were not dependant on any other water quality parameter.

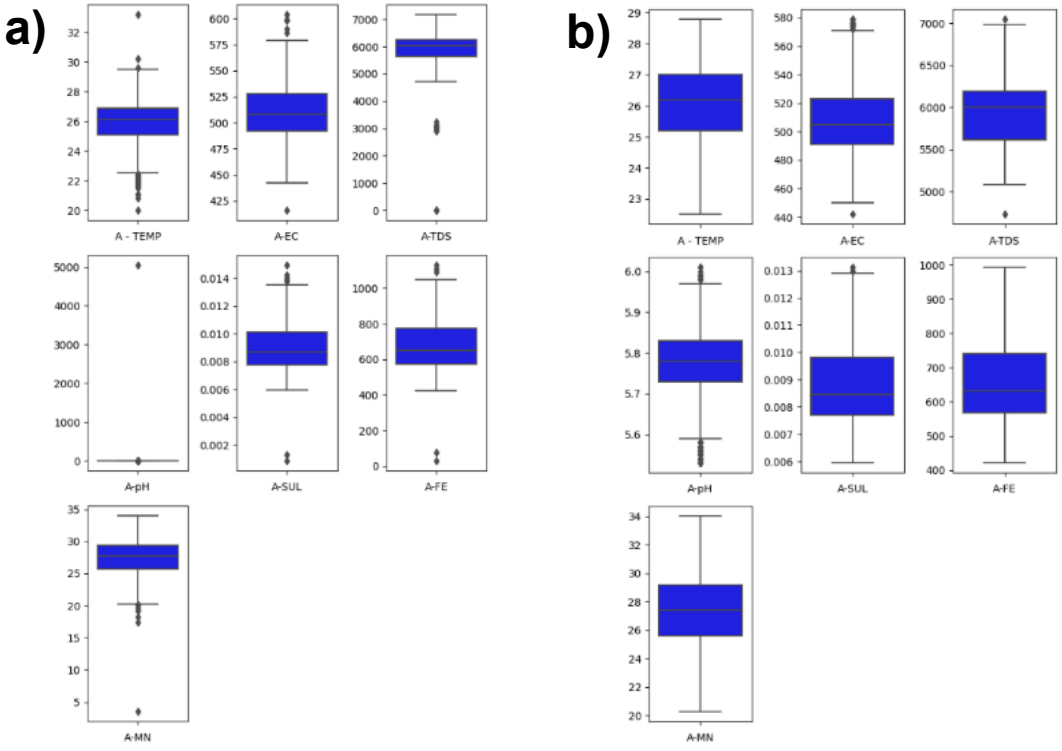


Figure 3.3 : Box and whisker diagrams of Central Rand Pump A indicating its statistical distribution a) Prior to outlier removal and b) After outlier removal.

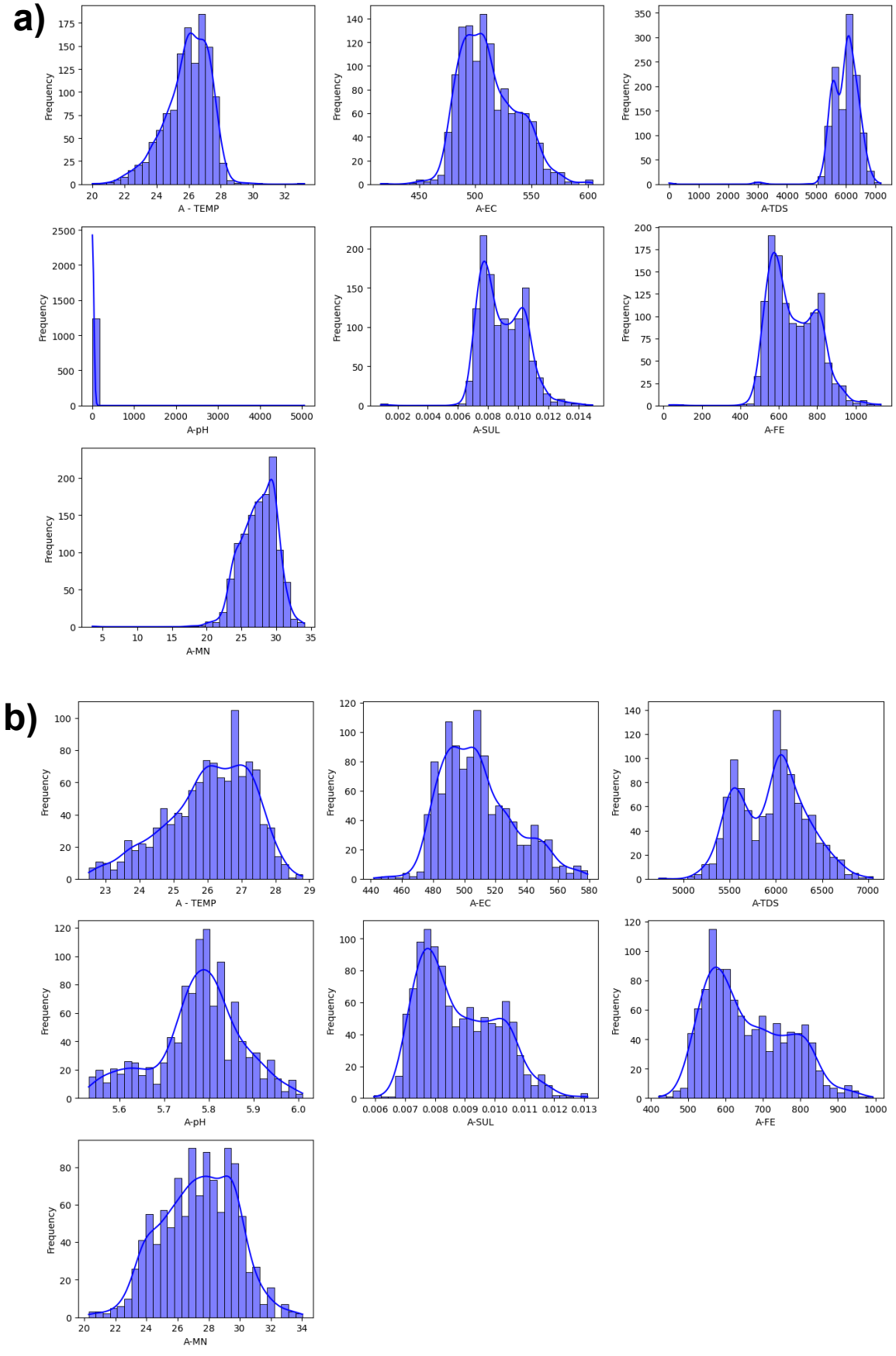


Figure 3.4 : Histogram and density plot traces indicating the statistical distribution of Central Rand Pump A a) Prior to outlier removal and b) After outlier removal.

The scatter matrix (Figure 3.5) when used in conjunction with the correlation matrix (Figure 3.6) indicate that TEMP and pH are negatively correlated (i.e., have an indirect relationship) with all other water quality parameters and that all other parameter are positively correlated (have a direct relationship) with each other. This showed that Pump A's variables had a significant amount of relation to each other and that several patterns were indeed present in the data. An increase in pH would increase the rate of dissolution of the minerals present in AMD, thus, basic ions would be released into solution that would neutralize the acidic species present, thereby, increasing the pH.[159,160]

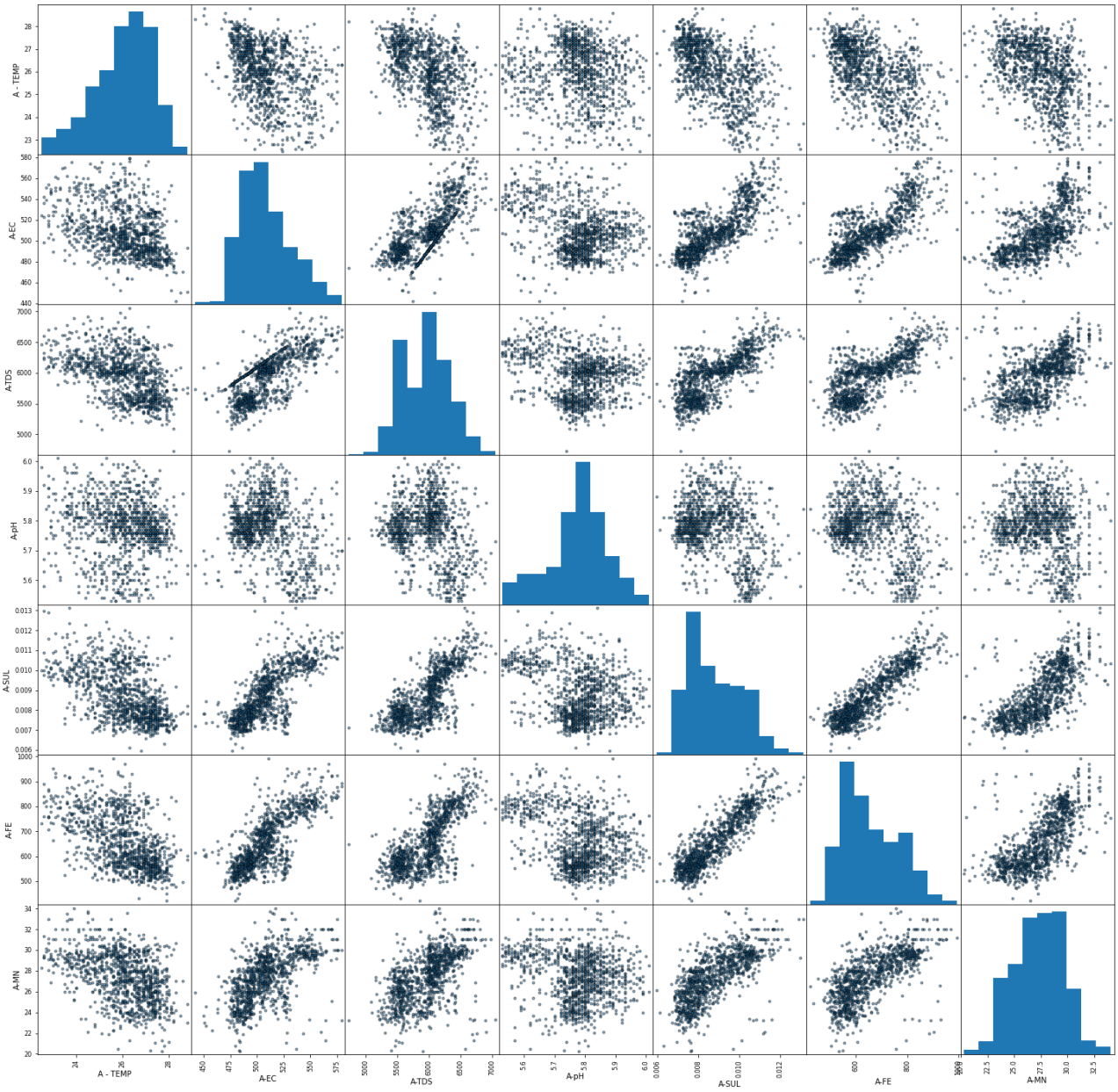


Figure 3.5 : Scatter matrix of Central Rand Pump A depicting the relationship between water quality parameters.

As seen in Figure 3.6, the strongest negative correlation variable was TEMP and this indicated that as temperature increased, all the other parameters decreased and this was noted as a significant finding since temperature is directly dependant on the season. FE was seen to be the most strongly positively correlated variable meaning that as FE increased, so too did the other variables (besides temperature). The strongest positive correlation was observed to be between FE and SUL (0.89).

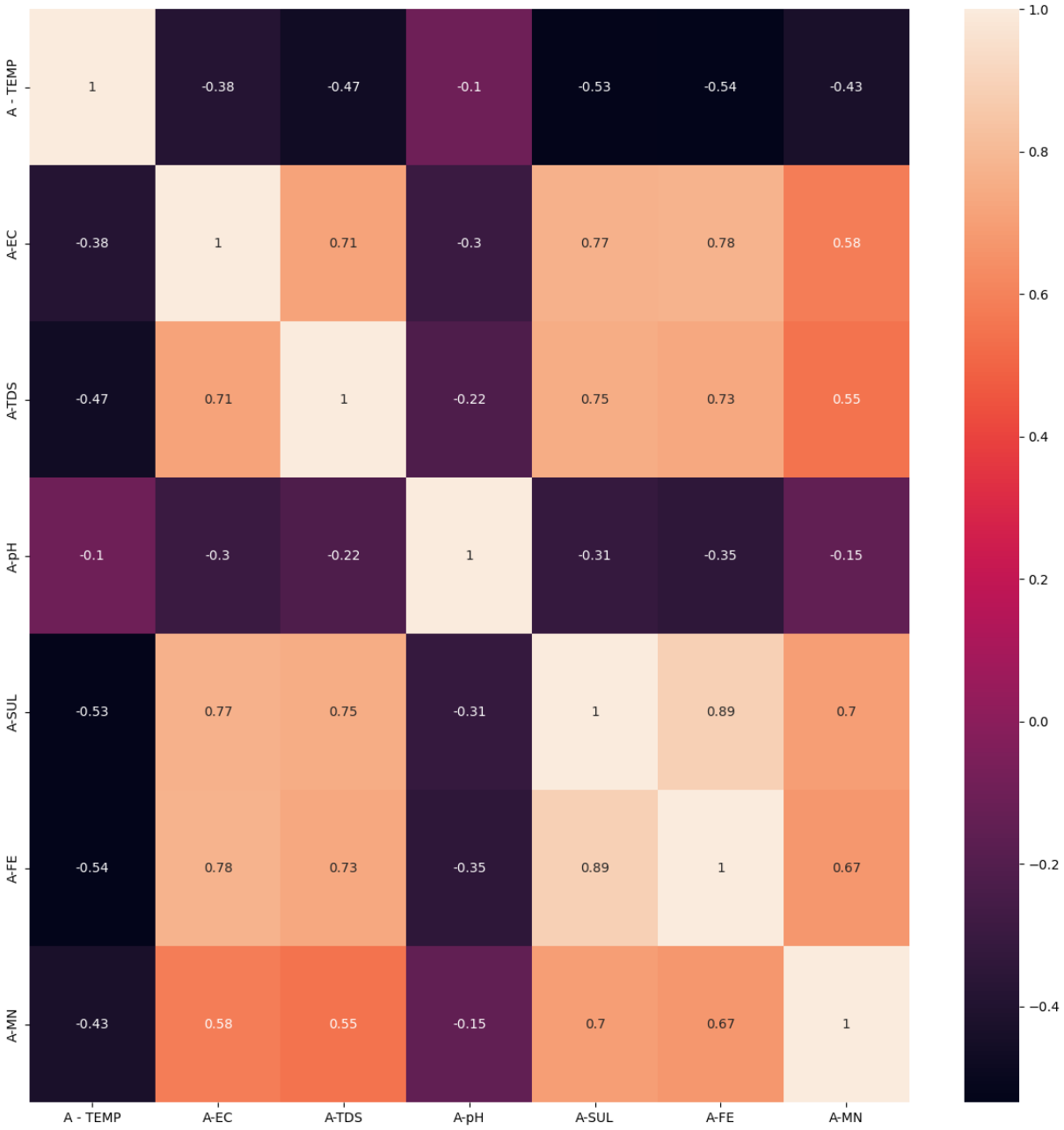


Figure 3.6 : Correlation matrix (heat map) of Central Rand Pump A depicting the relationship between water quality parameters (darker colours indicates a strong negative correlation relationship whilst lighter colours indicate a strong positive correlation).

3.1.2.2 Feature extraction and baseline regression algorithm comparison

Upon conducting PCA, from Figure 3.7a-c, two clusters were observed in the bi-plot and this was consistent with the double hump behaviour seen in the density plots which gave further evidence for the temperature and seasonality (hot and cold environments) dependence of the water quality parameters.

The loadings plot (Figure 3.7b) revealed that A-MN, A-TDS, A-FE and A-EC were strongly related to A-SUL. The Scree plot (Figure 3.7d) indicated that only two components were needed to explain more 78% of the variance in the data and this was deemed to be sufficient. These variables were deemed to be A-TDS and A-FE. Thus, a new dataframe consisting of 1107 rows of data with A-TDS and A-FE as predictors was constructed in order to predict A-SUL values as a target.

TDS has been found to be strongly correlated to metals since it is a measure of the inorganic material content (dissolved metals, calcium, magnesium, sulphates, etc) present in the water.[161] Pyrite is the primary source of Fe and sulphate in AMD, therefore their strong correlation is expected.[162]

Another explanation for the high FE and SUL correlation could be due to the concentrated sulphuric acid used to leach uranium oxides as a by-product from gold mine slurry.[163] However, this is mainly limited to those gold mines that extract uranium as a byproduct. Iron pyrites present in this gold sludge react with the concentrated sulphuric acid and are discharged as AMD.[163] The large number of mine tailings in the East, Central and West Rands contributes to high volumes of AMD that is pumped from the old mine shafts and aquifers. The pyrite in these tailings is undergoing continuous weathering, thus, ensuring a sustained release of AMD.

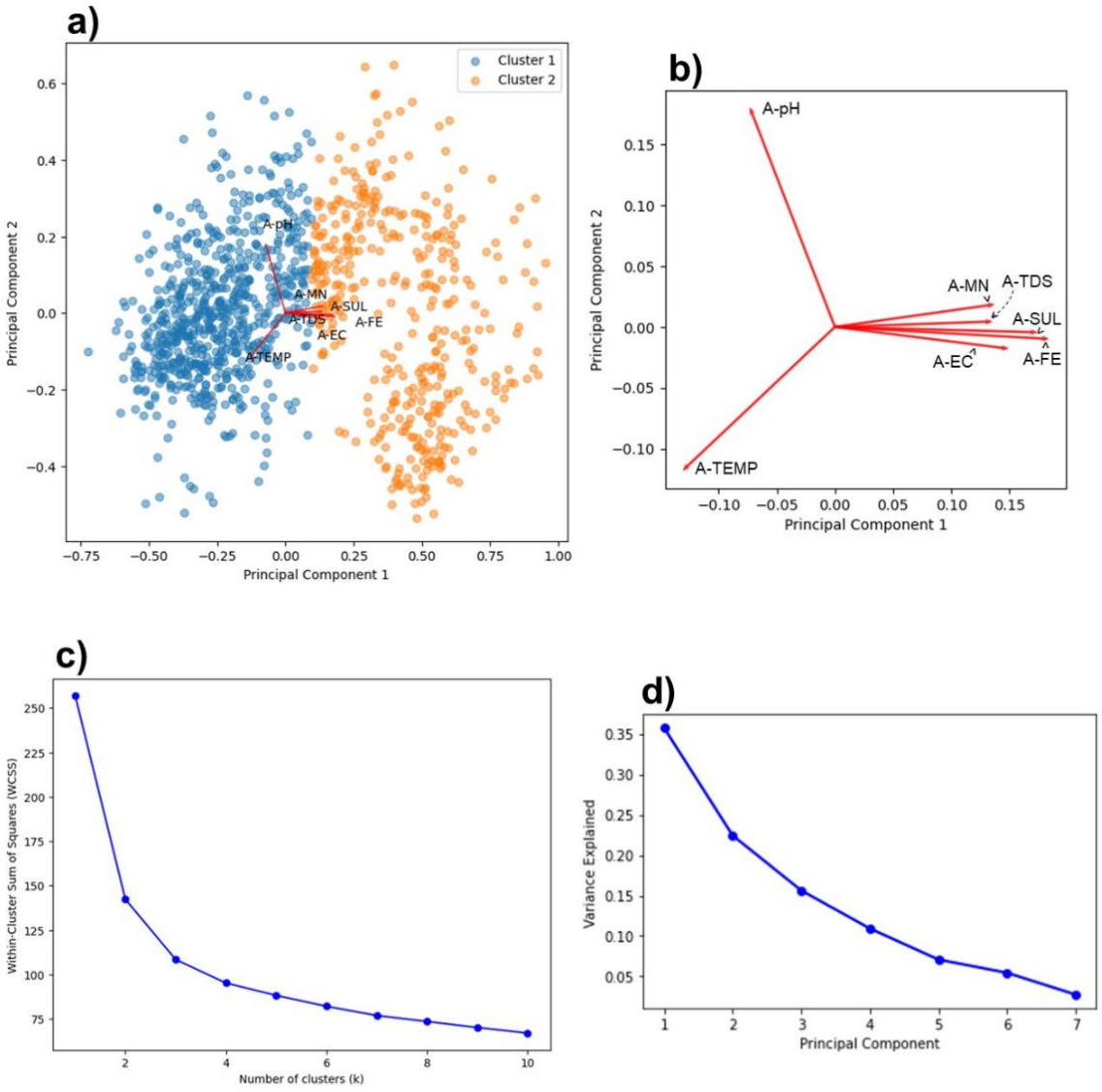


Figure 3.7 : Dimensionality reduction and feature extraction results obtained from PCA for Pump A, a) Bi-plot indicating the clustering of individual observations and its relation to the loadings plot, b) Expanded view of the loadings plot indicating the relationship between parameters, c) Elbow method plot indicating the optimal number of clusters and d) Scree plot indicating the amount of variance explained by each principal component.

As seen in Figure 3.8a, upon comparing individual machine learning algorithms in order to generate a baseline of which models performed the best for regression, it was found that the SVR performed the best (NMSE:-0.00532) whilst LASSO (NMSE:-0.03217) and EN (NMSE:-0.03217) performed the worst. All other models performed around the same level. Since there existed a high degree of correlation between variables (as seen in Figure 3.6) LASSO performed poorly since there was high bias and multicollinearity present in the data and LASSO only selects on variable and sets the rest to zero.[164] Since EN essentially combines LASSO and RD, the poorly performing LASSO could have tainted and ruined the predictive accuracy of EN due it inability to handle multicollinearity. The SVR is quite tolerant to outliers

and works well on smaller datasets[165], therefore, since Pump A's dataset only had 1107 rows of data the SVR showed good predictive accuracies.

As seen in Figure 3.8b, when combining all the baseline models into a single stacking regressor, the performance of the stacking regressor (NMSE:-0.00544) was better than that of the individual models except the SVR. This showed that combining the predictive accuracies of each of the models did improve the overall predictive performance of the model, however, the SVR still performed this best. This can be attributed to the SVR being tolerant towards outliers and appropriate for small datasets, whereas some of the other models combined in the stacking regressor were sensitive to outliers and required large datasets to be able to make accurate predictions. Therefore, it was decided to remove the bad performing models and create a new stacking regressor that only contained the best performing models in order to remove the tainted models (LASSO and EN). However, it is still worth nothing that the stacking regressor that included the LASSO and EN still performed exceptionally well having included them due to combining the "wisdom of the group".

As seen in Figure 3.8c, when combining only the best performing models into a single stacking regressor the predictive accuracy of the stacking regressor (NMSE:-0.00546) did not drastically improve and the SVR still performed the best. Thus, it was concluded that including worse performing models (LASSO and EN) in the stacking regressor had no effect on its predictive performance as long as the well performing models were present (SVR).

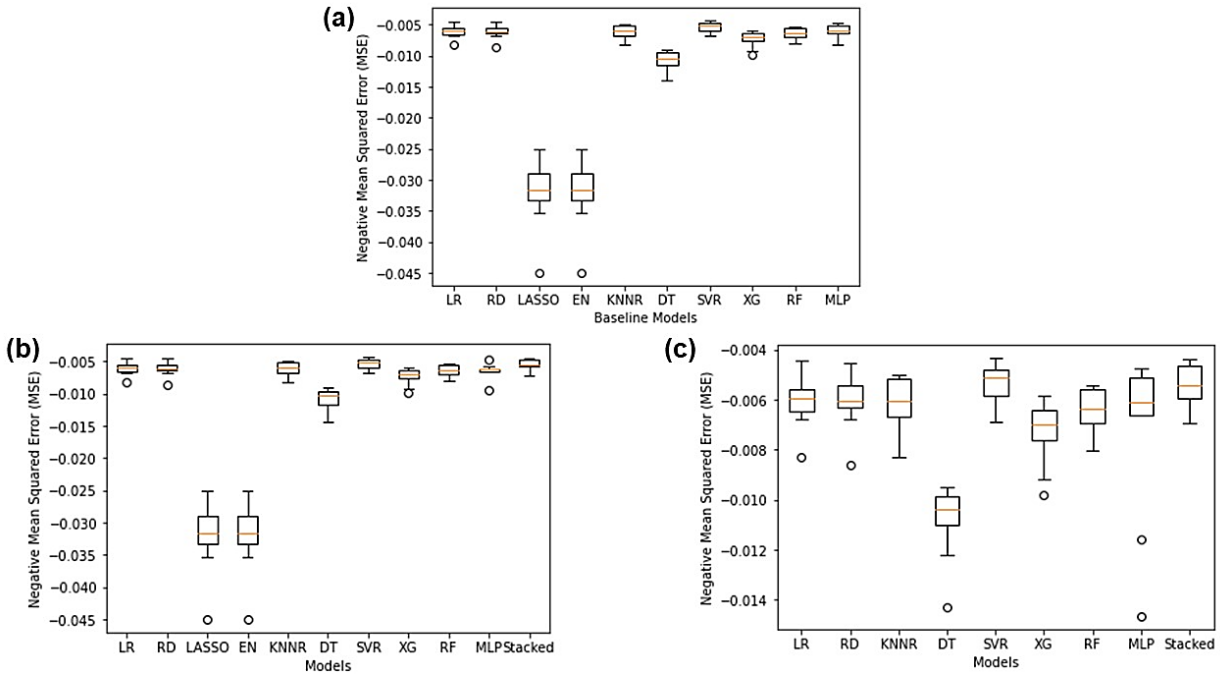


Figure 3.8 : Regression algorithm NMSE comparison for Central Rand Pump A showing a) All individual baseline models, b) All models and a stacking regressor containing all the models and c) All well performing models and a stacking regressor containing all the best performing models.

Upon testing the models, as seen in Figure 3.9a and Figure 3.9b, each of the baseline models and the stacking regressor containing all the models did achieve low MSE, low MAE and high R^2 values besides the LASSO and EN which achieved high MSE, high MAE and extremely low R^2 values (0.033252, 0.153985 and -0.000069 respectively for both). This served as further experimental evidence that the LASSO and EN were not the optimal models to be used to predict sulphate levels in Central Rand Pump A. However, during testing it was found that the stacking regressor performed the best (MSE:0.005252, MAE:0.055598 and R^2:0.842041) and this could be attributed to the reduced overfitting of the stacking regressor as opposed to the SVR (MSE:0.005275, MAE:0.055935 and R^2:0.841351) since the stacking regressor combined the individual models, thereby, averaging out the individual biases and variances.

When testing the stacking regressor using only the best performing models (MSE:0.005335, MAE:0.055679, R^2:0.839538), as seen in Figure 3.9c and Figure 3.9d, it was no longer clear cut as to whether the SVR or the stacking regressor performed better. The MSE, MAE and R^2 values also did not drastically improve from the stacking regressor that contained all the models. This served as further experimental evidence that the inclusion of worse performing models in the stacking regressor had no effect on its overall performance in Pump A.

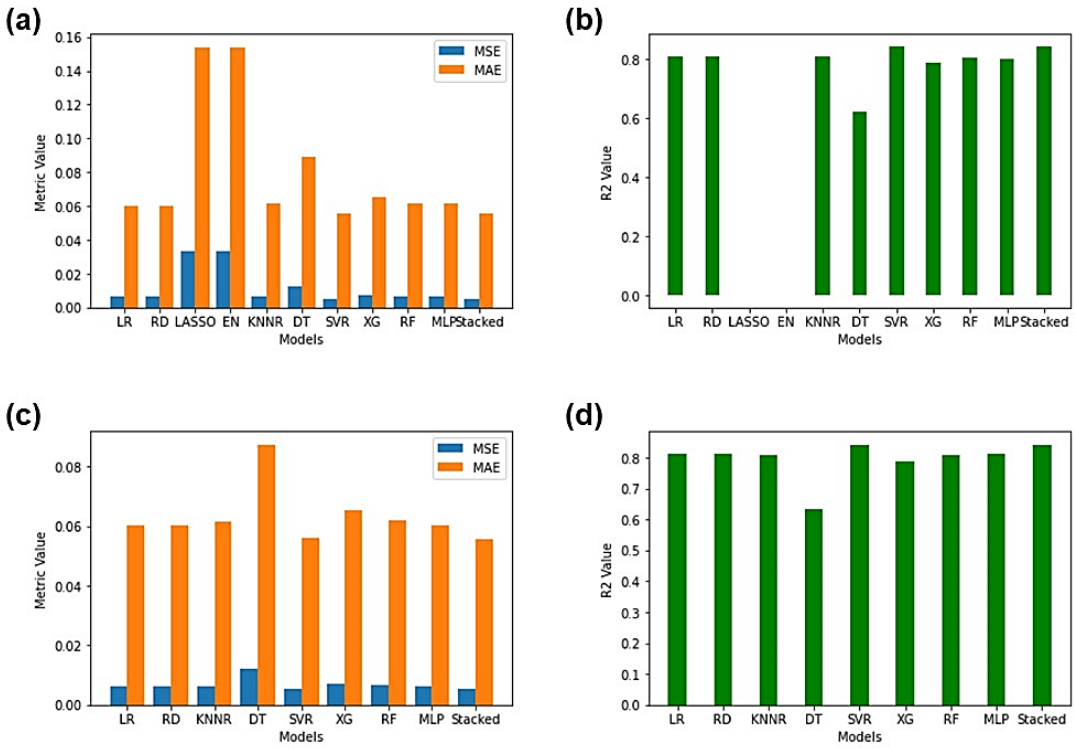


Figure 3.9 : Testing statistics (MSE, MAE and R^2) accuracy comparison of regression models trained on Central Rand Pump A a-b) Stacking regressor using all models and c-d) Stacking regressor using only the best performing models.

Table A3 containing the exact NMSE, MSE, MAE and R^2 values for each of the models found in Figure 3.8 and Figure 3.9 can be found in “Appendix I”.

3.1.3 AMD Feed - Pump B

The original sulphate levels pertaining to Central Rand Pump B were in the range of 3000-4000 mg/L, however, after geochemical modelling these values now lied in the range of 0.002-0.0015 mg/L. This decrease was consistent with that seen in Pump A and can be attributed to the lack of cationic species (i.e., calcium and magnesium) measured which had an effect on the charge balance.

As seen in Figure 3.10a and Figure 3.11a, the original distributions of the data were heavily skewed due to the presence of a significant amount of outliers. The presence of single bars and sharp peaks in Figure 3.11a were immediate indicators of outliers that were observed in Figure 3.10a. Therefore, outlier removal was an incredibly important data cleaning step since this helped to visualize and understand the inherent patterns present in the data much better. Outlier removal was, therefore, extremely necessary to improve the accuracy of the models since most models rely on or assume a normal distribution of data. After outlier removal, as observed in Figure 3.10b and Figure 3.11b, the distributions drastically changed and the same double hump behaviour, as seen for Pump A, was also observed for Pump B. This once again could be attributed to the seasonal dependence of the water quality parameters. Strong iron-sulphate correlations were once again observed which once again hinted at the presence of dissolved pyrites. All the statistical values used to plot the graphs can be found in Table A4 under “Appendix I”.

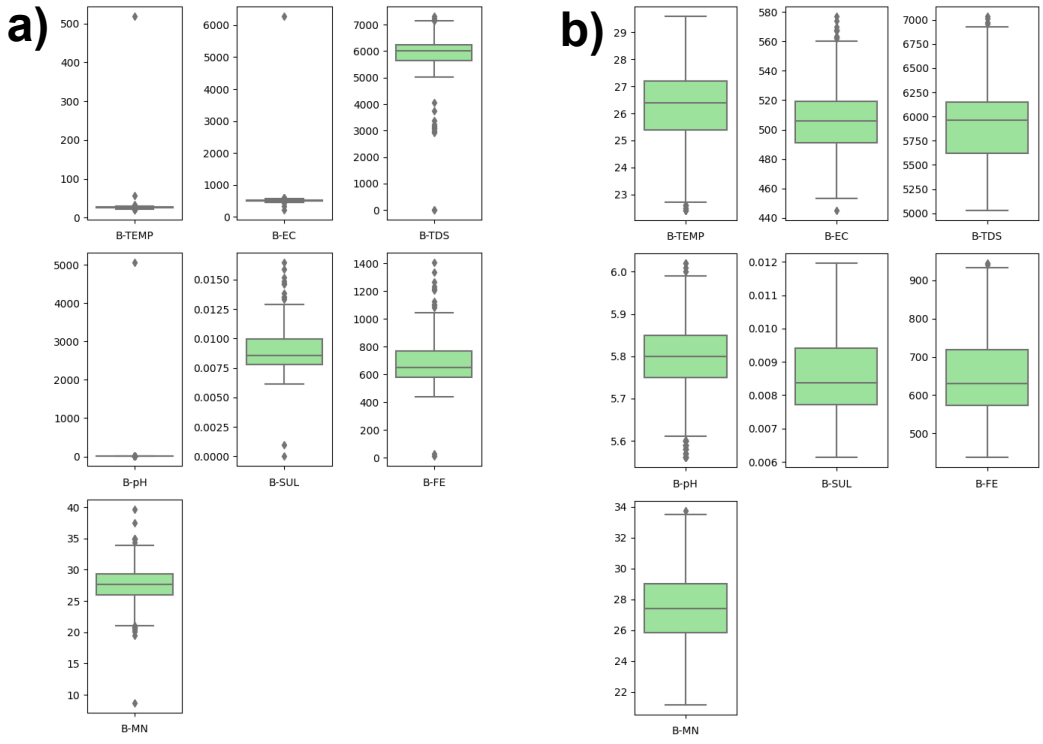


Figure 3.10 : Box and whisker diagrams of Central Rand Pump B indicating its statistical distribution a) Prior to outlier removal and b) After outlier removal.

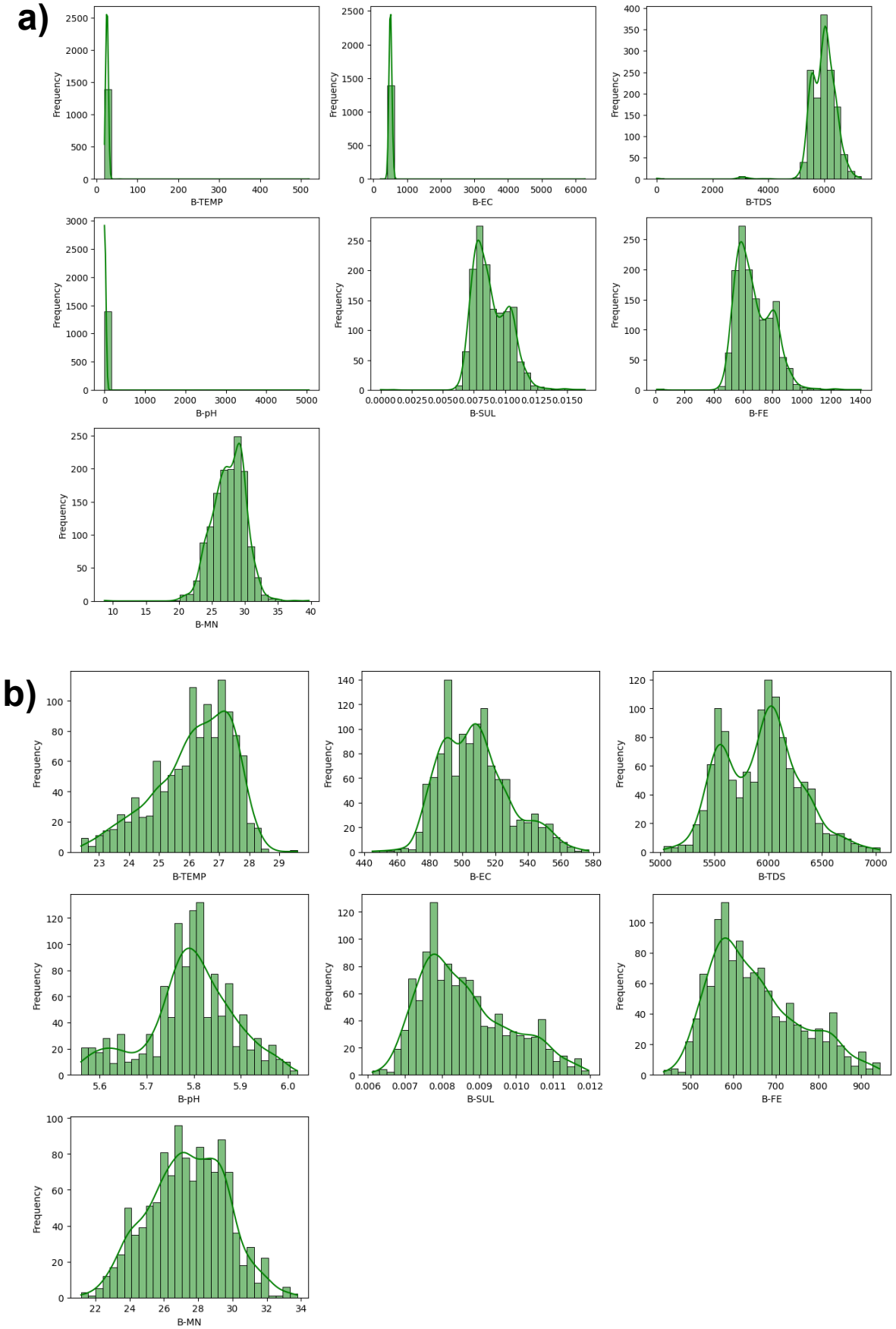


Figure 3.11 : Histogram and density plot traces indicating the statistical distribution of Central Rand Pump B a) Prior to outlier removal and b) After outlier removal.

From the scatter matrix (Figure 3.12) and its corresponding heat map (Figure 3.13), temperature was still negatively correlated to all other water quality parameters and this was the same trend observed for Pump A. Iron was perfectly positively correlated with sulphates since it attained a correlation of 1.0 and this strongly hinted at the presence of dissolved pyrites as the main constituent in AMD. Electrical conductivity and TDS were even more positively correlated with sulphates than Pump A and pH showed extremely little correlation to all other water quality parameters which was the observed to be the same with Pump A.

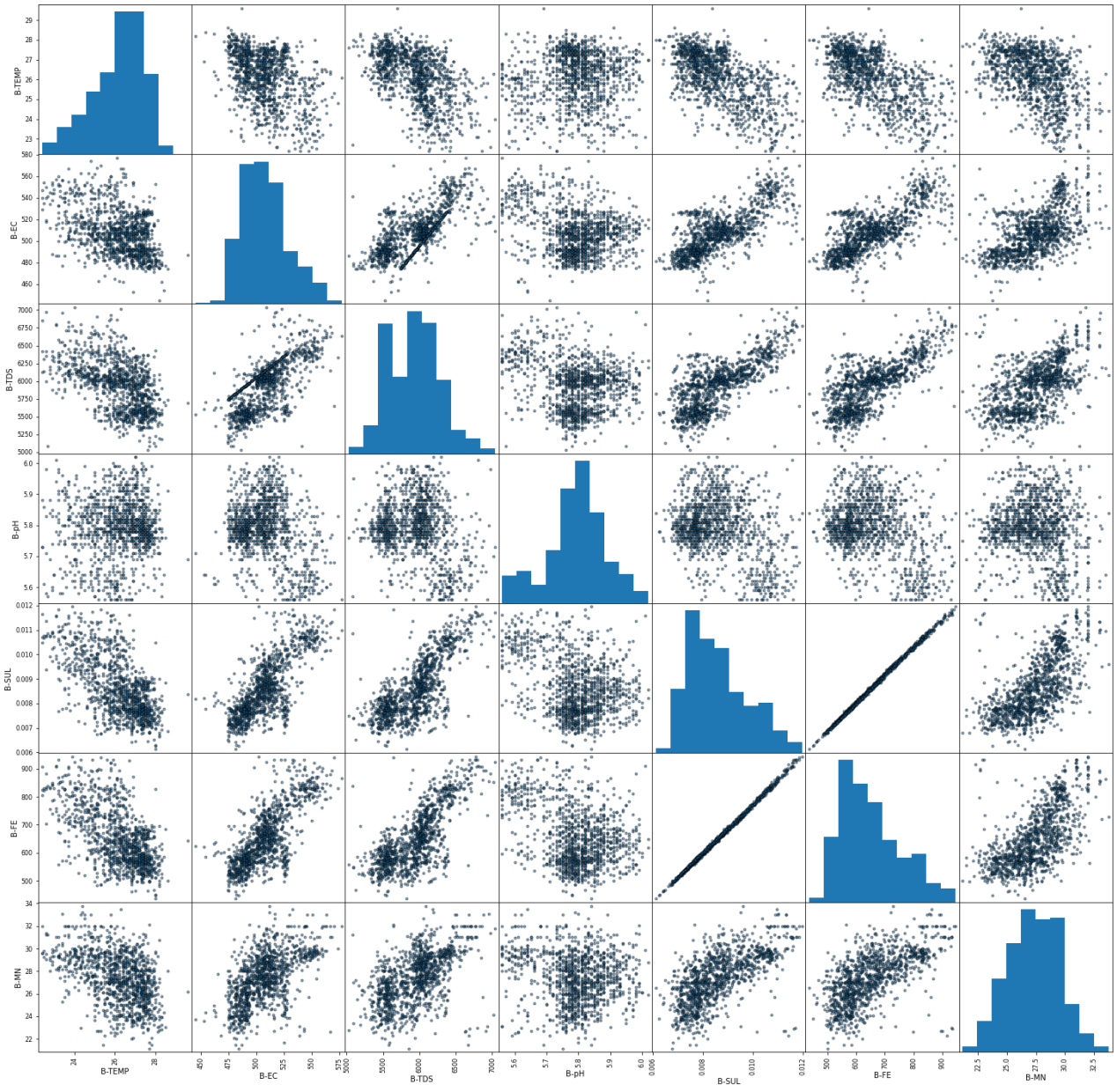


Figure 3.12 : Scatter matrix for Central Rand Pump B.

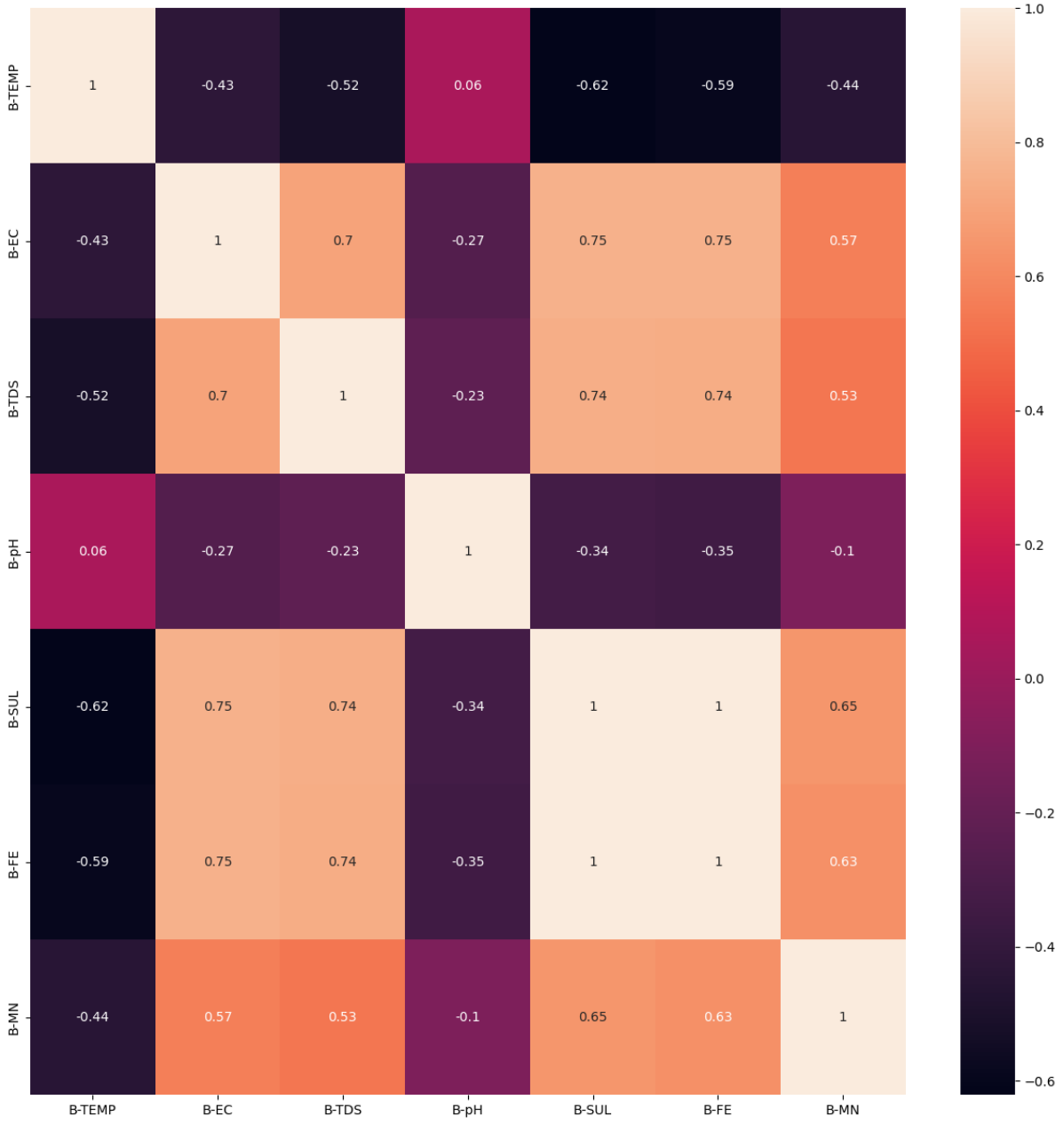


Figure 3.13 : Correlation matrix (heatmap) for Central Rand Pump B.

After normalizing the data to lie within the range of 0 to 1, no changes in the distributions and hence, the correlations of the data were observed. From the bi-plot (Figure 3.14a and Figure 3.14c) two clusters were again observed for Pump B and from the loadings plot (Figure 3.14b) it was observed that the same parameters (B-MN, B-TDS, B-FE and B-EC) were highly correlated to B-SUL, which agreed with what was observed for Pump A. The Scree plot (Figure 3.14d) revealed that two components, viz, B-TEMP and B-FE, were needed to explain 79% of the variance in the data. This agreed with the trends observed in the scatter matrix and the correlation matrix (Figure 3.13) for Pump B that showed the temperature having the strongest negative correlation (-0.62) with sulphate and iron having the strongest positive correlation

(1.0) with sulphate. These two components (B-TEMP and B-FE) were selected to be the final predictors to be used in regression to predict B-SUL as the target value.

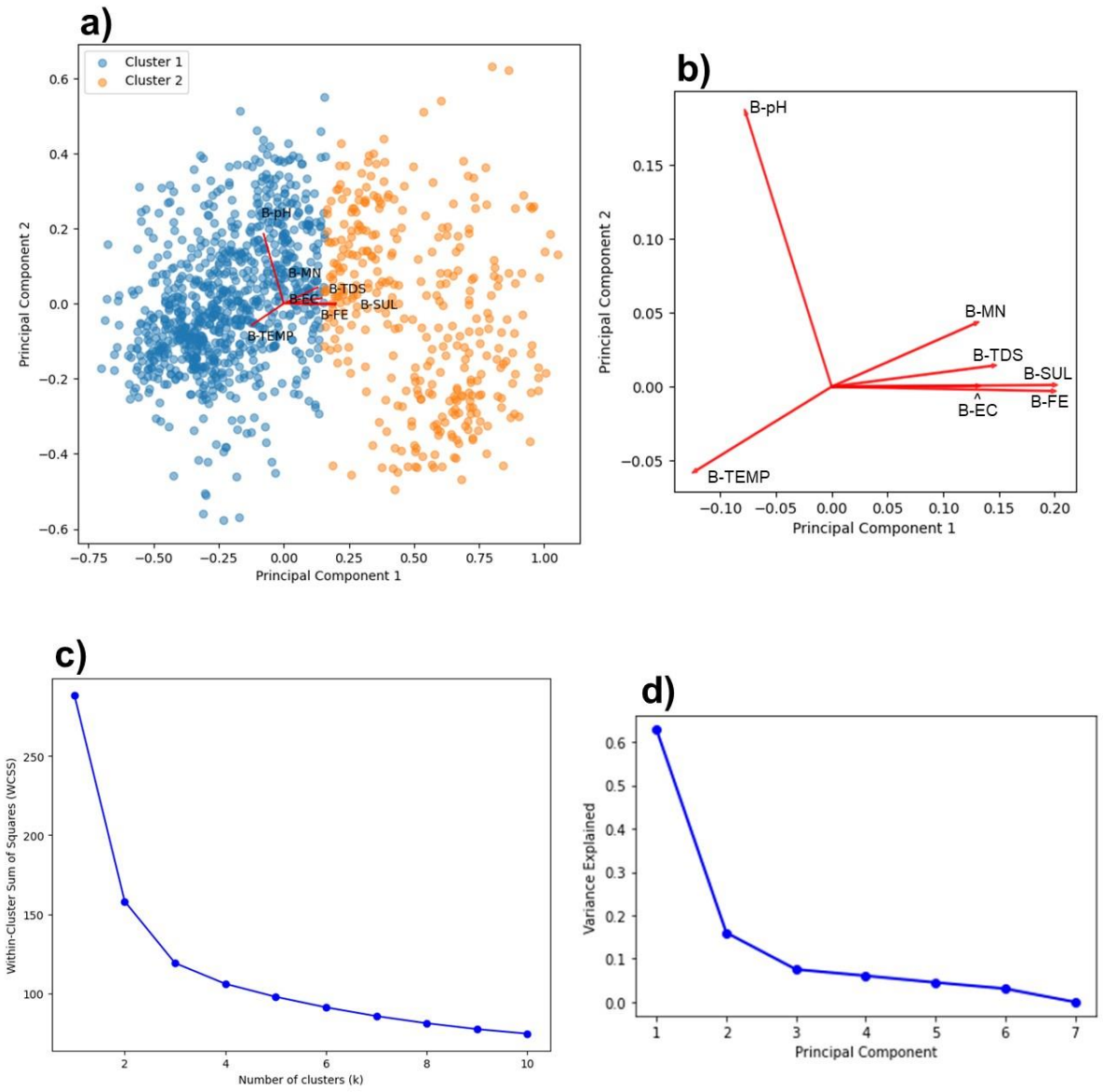


Figure 3.14 : Dimensionality reduction and feature extraction results obtained from PCA for Pump B, a) Bi-plot indicating the clustering of individual observations and its relation to the loadings plot, b) Expanded view of the loadings plot indicating the relationship between parameters, c) Elbow method plot indicating the optimal number of clusters and d) Scree plot indicating the amount of variance explained by each principal component.

As seen in Figure 3.15a, the LASSO and EN (NMSE:-0.04149) once again were the worst performing models due to their inability to handle multicollinearity. The best baseline model was LR (NMSE:-0.000018) and this correlated with what was observed in the scatter and correlation matrices since iron was perfectly linearly related to sulphates. Thus, LR was mathematically the perfect algorithm to use for predictions.

When combining all the models into a stacking regressor, as seen in Figure 3.15b, the stacking regressor (NMSE:-0.000016) outperformed LR which provided experimental evidence for the enhanced predictive power of ensemble models compared to single baseline models. Even though the stacking regressor contained poorly performing models such as LASSO and EN, it was still able to harness the combined predictive power of all the other models and was the best performing model as a result thereof.

When combining only the best performing models into a stacking regressor (NMSE:-0.000016) , as seen in Figure 3.15c, the performance of the stacking regressor did not improve compared to when all the models were combined. This was in accordance with what was observed for Pump A and provided further experimental evidence to the fact that including poorly performing models in a stacking regressor has no effect on its overall performance as long as the good performing models were present as well.

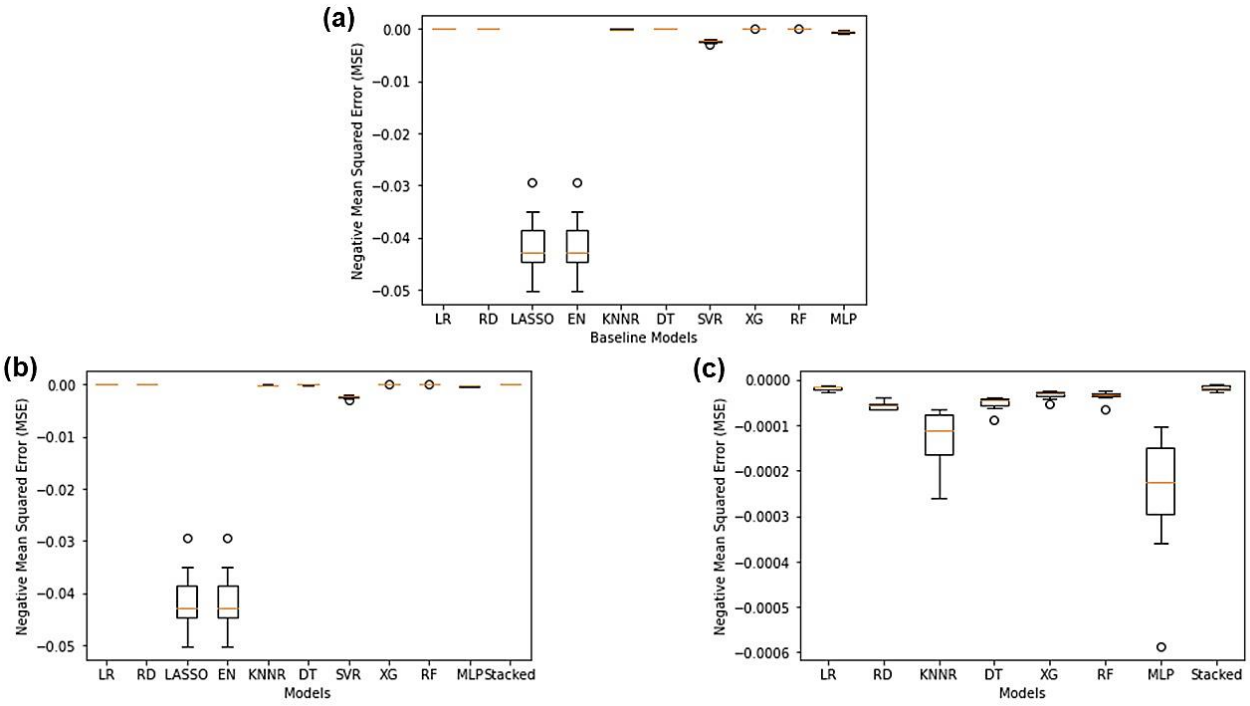


Figure 3.15 : Regression algorithm NMSE comparison for Central Rand Pump B showing a) All individual baseline models, b) All models and a stacking regressor containing all the models and c) All good performing models and a stacking regressor containing all the best performing models.

Upon testing the models, as depicted in Figure 3.16a and Figure 3.16b, LASSO and EN (MSE:0.043197, MAE:0.169071 and R^2:-0.000104) still performed the worst and the stacked ensemble using all the models performed the best (MSE:0.000011, MAE:0.002656 and R^2:0.999741). Noticeably LASSO and EN had negative R^2 values which indicated that the predictions made used these models were worse than simply using the mean value to replace missing values. As seen in Figure 3.16c, the stacking regressor using only good models (MSE:0.000011, MAE:0.002617 and R^2:0.999737) did show a slight improvement compared

to that of the stacking regressor that utilized all the models. The testing statistics (MSE, MAE and R^2) ultimately give a better and more accurate understanding of the model performance, therefore, it was determined that the best model to use for TL was the stacking regressor that utilized only the best performing models.

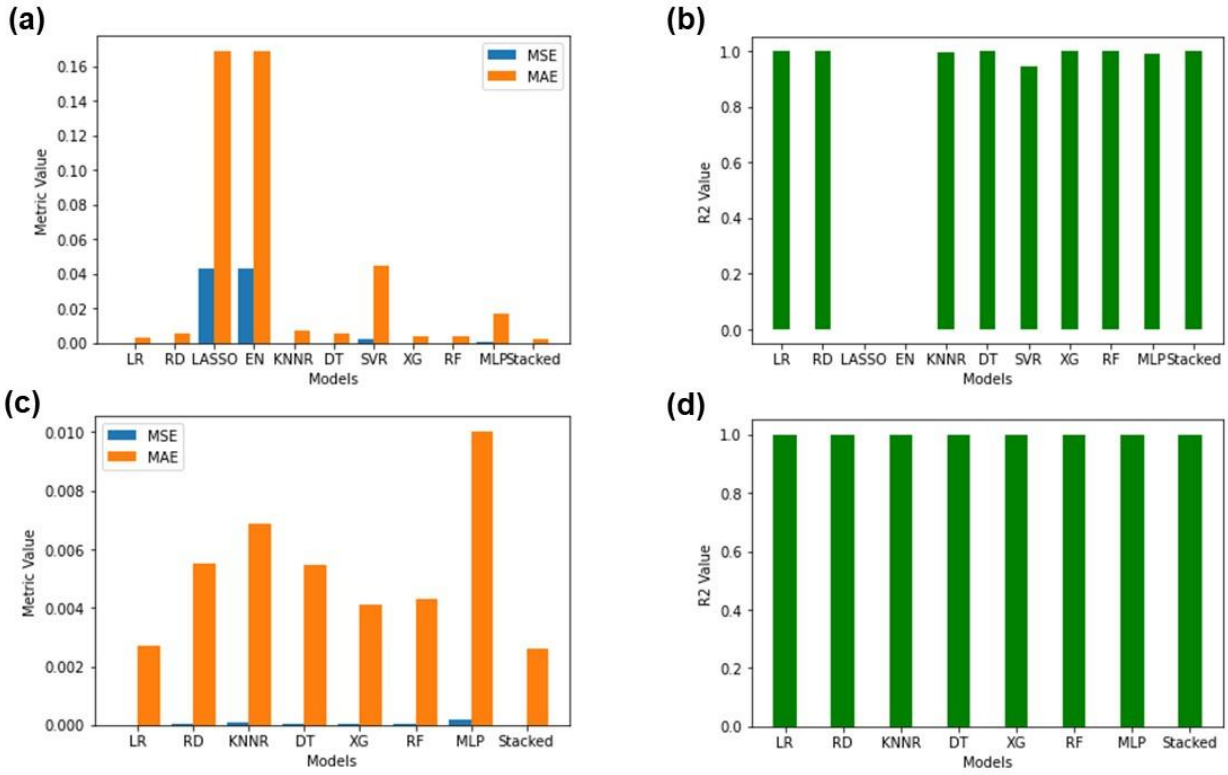


Figure 3.16 : Testing statistics (MSE, MAE and R^2) accuracy comparison of regression models trained on Central Rand Pump B a-b) Stacking regressor using all models and c-d) Stacking regressor using only the best performing models.

Table A5 containing the exact NMSE, MSE, MAE and R^2 values for each of the models found in Figure 3.15 and Figure 3.16 can be found in “Appendix I”.

3.1.4 Treated Water

The original sulphate values lied in the range of 2000-3000 mg/L, however, after geochemical modelling the sulphate values then lied in the range of 0.020-0.035 mg/L. The dramatic decrease in sulphate levels after geochemical modelling was due to unrealistic sulphate experimental values since these values did not balance with the cationic species recorded.

From Figure 3.17a and Figure 3.18a, the original dataset contained an enormous amount of outliers which heavily skewed the data. In particular, TDS and FE contained an entire range of outliers and this was unique when compared to Pump A and Pump B. After outlier removal (Figure 3.17b and Figure 3.18b) the distributions for each of the water quality parameters did significantly improve, however one unusual distribution was found for FE. Instead of a normal distribution, FE showed an almost smooth decrease with two missing bins which stood in stark

contrast to the FE distributions observed for Pump A and Pump B. With regards to all other water quality parameters, only single mean values were observed instead of the double hump behaviour as seen for Pump A and Pump B. These changes in distributions were indicative of the proper operating of the AMD treatment plant since all the water quality parameters were significantly altered after the treatment process which indicates a change in the overall water chemistry of the treatment process. Most of the metals such as FE and MN would have been removed as sludge during the treatment process and due to the addition of basic neutralizing agents the pH of the Treated Water was expected to increase, and indeed it did increase to 8.71 ± 0.179. The statistical values used to plot Figure 3.17 can be found in Table A6 in "Appendix I".

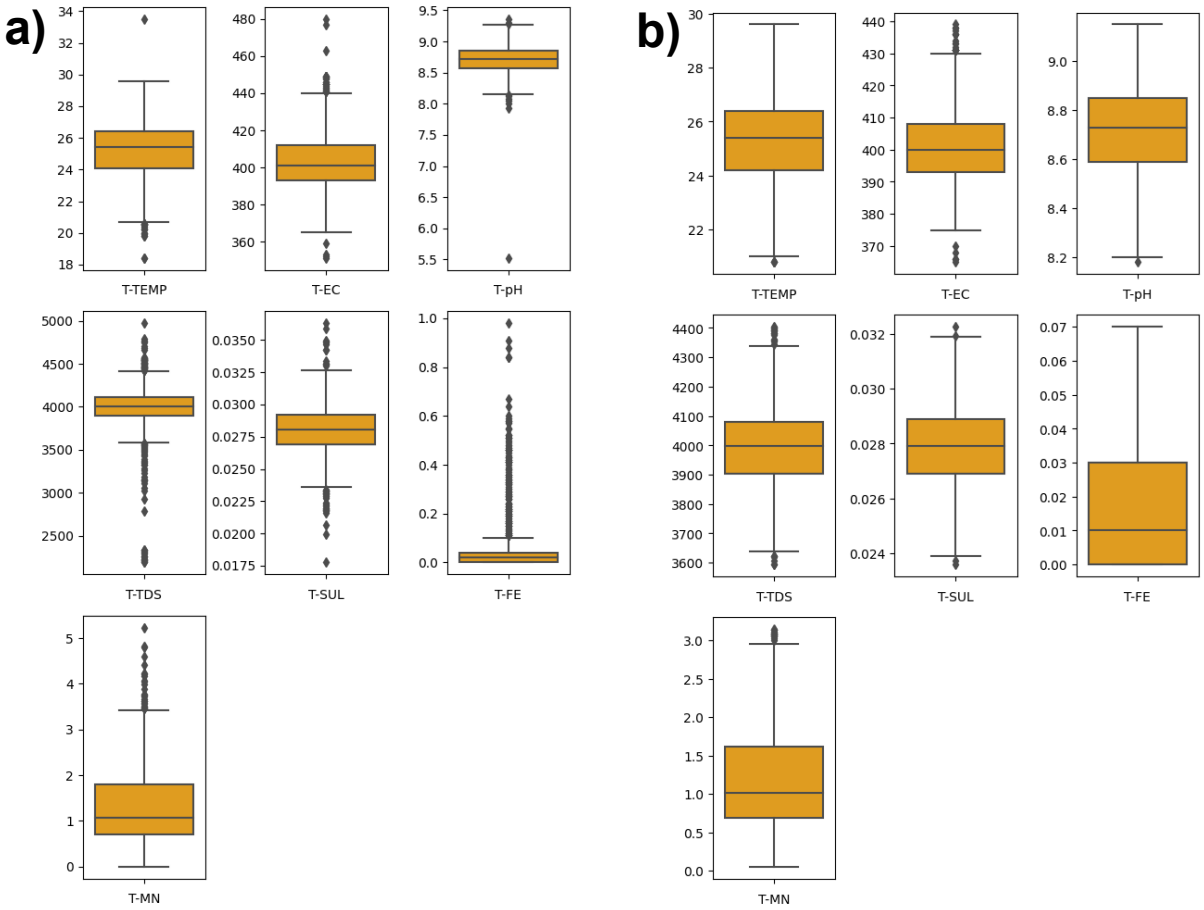


Figure 3.17 : Box and whisker diagrams of the Central Rand Treated Water indicating its statistical distribution a) Prior to outlier removal and b) After outlier removal.

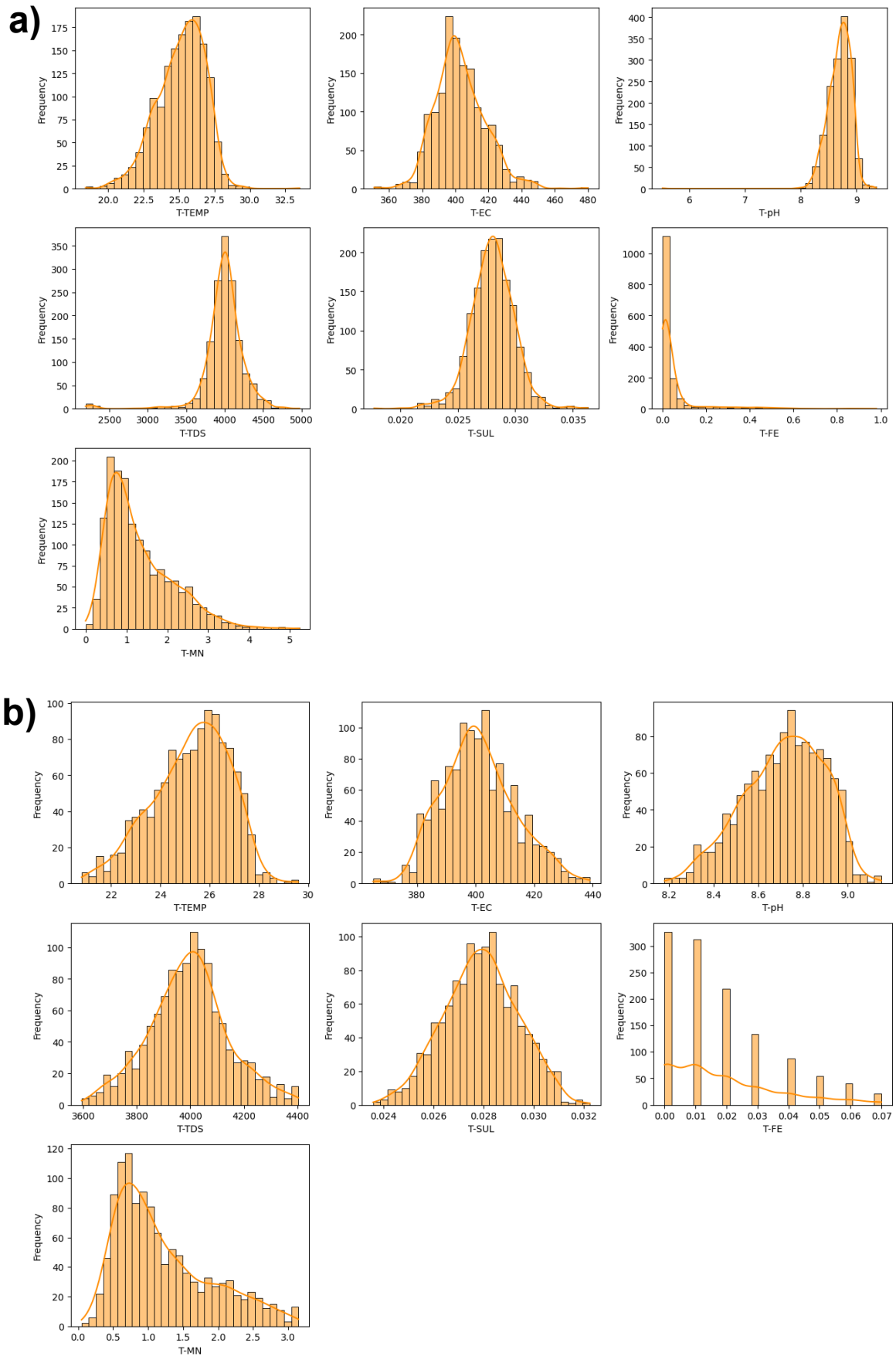


Figure 3.18 : Histogram and density plot traces indicating the statistical distribution of the Central Rand Treated Water a) Prior to outlier removal and b) After outlier removal.

From the scatter matrix (Figure 3.19) and the correlation matrix (Figure 3.20), there was almost no relationship between the water quality parameters (i.e., no inherent patterns were present). The presence of clusters around certain values instead of linear trends translated to a near zero correlation which was quite visually noticeable due to the presence of brighter colours in Figure 3.20 as opposed to lighter and darker colours as seen in the correlation matrices (Figure 3.6 and Figure 3.13) for Pump A and Pump B. The only strong correlation was found to be between pH and MN (-0.77) which indicated that as pH increased, the amount of manganese decreased. Noticeably, the distribution of FE was in distinct bins instead of a cluster or trendline, and this was incredibly unusual. This led to almost no correlation between FE and all other water quality parameters which was extremely strange due to FE showing such a high correlation to SUL in Pump A and Pump B.

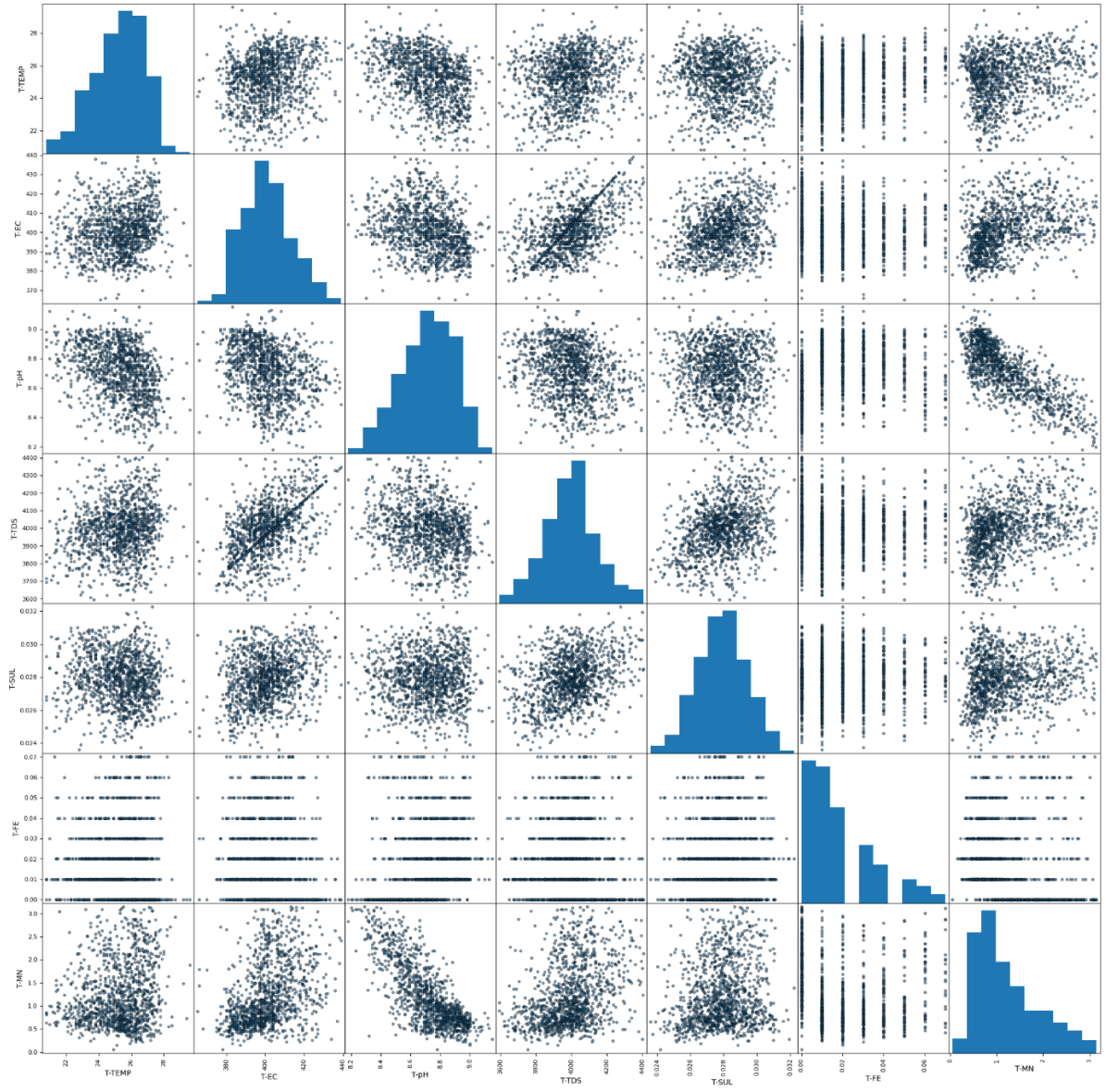


Figure 3.19 : Scatter matrix indicating the relationship between water quality parameters of Central Rand Treated Water.

Figure 3.20 : Correlation matrix indicating the correlation between water quality parameters for Central Rand Treated Water.

The bi-plot (Figure 3.21a-c) once again revealed two clusters which gave further evidence for the temperature dependence of the water quality parameter values. However, the loadings plot (Figure 3.21b) now indicated that T-FE was no longer related to T-SUL, instead T-MN, T-TDS, T-EC, T-TEMP and T-MN were strongly related to SUL which agreed with the trend observed in the scatter and correlation matrices. From the Scree plot (Figure 3.21d), only three components were needed to explain 74% of the variance in the data and these predictors were T-pH, T-TDS and T-MN. Due to the low correlation observed between variables, more predictors were needed in the Treated Water dataset (3 predictors) as compared to Pump A and Pump B (2 predictors).

Noticeably, Fe was not selected as a predictor for the Treated Water as opposed to that of Pump A and Pump B and this was in accordance with its unusual distribution and low correlation to SUL. The difference in predictors selected indicated that the chemistry of the Treated Water was significantly different to that of the untreated AMD (Pump A and Pump B). Due to basic neutralizing agents being added to reduce toxicity of the AMD, in the process any FE bound to SUL is liberated which dramatically alters the FE distribution and increases SUL concentrations. Therefore, pH is related to SUL since an increase in the amount of basic neutralizing agents added increases the pH which in turn liberates more SUL. MN may not have been effectively removed from the AMD during the treatment process and could have still been present in the treated effluent waters and manganese (II) sulphate which explains its correlation to SUL.

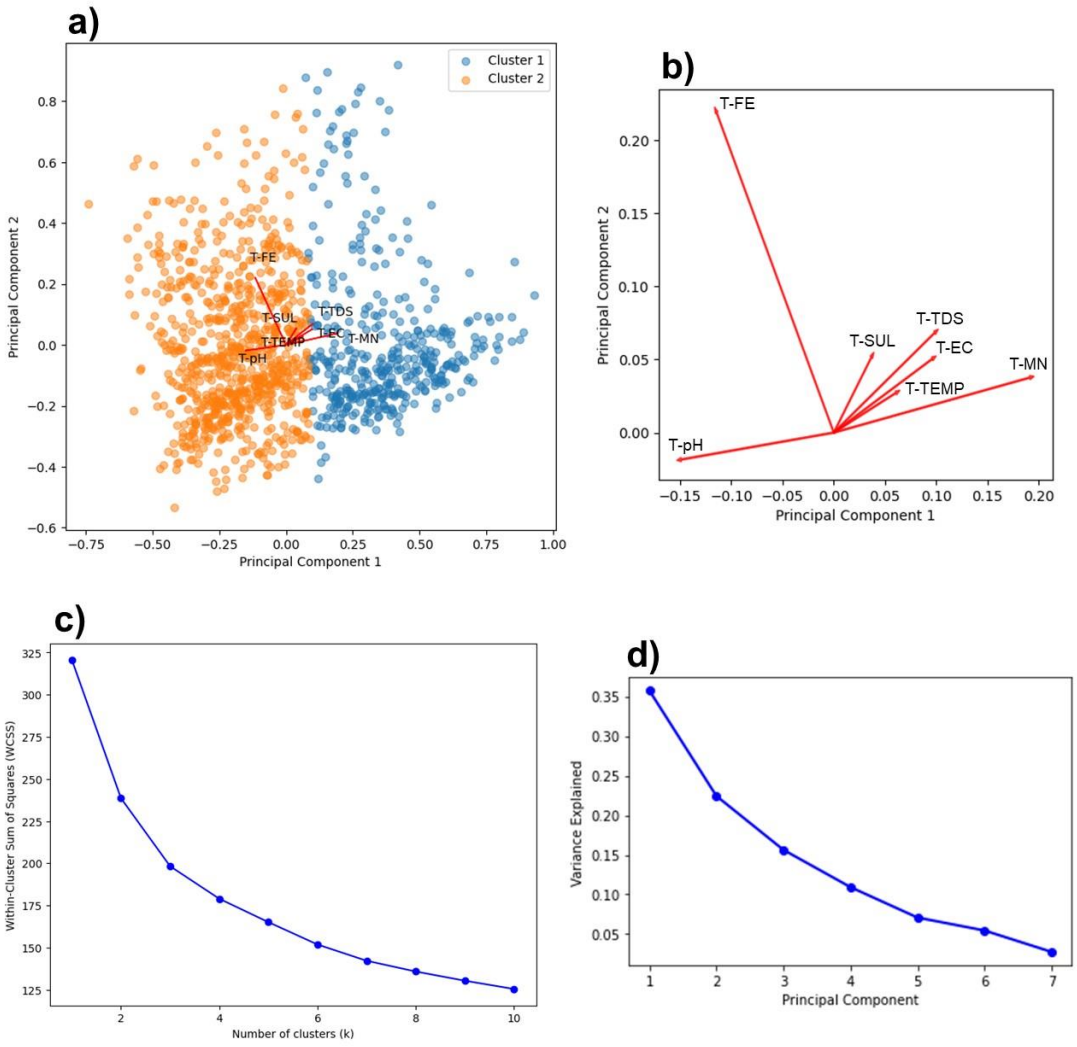


Figure 3.21 : Dimensionality reduction and feature extraction results obtained from PCA for the Treated Water, a) Bi-plot indicating the clustering of individual observations and its relation to the loadings plot, b) Expanded view of the loadings plot indicating the relationship between parameters, c) Elbow method plot indicating the optimal number of clusters and d) Scree plot indicating the amount of variance explained by each principal component.

As seen in Figure 3.22a and Table A7 (found in “Appendix I”), DT was the worst performing baseline model (NMSE:-0.048447) and the SVR once again was the best performing model (NMSE:-0.025181). DT are extremely susceptible to small variations in the data[166], due to the highly uncorrelated nature of the Treated Water data a lot of variations were present in it and hence, the DT performed poorly. SVR performed well due to its ability to handle outliers and was suitable to be used on the Treated Water dataset since it was small and only consisted of 1197 rows of data.

As seen in Figure 3.22b, the stacked ensemble model (which combined all the baseline models) did outperform all other models (NMSE:-0.024949) since it was able to combine the predictive accuracy of all the baseline models and learn from their mistakes.

As seen in Figure 3.22c, when only combining the best models into a stacking regressor, the stacking regressor actually performed worse (NMSE:-0.025005) than the stacking regressor that used all the models. A possible explanation for this is that stacking regressors not only combine predictive accuracies of the level 0 models, but they also learn from the level 0 model mistakes. Thus, when omitting the bad performing models from the stacking regressor, there was actually less knowledge to learn from since not many mistakes were made and so further improvements could not be made as to what not to do or what to improve on. This was quite an amazing discovery since it implied that including “failed” or “poorly performing” models in stacking regressors improved their overall performance for the Treated Water.

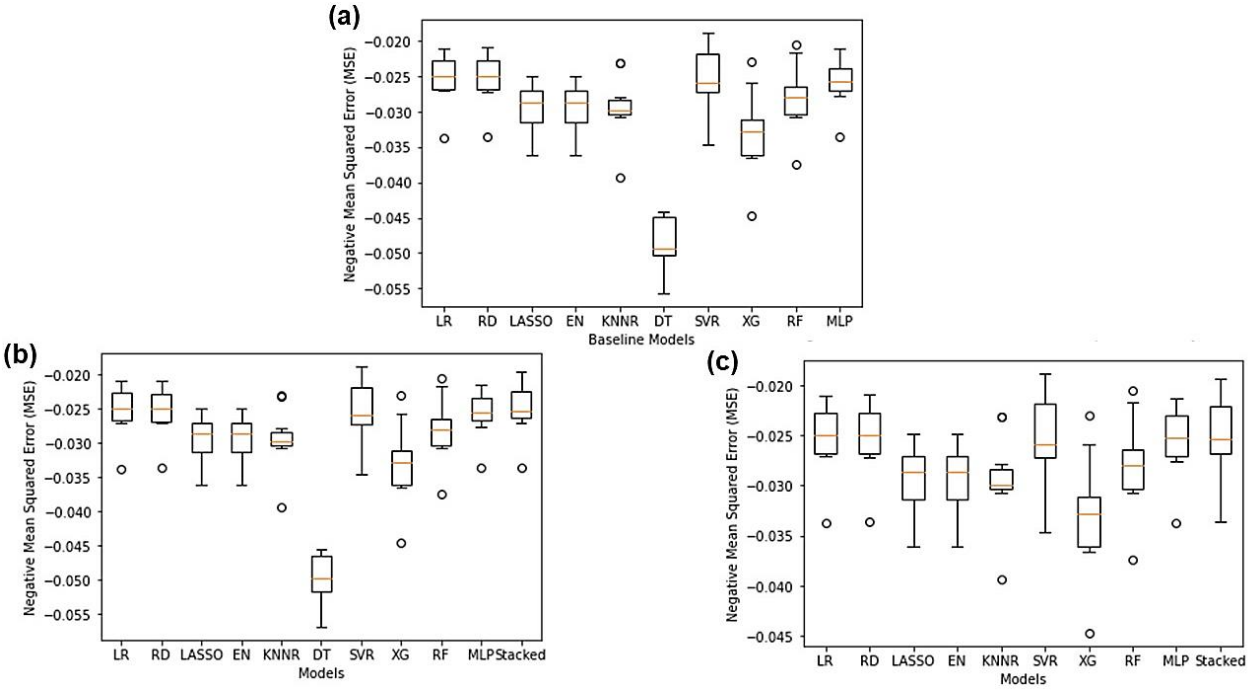


Figure 3.22 : Regression algorithm NMSE comparison for Central Rand Treated Water showing a) All individual baseline models, b) All models and a stacking regressor containing all the models and c) All good performing models and a stacking regressor containing all the best performing models.

As seen in Figure 3.23 the DT once again performed the worst during testing (MSE:0.055957, MAE:0.191006 and R^2:-0.967734) and the best performing model was the stacking regressor that combined all the models (MSE:0.024362, MAE:0.121571 and R^2:0.143315). However, the R^2 values obtained for the Treated Water were absolutely dismal since the highest value that could possibly be obtained by even the best performing model was 0.14 and a lot of models such as LASSO, EN, KNNR, DT, XG and RF all produced negative R^2 values. Any model that achieved a negative R^2 value indicated that this model performed worse at predicting a sulphate value than just replacing any missing value with the mean. Upon comparison to Pump A and Pump B this stood in stark contrast due to the poor R^2 values. Therefore, due to the poor R^2 values it was determined that any models trained on the Treated Water data were unsuitable to be used for transfer learning.

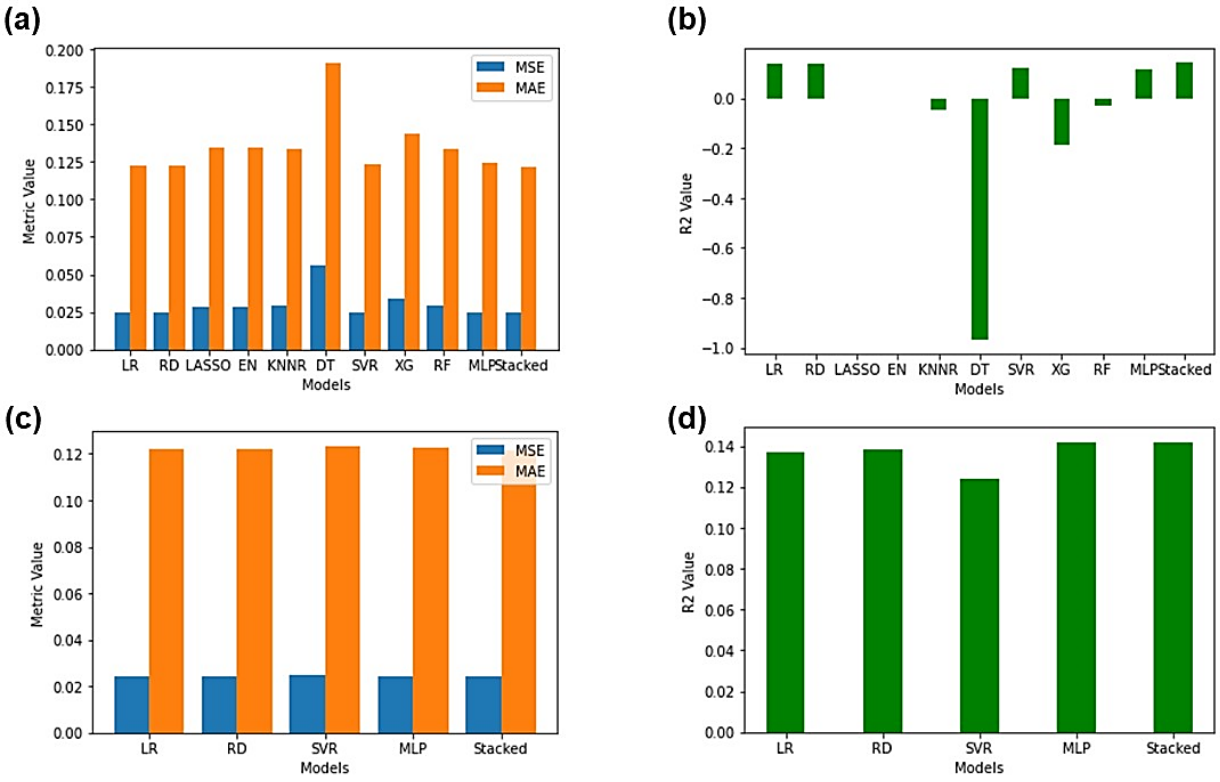


Figure 3.23 : Testing statistics (MSE, MAE and R^2) accuracy comparison of regression models trained on Central Rand Treated Water a-b) Stacking regressor using all models and c-d) Stacking regressor using only the best performing models.

Several attempts were made to improve the R^2 values of the models pertaining to the Treated Water data. These attempts included:

1. Removing FE from the Treated Water dataset due to its abnormal distribution that could have affected any statistical calculations.

2. Using a “StandardScaler” instead of a “MinMaxScaler” to transform the values to a standard Gaussian distribution such that all the means were 0 and the standard deviation was 1.
3. Using a “Normalizer” instead of a “MinMaxScaler” to transform the values such that each row in the dataset had a length of 1.
4. Removing FE and applying the “Normalizer”.

Unfortunately, none of the above-mentioned attempts were successful since the highest R^2 value for any model ever obtained was 0.15 and this came at a cost of a much higher MSE and MAE. However, and interesting discovery was made as seen in Table 3.1, which indicated that the most important features as identified by PCA and RFE were highly dependent on the type of data transform used. The Variance explained (%) in Table 3.1 further highlighted the importance of picking the right transform for the data since in some cases (especially for the “Normalizer”) 98% of the variance could be explained using a single feature (predictor) which is highly unlikely given the distribution and correlation of the Treated Water data.

Table 3.1 : Dependence of important features on type of data transform applied.

Type of transform applied	Most important features	Variance explained (%)
Removed T-FE	T-TEMP, T-EC and T-TDS	79
StandardScaler	T-EC, T-pH, T-TDS and T-MN	81
Normalizer	T-FE	98
Removed T-FE and Normalizer	T-pH	98

Since no transform applied could successfully improve the R^2 values, it was determined that the original Treated Water data contained inherent inaccuracies or simply, contained no inherent patterns from which machine learning algorithms could identify and learn from in order to predict sulphate values. Thus, the models trained on the Treated Water data could not be used for any transfer learning.

The lack of any inherent patterns in the Treated Water dataset were most likely to come from the AMD treatment process itself which significantly altered the chemistry of the AMD water. The treatment plant may not have been operating optimally on some days or may have made significant errors when testing the Treated Water which could have led to inaccuracies being present in the data. Another reason for the lack of inherent patterns could be that treated AMD may just be so chemically altered due to the various additions of reagents and processes that it goes through that there exists no inherent patterns between water quality parameters.

In future, several sampling spots of the Treated Water should be measured overtime to remove any errors associated with a particular sampling spot in order to generate a more accurate and trustworthy dataset.

3.1.5 Overall

Models trained on Central Rand Pump B's data were the best performing models, the stacking regressor in particular performed the best and was selected to be used as the model to conduct transfer learning with. Pump A's models performed significantly well but were outperformed by Pump B's models. The Treated Water models performed poorly due to inherent inaccuracies and a lack of pattern within the Treated Water dataset.

The number of predictors needed to achieve at least 74% of explained variance was at most three which indicated that the number of predictors (water quality parameters) was not important, but rather the quality and type of predictors was important and this was consistent with what was reported by Abyaneh[167].

3.2 EVALUATION OF EAST RAND SULPHATE PREDICTION USING TRANSFER LEARNING

3.2.1 Data visualizations and statistics

From Figure 3.24a and Figure 3.254a, similarly to what was observed for the Central Rand dataset the distribution of data in the East Rand contained a lot of outliers which heavily skewed its distributions. This indicated that the presence of outliers was a general AMD treatment plant feature and did not only come from one plant. This was reflective of the nature of water treatment plant and inherent variability of traditional analytical chemistry procedures used. Water quality parameters are also heavily dependent on weather, seasons, climate, and geology which significantly varies during each month and leads to complex non-linear relationships.[168] Therefore, the presence of outliers pertaining to water quality parameters did give evidence to its intrinsic variable nature. After outlier removal and geochemical modelling (Figure 3.24b and Figure 3.25b) distributions of the water quality parameters were normal and showed similar relations to each other as observed for Central Rand Pump A and Pump B. The same double hump behaviour indicating a bimodal distribution were observed and the exact mean values and associated statistics can be found in Table A8 in “Appendix J”. Interestingly, the SUL values did show a very similar density plot shape to that of TEMP and FE which was indicative of the East Rand plant SUL values being very similar to those of the Central Rand plant. This was a good indication that the Pump B model would be able to adequately perform when used for TL since the distributions of the predictors in the East Rand and Central Rand datasets were similar.

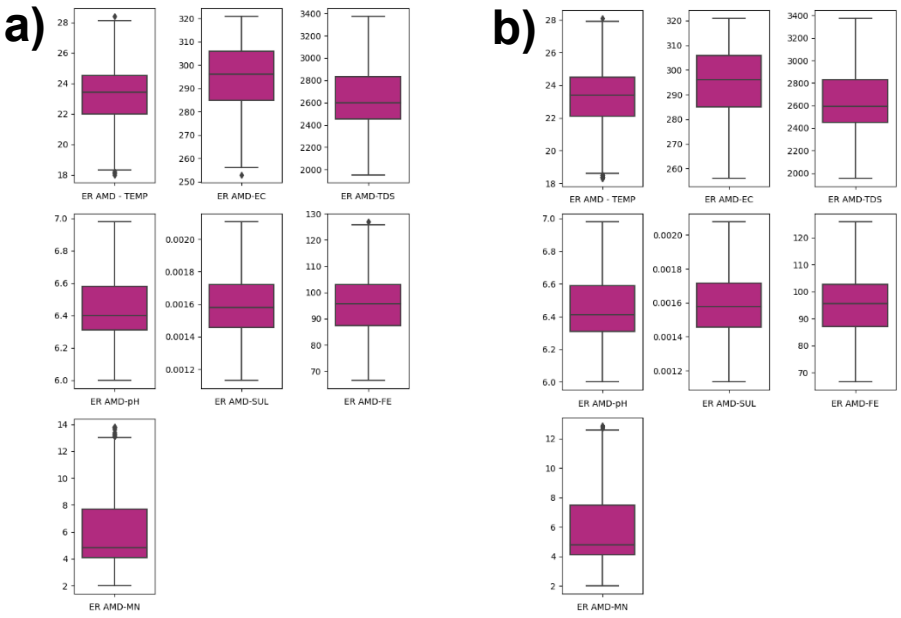


Figure 3.24 : Box and whisker diagrams of the East Rand AMD indicating its statistical distribution a) Prior to outlier removal and b) After outlier removal.

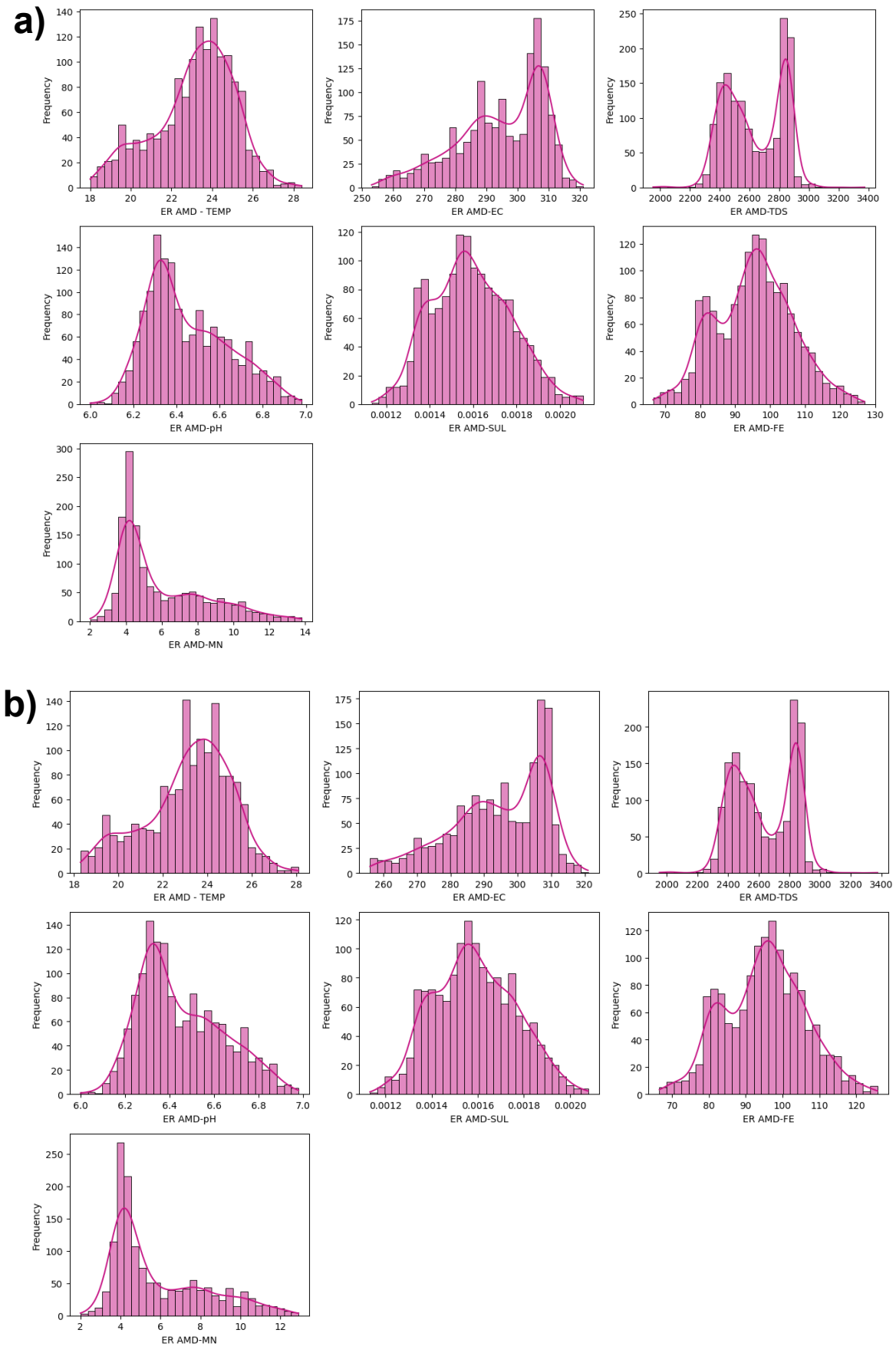


Figure 3.25 : Histogram and density plot traces indicating the statistical distribution of the East Rand AMD a) Prior to outlier removal and b) After outlier removal.

From Figure 3.26 and Figure 3.27 SUL did show an almost perfect positive linear relationship (0.98) to FE which once again provided evidence for the presence of high amounts of dissolved pyrites. SUL did show a slight negative correlation (-0.37) to TEMP. These correlations provided evidence that the Pump B stacking regressor was suitable to be used for TL since it had similar correlations between the predictors and target. The correlation matrix did reveal that the East Rand AMD did have a lot of structure and strong correlations within the dataset since the heatmap colours tended to the lighter (more positive) and darker (more negative) sides.

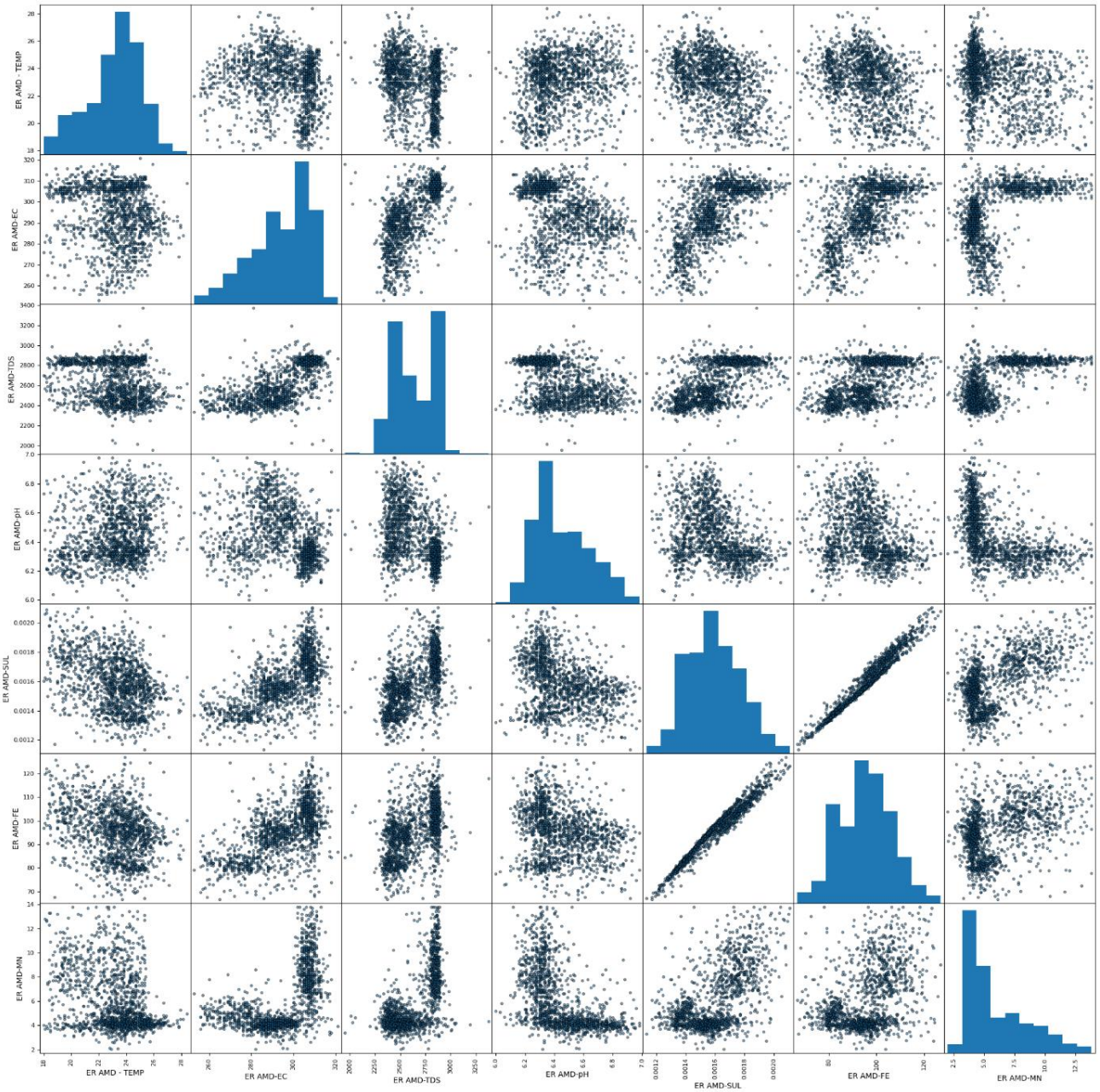


Figure 3.26 : Scatter matrix for East Rand AMD after outlier removal and geochemical modelling.

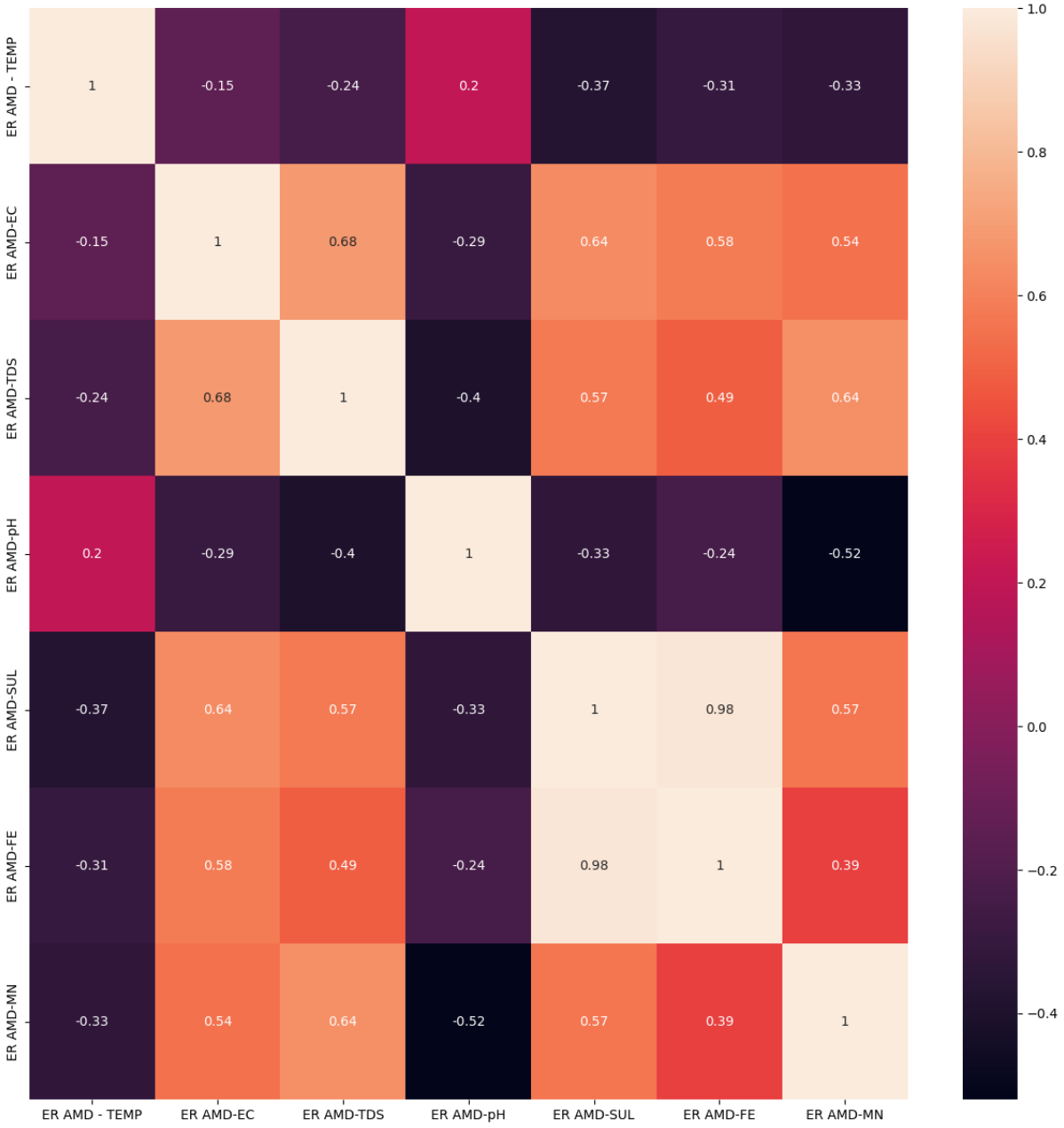


Figure 3.27 : Correlation matrix for East Rand AMD after outlier removal and geochemical modelling.

From the 3D PCA bi-plot (Figure 3.28a), two distinct clusters were observed which was consistent with the bimodal distribution observed in the density plot traces (Figure 3.25) for the ER AMD. The PCA loadings plot (Figure 3.28b) revealed that TEMP and FE was strongly related to SUL and this was consistent with what was observed in the ER correlation matrix (Figure 3.27). PCA revealed that the ER AMD dataset bore an extremely high similarity to that of Central Rand Pump B, and this indicated that TL would be possible between the aforementioned domains.

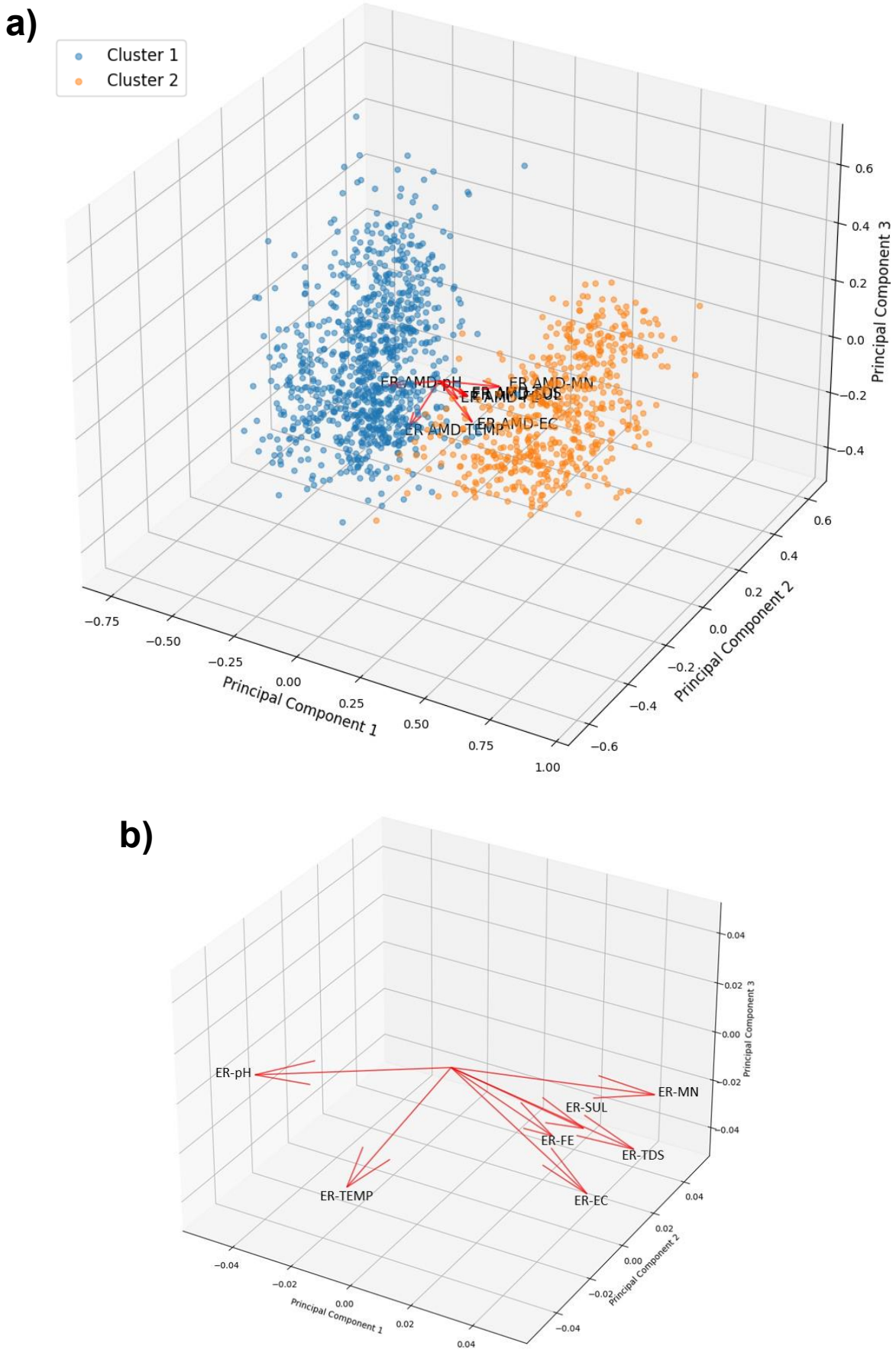


Figure 3.28 : a) PCA bi-plot of the ER AMD indicating the presence of two clusters, b) Expanded view of the ER AMD loadings plot indicating the correlation between variables.

3.2.2 Transfer Learning

Transfer Learning was successfully conducted using the stacking regressor (Figure 3.29) trained on Central Rand Pump B's data and the predicted sulphate values closely resembled those of the true sulphate values since the testing statistics of MSE:0.00124, MAE:0.0290 and R^2:0.963 indicated a high level of precision and accuracy. The low MSE and MAE values coupled with the high R^2 value indicated that the Pump B model was the correct model to use for TL learning and achieved the desired outcome.

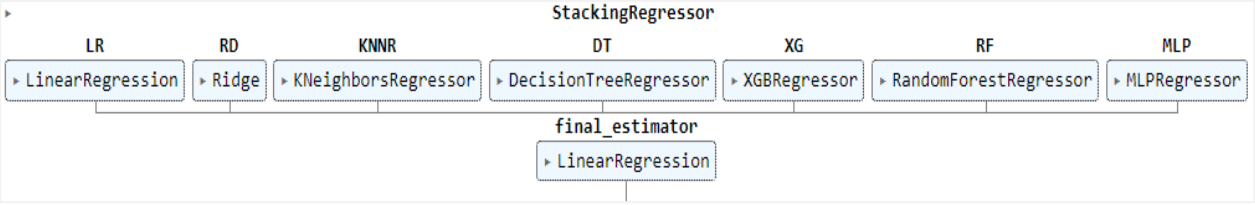


Figure 3.29 : Central Rand Pump B's stacking regressor used for TL to East Rand AMD feed.

From Figure 3.30a, it was clear to see that the predicted values did lie extremely close to the true values. Lower sulphate concentrations were predicted much more accurately and lied closer to the true values when compared to higher sulphate concentrations. Higher sulphate concentrations were a lot more dispersed and contained a higher amount of variability, therefore, these values could partially be predicted using TL.

From Figure 3.30b and Table (found in "Appendix J"), the mean values of the true sulphate levels (0.0159 ± 0.000180 mg/L) was close to that of the predicted sulphate levels (0.00160 ± 0.000176 mg/L) and the inter-quartile range was also similar.

From Figure 3.30c and Figure 3.30d, it was found that the discrepancy between the true values and the predicted values lied specifically in two regions namely, 0.0015 and 0.0017 mg/L. These two regions lay on the depressed areas of the density plot which was off from the normal distribution of the plot and, hence, was not accurately predicted due to its statistical variation. The predicted sulphate values closely followed the shape of the given iron values which was expected due to its extremely high correlation with sulphur.

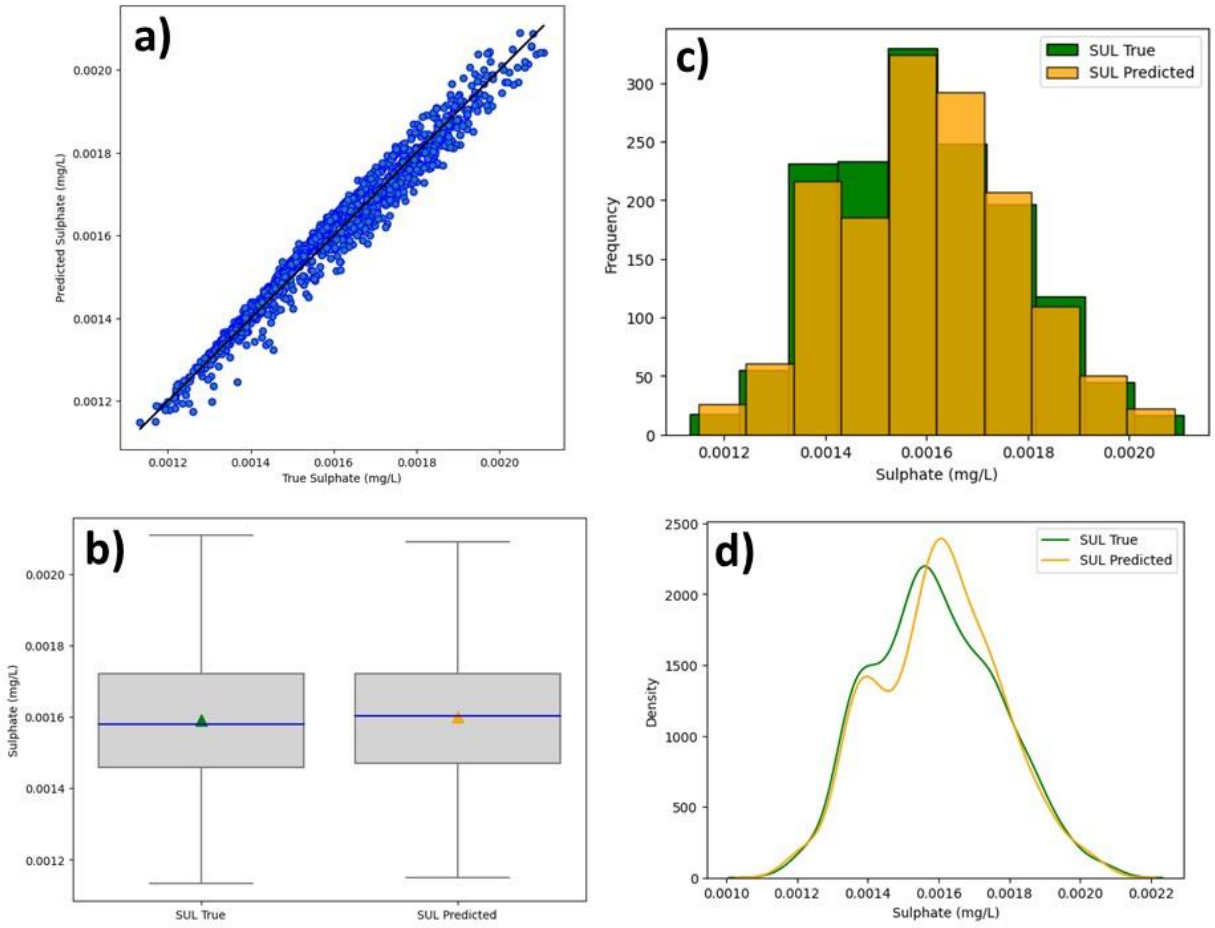


Figure 3.30: a) Scatter plot indicating the predictive accuracy of Central Rand's stacking regressor after TL onto the East Rand AMD dataset (black line is y=x which indicates a perfect prediction), Comparison of the b) Box and whisker diagrams, c) histograms and d) density plots between the true and predicted sulphate values (mg/L).

Overall, TL was successful in predicting the sulphate levels in the AMD feed from the East Rand dataset and showed that the stacking regressor model that was trained on only the best performing models from Central Rand Pump B was suitable to be used for TL.

3.3 IMPUTATION OF MISSING SULPHATE LEVELS IN THE WEST RAND DATASET USING TRANSFER LEARNING

The power of AI and ML was noticed when conducting TL onto the West Rand dataset since only two parameters (viz., TEMP and FE) were needed to predict SUL. TEMP can be easily and cheaply measured using a thermometer and FE can be measured along with other elements simultaneously using Inductively Coupled Atomic Absorption Spectroscopy (ICP-AAS).[169] Despite no true sulphate values being measured over the years of 2015-2020, TL was still deemed to be useful in predicting sulphate levels in order to provide a baseline of historical sulphate levels in order to gain a broad understanding of sulphate distribution in the West Rand. Whilst no accuracy metrics could be calculated in the West Rand due to a lack of true values, the high accuracy achieved in the East Rand dataset was a good indication that the trained stacking regressor (Central Rand Pump B) could accurately predict sulphate levels in another AMD treatment plant.

3.3.1 TL onto KGR inflow (Untreated AMD)

After outlier removal, the predictors (TEMP and FE) were able to successfully predict the missing sulphate values over the years of 2015-2020 and the sulphate levels showed a strong correlation to the iron values used to predict them as seen in Figure 3.31. The sulphate levels were predicted to lie in the range of 0.023-0.0982 mg/L with a mean value of 0.393 ± 0.196 mg/L (further statistics can be found in Table A10 in “Appendix K”). Four main peak values were found to have existed for the predicted sulphate values (Figure 3.31h) which suggested that the sulphate levels did not follow a normal distribution but rather varied quite a lot. This could be due to several factors such as seasons, weather or inherent inaccuracies present in traditional chemistry approaches used to analyse the water since these all affect the FE distribution which in turn affects the SUL distribution.

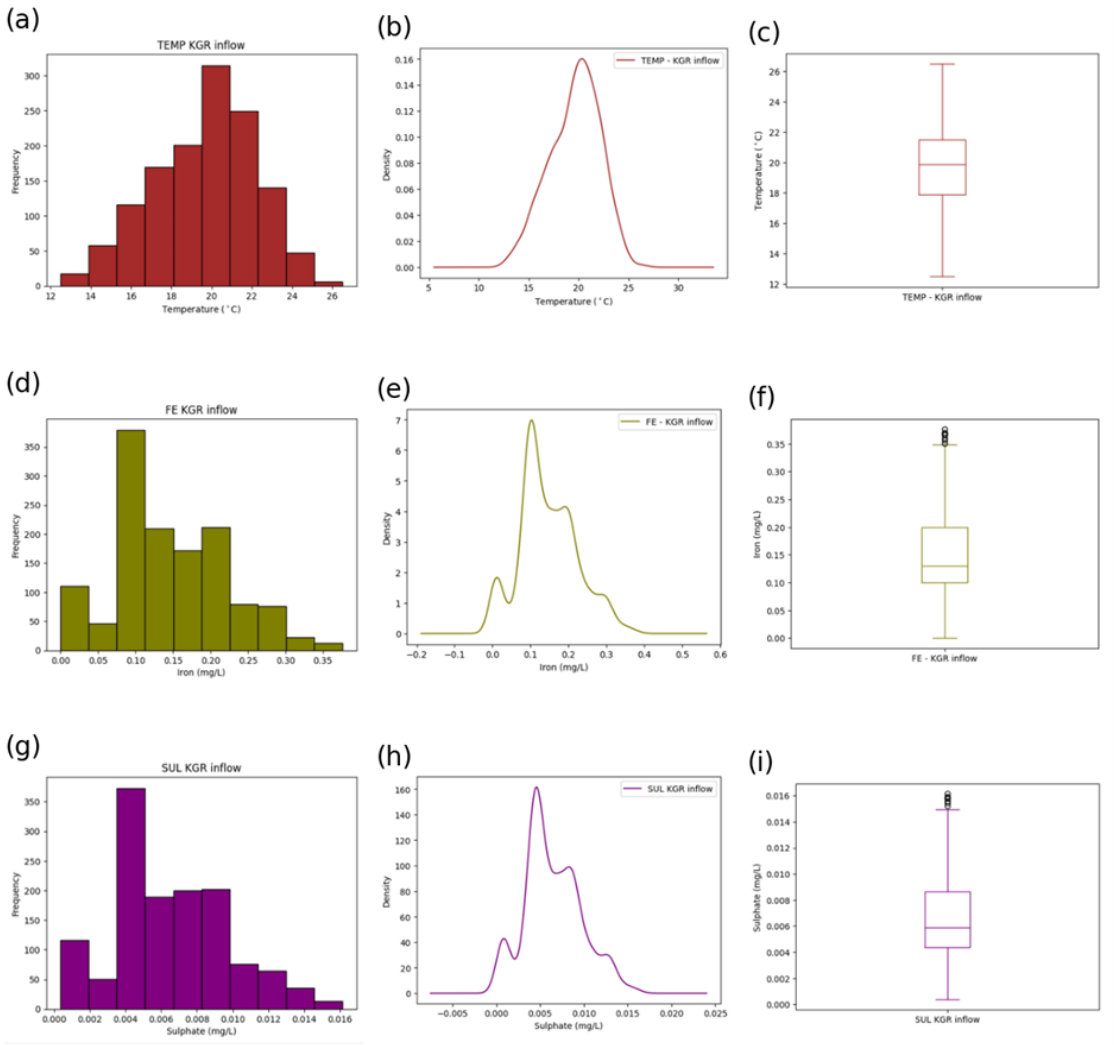


Figure 3.31 : Statistical distributions of predictors a-c) temperature, d-f) iron after outlier removal and the resulting g-i) sulphate levels predicted (mg/L) after TL to the West Rand "KGR inflow" dataset.

Whilst no true sulphate values were available for comparison, since the missing values were imputed and were shown to have an almost identical trend to the iron values, which was also observed for the Central Rand and East Rand datasets, it was assumed that since the Central Rand and East Rand datasets obtained a high level of accuracy using the same stacking regressor, the West Rand predicted sulphate values also had a high level of accuracy.

3.3.2 TL onto Treated flow into KGR (Treated AMD)

As observed in Figure 3.32, the predicted sulphate levels for the Treated Water also followed the same distribution as that of the iron levels. The sulphate levels lied in the range of 0.000343 – 0.0162 mg/L with a mean value of 0.00611 ± 0.00338 mg/L which indicated a high degree of variability within the predicted values. Three distinct peaks were observed in the sulphate distribution (Figure 3.32h) which indicated that the sulphate values were not normally distributed and this could have led to the high degree of variability. Further statistical data pertaining to Figure 3.32 can be found in Table A11 under "Appendix K".

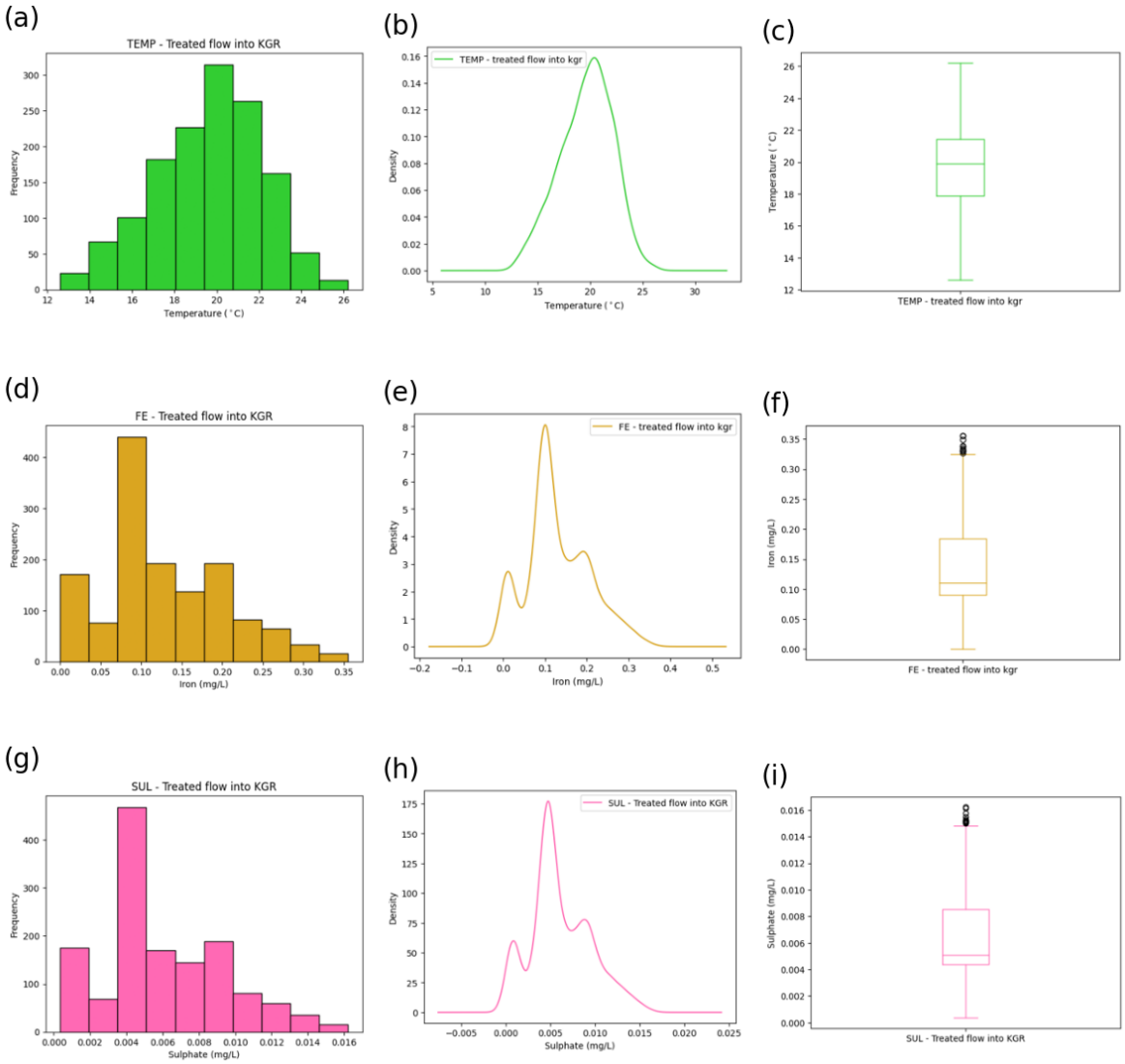


Figure 3.32 : Statistical distributions of predictors a-c) temperature, d-f) iron after outlier removal and the resulting g-i) sulphate levels predicted (mg/L) after TL to the West Rand "Treated flow into KGR" dataset.

Overall, TL was successfully used to impute the missing sulphate levels in the West Rand AMD treatment plant and this allowed for a baseline comparison to be made between all three AMD treatment plants. This would be beneficial in understanding the inherent variabilities present in each of the datasets and how the tree treatment plant sulphate levels lied in comparison to each other.

3.4 SULPHATE LEVEL COMPRARISON IN ALL THREE AMD TREATMENT PLANTS

Upon comparing the predicted and true sulphate levels between all three treatment plants, as observed in Figure 3.33, it was found that all three treatment plants significantly differed from each other since the sulphate levels for each plant fell in different ranges. Noticeably, for the Central Rand treatment plant, Pump A and Pump B lied close to each other but the Treated Water had a much higher sulphate value range. The distributions for The Central Rand plant were all close to normal.

For the East Rand treatment plant, the sulphate levels were more precise as seen in the presence of sharp peaks instead of a wide distribution. TL was also observed to be accurate in predicting values since the predicted and true density plots for the East Rand treatment plant lied directly on top of each other. The power of TL and AI was also noted, since the Central Rand plant had such a drastic different distribution when compared to the East Rand plant, but the Central Rand Pump B stacking regressor was able to accurately predict the sulphate levels in the East Rand plant despite the differing distributions.

The West Rand AMD and Treated Water lay incredibly close to each other due to them both having similar iron and, hence, sulphate levels. The West Rand data also showed the highest degree of variance in the distribution of the sulphate levels which indicated that the West Rand data itself had inherent variability and extremely low precision.

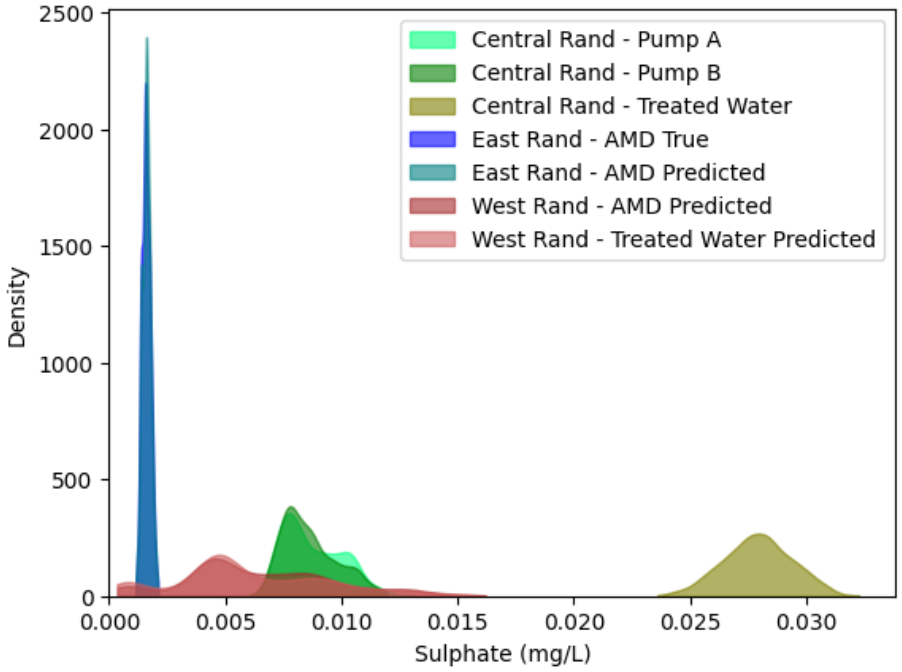


Figure 3.33 : Comparison of sulphate levels (true and predicted) between the Central Rand, East Rand and West Rand AMD treatment plants.

3.5 “Sulfind”: A GUI FOR AMD SULPHATE LEVEL PREDICTION

3.5.1 Jupyter Notebook based GUI

In order to make the accurately trained SE-ML regressor easily accessible to those with no or low knowledge coding skills, as well as to those with limited knowledge regarding ML techniques, a GUI was created. This GUI, as depicted in Figure 3.34, was named “Sulfind” and took six inputs which were the unnormalized minimum, maximum and actual values of both temperature (°C) and iron (mg/L) levels. The minimum and maximum values were used to normalize the value of interest which were fed as predictors to the trained Central Rand Pump B SE-ML regressor that combined only the best values.

This GUI was housed in a specific Jupyter Notebook and when clicking “Run” the GUI code was executed and the GUI popped up in another window. Upon clicking the “Predict Sulphate (mg/L)” button, the predicted sulphate level appeared under it which made this GUI extremely accessible and easy to use to predict sulphate levels (mg/L) in AMD.

The accuracy of ten sulphate values returned by the GUI were compared to those obtained through traditional predictive coding, and it was found that there was no difference in the values obtained (to six decimal places). This indicated that developing the GUI had no effect on the predictive performance of the regression model and its accuracy was retained. The tested values were selected to span over an entire range of input temperatures and concentrations to fairly test the performance of the GUI.

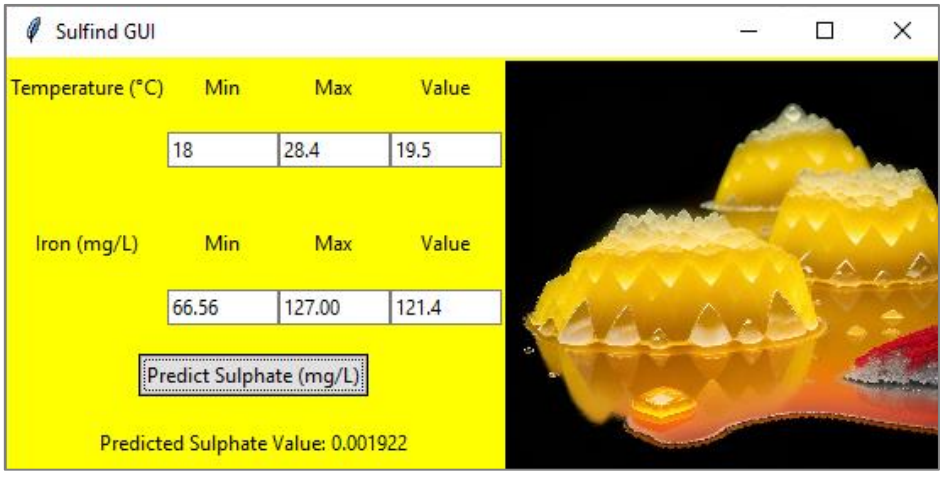


Figure 3.34 : Screenshot of "Sulfind", the AMD sulphate level predictor GUI. The image on the right was created using “WOMBO dream ai”[170] by giving it a prompt of “yellow crystals growing out of red toxic water” and an input image[41] of mineralogical sulphur as well.

3.5.2 Web based GUI

To avoid people becoming confused by the Jupyter Notebook interface and to further contribute to the “no code” nature of the GUI, a simplified web version of the GUI was created and is available for use at the following link https://sulfind-webapp-gui-c4ce20a6aee6.herokuapp.com/. This link removed any and all programming and ML knowledge needed to use the accurately trained SE-ML regressor and is easily accessible through a browser.

ChatGPT-3.5 proved to be incredibly useful for generating the web GUI code using the Python package “Flask” and also provided clear and easy to follow instructions on how to deploy the web GUI using Heroku. Screenshots of the web version of “Sulfind” can be found in Figure 3.35.

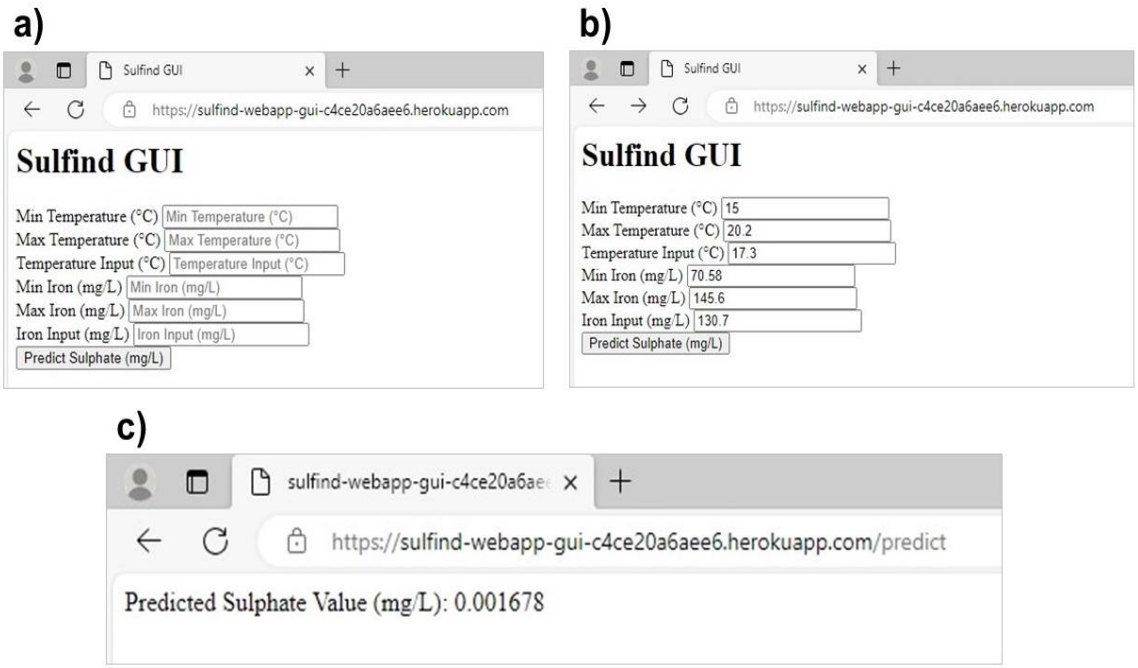


Figure 3.35: Screenshots of the web version of the "Sulfind" GUI deployed using Heroku, a) Landing page of the GUI without any input parameters entered, b) Desired input parameters entered into the GUI and c)The predicted sulphate value based on the given input parameters obtained after clicking the “Predict Sulphate (mg/L)” button.

The GUI has shown that it is possible to augment analytical data in instances where limited parameters of water chemistry are possible to analyse and using these to deduce or approximate those that may not be available or possible to analyse. While “Sulfind” presents a very simplified case, extensions can be made to handle more complex environments by building onto the library used. This presents an important benchmark on which to develop other GUIs that are applicable to other parameters of interest e.g., phosphates, nutrients and organic compounds among others using ChatGPT.

3.5.3 ChatGPT prompting “Tips and Tricks”

Several lessons were leant regarding how to prompt ChatGPT to create a working GUI code specified to one’s linking, as such, some “Prompting Tips and Tricks” are given below:

Initial Prompt:

- Start off by giving as much detail as possible in the initial prompt, such as what you already have (e.g. a trained ML model), the coding software you use (e.g. Google Colab or Jupyter Notebooks), the language you want to use (e.g. Python) and what you hope to create (e.g. a GUI).
- Be specific on what type of input you want your GUI to take and what type of output it should give back e.g., where it should take single, multiple or even an Excel file as input.
- Start off by creating a simple GUI. Colours, layout and images can be adjusted after a functioning GUI is created. Keep it simple at the start, then build on it. This would mean starting off with a library such as Tkinter and then transiting to more advanced ones such as PyQt and Kivy.

An example of a good initial prompt would be, “I want to create a GUI named "Sulfind" that uses a regression model that I have trained to predict sulphate levels. There are 6 inputs namely the minimum, maximum and actual values of “Temperature” and “Iron”. The minimum and maximum values are used to normalize the actual values. The sulphate level is determined by passing the normalized actual values to the regression model and then making a prediction . Please give me the Python code that is required to create this GUI in the Jupyter Notebook environment using the library named Tkinter. After I run the code I want the GUI to pop up so that I can vary the input values, and by clicking a single button I want to see the final predicted "Sulphate" level.”

Prompting in general:

- Prompt for clarification regarding steps that are new to you or that you need more explanation on. This holds true when using ChatGPT to learn how to use new software as well.

e.g. “From the steps that you have given me above pertaining to creating the GUI, how do I do step 4? How do I save my model as joblib.dump ? Show me additional steps”

- Conversations should be limited to one main idea that gradually evolves throughout prompting instead of introducing brand new ideas after each prompt.

Debugging:

- For code debugging, it is useful to not only provide the error message that one receives but to also provide ChatGPT with the code one is working on to receive a more tailored response that minimizes the introduction of more human errors that could occur by replacing bits of code with generic responses.

e.g. “I’m running this code in a Jupyter Notebook:

[provide actual code]

I get this error when trying to install xgboost:

---------- AttributeError Traceback (most recent call last) Cell In[3], line 1 ----> 1 get_ipython().system('pip install xgboost scikit-learn') File /lib/python3.11/site□packages/IPython/core/interactiveshell.py:2590, in InteractiveShell.system_piped(self, cmd) 2585 raise OSError("Background processes not supported.") 2587 # we explicitly do NOT return the subprocess status code, because 2588 # a non-None value would trigger :func:`sys.displayhook` calls. 2589 # Instead, we store the exit_code in user_ns. -> 2590 self.user_ns['_exit_code'] = system(self.var_expand(cmd, depth=1)) File /lib/python3.11/site□packages/IPython/utils/_process_posix.py:129, in ProcessHandler.system(self, cmd) 125 enc = DEFAULT_ENCODING 127 # Patterns to match on the output, for pexpect. We read input and 128 # allow either a short timeout or EOF --> 129 patterns = [pexpect.TIMEOUT, pexpect.EOF] 130 # the index of the EOF pattern in the list. 131 # even though we know it's 1, this call means we don't have to worry if 132 # we change the above list, and forget to change this value: 133 EOF_index = patterns.index(pexpect.EOF) AttributeError: module 'pexpect' has no attribute 'TIMEOUT'

Please find the error in the code and provide me with the new updated code that fixes this error.”

Testing the GUI:

- Always check that the GUI is conducting the correct mathematics by verifying the GUI output with the true output. An easy way to do this is by passing through a set of simple inputs where you can calculate the true output yourself. You can also use a validation dataset that has true or known values of what you want to predict. Conceptually, this is similar to analyzing standard or certified reference materials in an instrument to assess its accuracy.

- Be patient, it took five conversations that span over 70 pages to create the GUI mentioned in this study. ChatGPT did not get it right on the first attempt, but through a human error message negative feedback loop success was achieved.

Deploying the GUI:

- In order to deploy GUIs to the web, ChatGPT should be prompted to create the web application code using the Python package “Flask” (not “Tkinter”).

  e.g. “I want to create a GUI that I can deploy over the web instead of inside the Jupyter Notebook environment. Please rewrite the following code to use the Python library Flask instead of Tkinter:

  [provide actual code]”

- Clear instructions on how to deploy the GUI to the web using the cloud-based services platform, “Heroku”, can also be obtained by prompting ChatGPT. If necessary, more information pertaining to each step that ChatGPT suggests can be acquired by asking ChatGPT to provide more details for each specific step.

  e.g. “From Step 2 that you have given above, please provide me with additional information on how to create the requirements.txt file.”

- If ChatGPT seems to be overcomplicating a solution or starts to send the prompter to a significant amount or sites, it is best to end the current conversation and start a new one. ChatGPT does admit to getting things wrong and sometimes provides a much more complex solution to what is sometimes an easy fix. In situations like this, it is best to research solutions the “old-fashioned” way (i.e., using Google) and then prompt ChatGPT to solve the problem using the “old-fashioned” solution as a scaffolding.

  e.g. “I want to create a GUI that I can deploy over the web instead of inside the Jupyter Notebook environment. I have researched how to do this and want to use the Python library Flask and not Dash (which is what you have been using so far). Please rewrite the code to use Flask instead.”

4. CONCLUSIONS AND FUTURE WORK

This chapter presents the implications of the findings, general conclusions and recommendations that give insights into what future work and studies could focus on.

4.1 Conclusions

Access to clean water is still one of the most pressing challenges of the century with unpolluted natural water becoming an even more scarce resource. The implementation of sustainable wastewater treatment technologies would ensure that a reliable and efficient circular economy is created whereby, toxic water produced from one industry can be treated to recover not only clean water but also valuable by-products.

South Africa relies heavily on mining in order to stimulate economic growth, as such a lot of mining related pollution is formed. One such major concern is AMD which leaches out of mine tailing dumps into the water table below, thereby, contaminating the water. Therefore, AMD treatment processes have been implemented to reduce the toxicity of AMD and one such process utilizes the addition of basic neutralizing agents in order to react with acidic and toxic species to form high density sludge which is then removed and disposed of. However, high sulphate concentrations still remain in the discharged treated effluent water streams which can further undergo a series of chemical reactions to form octathiocane, a commercial product, which could be recovered.

The challenge lies in traditional chemistry approaches used to determine the sulphate concentrations such as gravimetry, IC, ICP-AES, spectrophotometry, gravimetry and titrimetry. These methods are extremely time-consuming, expensive, inefficient and make use of hazardous chemicals which have led to recorded sulphate values being extremely high and inaccurate of the true values.

Therefore, this study sought to apply modern, efficient and cheap computational methods such as geochemical modelling, ML regression algorithms, stacking ensemble ML and transfer learning in order to predict sulphate levels using historical data. This removed the need the need to conduct costly and exhaustive traditional analytical experiments and analysis procedures. The aim was to provide data (accurate sulphate concentrations) as a pre-cursor for future AMD remediation studies e.g., water cleanup and recovery of useful minerals such as gypsum, octathiocane (S_8) and metals, thus creating a circular economy.

Some important implications can be drawn from the study based on the findings. The successful use of geochemical modelling has shown that solution thermodynamics and equilibria remain important as aspects determining the chemistry of water. Much as statistical models can unravel patterns in the data, the chemistry remains key in explaining the real processes underlying the observations. This was quite apparent in models that were developed using treated water and those using untreated water.

The use of ML techniques to complement sparse datasets has shown that these techniques play an important role in dealing with complex data and can be relied upon to give better results compared to traditional techniques. More important, has been the use of these techniques together with geochemical modelling to give better understanding of chemical processes occurring in the water that determine what the ML techniques are revealing as observed patterns. This does not only have implications for water related studies, but other studies such as those on drug design, drug docking, catalysis and ecotoxicity. Chemistry data used in drug-related research, for instance, is generated from traditional computational chemistry methods (e.g., density functional theory) and as such, ML can be introduced to augment these and to achieve better and more robust models.

A simple approach to creating an application based on results from ML models that can potentially be used to approximate missing values of parameters has been demonstrated. This has implications for efforts that are aimed at automating the water analysis process. This means that such an app can be combined with other methods involving internet of things e.g., connected sensors that can relay real time analytical data to a data cache. The app development also shows the possibility of scaling up this approach to software development, not only for water chemistry, but also for other areas in chemistry that were indicated previously. As indicated in the relevant publication, development of such apps can be taught to chemistry students.

Based on the findings and their implications above, the following conclusions were reached about the study.

1. Historical data obtained from the three AMD treatment plants in Johannesburg, South Africa (*viz*., Central Rand, East Rand and West Rand) contained inherent inaccuracies and many outliers which must be removed or corrected for by modelling in order to ensure good clean data is used to train models. The presence of bad datapoints or a dataset that has no correlation leads to extremely low predictive accuracies.

2. Datasets must be thoroughly cleaned using geochemical modelling in order to thermodynamically correct for charge imbalance and this significantly altered the

sulphate levels. This gives further proof for the inaccuracies present in traditional chemistry approaches since they greatly overestimate the sulphate levels.

3. Temperature (°C), Total Dissolved Solids (mg/L) and most importantly, iron (mg/L) were found to be the most important and most correlated values that were needed to predict sulphate levels. The incredible linear correlation between iron and sulphate indicated that one of the main constituents in AMD could be dissolved pyrites.

4. Water quality parameter levels were found to be highly dependent on a variety of factors such as weather, season, geography and location of the treatment plants. The Treated Water chemistry was significantly different to that of the untreated AMD due to the addition of basic neutralizing agents and the various physical processes it goes through during the treatment process.

5. The large Central Rand dataset was successfully used to train a variety of ML regression models using the untreated AMD (Pump A and Pump B) subsets. Models trained on the Treated Water subset did not perform well as the maximum R^2 value obtained was 0.14. This was due to the Treated Water subset being highly uncorrelated and contained too many outliers. Thus, no usable model was obtained from the Treated Water dataset.

6. Several baseline models were trained and compared on all Central Rand subsets and revealed that ensemble methods (bagging, boosting and stacking) did outperform individual models. However, when comparing stacking ensemble ML that combined all the baseline models with stacking ensemble ML that only combined the best performing models, it was found that there was no significant improvement in excluding bad models from the stack as long as the good models were included. In one case, it was actually beneficial to include the bad performing models. All models were trained in under 2 minutes which proved the benefit in using ML approaches when compared to traditional approaches.

7. Central Rand Pump B produced the best performing model, which was a stacking regressor that combined the best models namely, LR, RD, KNNR, DT, XG, RF and MLP as the level 0 models and LR as the level 1 model. This model performed phenomenally well and achieved MSE of 0.000011, MAE of 0.002617 and R^2 of 0.999737 in under 2 minutes. This model was selected to be used for TL to the East Rand and West Rand datasets.

8. TL was successfully conducted on the cleaned and modelled East Rand AMD dataset using the Central Rand Pump B stacking regressor and achieved a high level of accuracy (MSE:0.00124, MAE:0.0290 and R^2:0.963) between the predicted and true sulphate values. This was achieved despite a drastic difference in the distributions between the Central Rand and East Rand datasets which further proved the power of utilizing ML in environmental sciences and analytical chemistry.

9. The East Rand Treated Water subset could not be cleaned using geochemical modelling due to inherently faulty data and, therefore, no sulphate values were predicted on this dataset using TL.

10. TL was successful in imputing missing values in the West Rand dataset by predicting sulphate levels in the cleaned and modelled West Rand AMD and Treated Water datasets. No true values for sulphate levels in the West Rand dataset was given and so no accuracy comparisons could be made. However, a general baseline idea of the amount of sulphate present in the West Rand treatment plant could now be understood.

11. The sulphate levels in all three treatment plants (Central Rand, East Rand and West Rand) were found to greatly differ from each other with Central Rand having the most normal distribution, East Rand having the most precise distribution and West Rand having the most variable distribution.

12. Whilst the sulphate levels in the treated effluent waters could not reliably be predicted due to inherent issues and poor correlations within the treated water datasets, sulphate levels in the untreated AMD datasets were successfully predicted with a high degree of accuracy.

13. ChatGPT was successfully used to generate working code needed to create a GUI ("Sulfind") for the trained regression model to predict sulphate levels when combined with a human negative error message feedback loop. A web-based version of the GUI can be found at the following link https://sulfind-webapp-gui-c4ce20a6aee6.herokuapp.com/.

14. Since the datasets obtained were for the years of 2015-2020, it was expected that data from March 2020 onwards would be askew due to the onset of South African national lockdowns due to the Covid 19 pandemic. However, surprisingly, the data showed no significant change nor variation from March 2020 onwards when compared to previous

years. This was an indication that the laboratory analysts and AMD plant operators were deemed to be “essential workers” and was encouraging to see that chemistry played a vital role in the economy and our lives.

15. Three AMD treatment plant sulphate datasets were accurately and successfully produced using ML regression, ensemble and transfer learning techniques. This data can be used as a pre-cursor for future AMD remediation studies in order to potentially recover octathiocane and create a circular economy.

4.2 Future Work

Some recommendations are presented that can help to expand the scope of this study and to improve the accuracy and reliability of the results where inconclusive datasets are encountered.

1. To shift the storage of the AMD treatment plant datasets to a secure cloud-based platform such as Google Cloud, IBM Cloud or Amazon Web Services (AWS) as this will allow for quicker and easier scalability and deployment of the created GUIs. This will also enable hosting of other data e.g., data collected from sensors in case where approaches based on the internet of things have to be implemented.

2. To extend the Sulfind GUI to be able to handle Excel Spreadsheets as inputs and provide a predicted sulphate levels as an Excel file output. This can be introduced as an assignment or project for chemistry students, for instance.

3. Additional AMD treatment plant data should be gathered and analysed to enhance the amount of training data and to remove any inherent biases present in the model from being trained on only the Central Rand dataset. This implies that analytical work in the two other plants should be improved for this to be achieved.

4. AMD treatment plant datasets that contain measured calcium (Ca) and magnesium (Mg) values should be sought after as these cations are usually present in high abundance in the treated waters and would significantly affect the charge balance used during geochemical modelling. The three AMD plants should be advised to start measuring the Ca and Mg levels and record them.

5. Baseline regression models and stacking regressors should be trained on the East Rand dataset and TL should be conducted from the East Rand dataset to the Central

Rand dataset (i.e., reversed procedure) in order to further validate the accuracy and applicability of TL. However, this can only be achieved if reliable analytical data is available.

6. Since water quality data is inherently stochastic and contains many parameters, in addition to geochemical modelling, physics-informed machine learning should be used to incorporate noisy high-dimensional data into algorithms.[171] This would improve the overall accuracy of the trained model.

7. Grid search parameter optimization should be conducted after training models to ensure that the best combination of parameters per model are discovered. This will ensure that the accuracy of the models is further improved and will result in an even fairer comparison between models.

REFERENCES

(1) Westall, F.; Brack, A. The Importance of Water for Life. *Space Sci Rev* **2018**, *214* (2), 50. https://doi.org/10.1007/s11214-018-0476-7.

(2) Ball, P. The Importance of Water. In *Astrochemistry and Astrobiology*; Springer Berlin Heidelberg: Berlin, Heidelberg, 2013; pp 169–210. https://doi.org/10.1007/978-3-642-31730-9_6.

(3) Gleick, P. The Human Right to Water. *Water Policy* **1998**, *1* (5), 487–503. https://doi.org/10.1016/S1366-7017(99)00008-2.

(4) Kemp, D.; Bond, C. J.; Franks, D. M.; Cote, C. Mining, Water and Human Rights: Making the Connection. *J Clean Prod* **2010**, *18* (15), 1553–1562. https://doi.org/10.1016/j.jclepro.2010.06.008.

(5) *Goal 6 | Ensure availability and sustainable management of water and sanitation for all*. https://sdgs.un.org/goals/goal6 (accessed 2023-02-06).

(6) *SDG Report 2022_Goal 6 infographic.png (5401×7201)*. https://sdgs.un.org/sites/default/files/2022-07/SDG%20Report%202022_Goal%206%20infographic.png (accessed 2023-02-06).

(7) *Goal 13 | Take urgent action to combat climate change and its impacts*. https://sdgs.un.org/goals/goal13 (accessed 2023-02-12).

(8) Cosgrove, W. J.; Loucks, D. P. Water Management: Current and Future Challenges and Research Directions. *Water Resour Res* **2015**, *51* (6), 4823–4839. https://doi.org/10.1002/2014WR016869.

(9) Boretti, A.; Rosa, L. Reassessing the Projections of the World Water Development Report. *NPJ Clean Water* **2019**, *2* (1), 15. https://doi.org/10.1038/s41545-019-0039-9.

(10) Smol, M.; Adam, C.; Preisner, M. Circular Economy Model Framework in the European Water and Wastewater Sector. *J Mater Cycles Waste Manag* **2020**, *22* (3), 682–697. https://doi.org/10.1007/s10163-019-00960-z.

(11) Ghimire, U.; Sarpong, G.; Gude, V. G. Transitioning Wastewater Treatment Plants toward Circular Economy and Energy Sustainability. *ACS Omega* **2021**, *6* (18), 11794–11803. https://doi.org/10.1021/acsomega.0c05827.

(12) Abu-Ghunmi, D.; Abu-Ghunmi, L.; Kayal, B.; Bino, A. Circular Economy and the Opportunity Cost of Not “closing the Loop” of Water Industry: The Case of Jordan. *J Clean Prod* **2016**, *131*, 228–236. https://doi.org/10.1016/j.jclepro.2016.05.043.

(13) Delgado, A.; Rodriguez, D. J.; Amadei, C. A.; Makino, M. *WATER IN CIRCULAR ECONOMY AND RESILIENCE (WICER)*; 2021. www.worldbank.org.

(14) Hoosain, M. S.; Paul, B. S.; Doorsamy, W.; Ramakrishna, S. The Influence of Circular Economy and 4IR Technologies on the Climate–Water–Energy–Food Nexus and the SDGs. *Water (Switzerland)* **2023**, *15* (4). https://doi.org/10.3390/w15040787.

(15) Rodriguez-Anton, J. M.; Rubio-Andrada, L.; Celemín-Pedroche, M. S.; Alonso-Almeida, M. D. M. Analysis of the Relations between Circular Economy and Sustainable Development Goals. *International Journal of Sustainable Development and World Ecology* **2019**, *26* (8), 708–720. https://doi.org/10.1080/13504509.2019.1666754.

(16) Masindi, V.; Chatzisymeon, E.; Kortidis, I.; Foteinis, S. Assessing the Sustainability of Acid Mine Drainage (AMD) Treatment in South Africa. *Science of The Total Environment* **2018**, *635*, 793–802. https://doi.org/10.1016/j.scitotenv.2018.04.108.

(17) Simate, G. S.; Ndlovu, S. Acid Mine Drainage: Challenges and Opportunities. *J Environ Chem Eng* **2014**, *2* (3), 1785–1803. https://doi.org/10.1016/j.jece.2014.07.021.

(18) Akcil, A.; Koldas, S. Acid Mine Drainage (AMD): Causes, Treatment and Case Studies. *J Clean Prod* **2006**, *14* (12–13), 1139–1145. https://doi.org/10.1016/j.jclepro.2004.09.006.

(19) Henson, J. M.; Casstevens, C. Natural Hazard Regulation: Adaptations for an Urban River. In *Imperiled: The Encyclopedia of Conservation*; Elsevier, 2022; pp 192–197. https://doi.org/10.1016/B978-0-12-821139-7.00197-5.

(20) *Acid Mine Drainage and Human Rights*. https://www.sahrc.org.za/home/21/files/AMD%20Booklet.pdf (accessed 2023-04-19).

(21) Kemp, D.; Bond, C. J.; Franks, D. M.; Cote, C. Mining, Water and Human Rights: Making the Connection. *J Clean Prod* **2010**, *18* (15), 1553–1562. https://doi.org/10.1016/j.jclepro.2010.06.008.

(22) du Toit, D. *The heat of acid mine drainage*. The heat of acid mine drainage (accessed 2023-06-27).

(23) Naidu, G.; Ryu, S.; Thiruvenkatachari, R.; Choi, Y.; Jeong, S.; Vigneswaran, S. A Critical Review on Remediation, Reuse, and Resource Recovery from Acid Mine Drainage. *Environmental Pollution* **2019**, *247*, 1110–1124. https://doi.org/10.1016/j.envpol.2019.01.085.

(24) Johnson, D. B.; Hallberg, K. B. Acid Mine Drainage Remediation Options: A Review. *Science of The Total Environment* **2005**, *338* (1–2), 3–14. https://doi.org/10.1016/J.SCITOTENV.2004.09.002.

(25) Skousen, J.; Rose, A.; Geidel, G.; Foreman, J.; Evans, R.; Hellier, W. *HANDBOOK OF TECHNOLOGIES FOR AVOIDANCE AND REMEDIATION OF ACID MINE DRAINAGE*; The National Mine Land Reclamation Center: West Virginia, 1998.

(26) *5.3 Mineral Groups*. University of Saskatchewan. https://openpress.usask.ca/physicalgeology/chapter/5-3-mineral-groups-2/ (accessed 2023-06-27).

(27) Zdun, T. MODELLING THE HYDRODYNAMICS OF COLLIE MINING VOID 5B, The University of Western Australia, 2001.

(28) The Global Acid Rock Drainage Guide. In *INAP: The International Network for Acid Prevention*; INAP: The International Network for Acid Prevention, 2018.

(29) Dold, B. *Waste Management*; Sunil, E., Ed.; InTech, 2010. https://doi.org/10.5772/185.

(30) Blowes, D. W.; Ptacek, C. J.; Jambor, J. L.; Weisener, C. G. The Geochemistry of Acid Mine Drainage. In *Treatise on Geochemistry*; Elsevier, 2003; pp 149–204. https://doi.org/10.1016/B0-08-043751-6/09137-4.

(31) Dhir, B. Biotechnological Tools for Remediation of Acid Mine Drainage (Removal of Metals From Wastewater and Leachate). In *Bio-Geotechnologies for Mine Site Rehabilitation*; Elsevier, 2018; pp 67–82. https://doi.org/10.1016/B978-0-12-812986-9.00004-X.

(32) Netshiongolwe, K. E.; Tutu, H.; Nuapia Yannick. Repurposing of Sludge Generated from the Treatment of Acid Mine Drainage, 2022.

(33) *Chapter 7 - Drainage Treatment*. http://www.gardguide.com/index.php/Chapter_7 (accessed 2023-02-06).

(34) De Beer, M.; Maree, J. P.; Wilsenach, J.; Motaung, S.; Bologo, L.; Radebe, V. *Acid Mine Water Reclamation Using the ABC Process*; 2010.

https://researchspace.csir.co.za/dspace/bitstream/handle/10204/5137/De%20Beer2_2010.pdf?sequence=1&isAllowed=y (accessed 2023-06-28).

(35) Kushkevych, I.; Hýžová, B.; Vítězová, M.; Rittmann, S. K.-M. R. Microscopic Methods for Identification of Sulfate-Reducing Bacteria from Various Habitats. *Int J Mol Sci* **2021**, *22* (8), 4007. https://doi.org/10.3390/ijms22084007.

(36) Steudel, R. Mechanism for the Formation of Elemental Sulfur from Aqueous Sulfide in Chemical and Microbiological Desulfurization Processes. *Ind Eng Chem Res* **1996**, *35* (4), 1417–1423. https://doi.org/10.1021/ie950558t.

(37) *Octathiocane | S8 - PubChem*. https://pubchem.ncbi.nlm.nih.gov/compound/Octathiocane#section=Crystal-Structures (accessed 2023-02-06).

(38) *Octathiocane | S589550* . https://www.smolecule.com/products/s589550 (accessed 2023-02-06).

(39) Carretero, M. I.; Pozo, M. Clay and Non-Clay Minerals in the Pharmaceutical and Cosmetic Industries Part II. Active Ingredients. *Appl Clay Sci* **2010**, *47* (3–4), 171–181. https://doi.org/10.1016/j.clay.2009.10.016.

(40) *Sulfur: Uses, Interactions, Mechanism of Action | DrugBank Online*. https://go.drugbank.com/drugs/DB09353 (accessed 2023-02-07).

(41) Stolz, J. *Sulphur: Mineral information, data and localities.* https://www.mindat.org/min-3826.html (accessed 2023-06-28).

(42) Reisman, D. J.; Sundaram, V.; Al-Abed, S. R.; Allen, D. Statistical Validation of Sulfate Quantification Methods Used for Analysis of Acid Mine Drainage. *Talanta* **2007**, *71* (1), 303–311. https://doi.org/10.1016/j.talanta.2006.04.002.

(43) Roy, A.; Das, B. K.; Bhattacharya, J. Development and Validation of a Spectrophotometric Method to Measure Sulfate Concentrations in Mine Water without Interference. *Mine Water Environ* **2011**, *30* (3), 169–174. https://doi.org/10.1007/s10230-011-0140-x.

(44) Raja, P. M. V; Barron, A. R. *1.5: ICP-AES Analysis of Nanoparticles - Chemistry LibreTexts*. https://chem.libretexts.org/Bookshelves/Analytical_Chemistry/Physical_Methods_in_Chemistry_and_Nano_Science_(Barron)/01%3A_Elemental_Analysis/1.05%3A_ICP-AES_Analysis_of_Nanoparticles (accessed 2023-04-30).

(45) SI Analytics. *Determination of Sulfate in Drinking, Surface and Waste Water*. https://www.xylemanalytics.com/en/File%20Library/Resource%20Library/SIA/09%20Application%20Papers/UK/SO4-with-Ca-ISE-and-EGTA_EN.pdf (accessed 2023-02-07).

(46) Anechiţei, L.; Cojocaru, T.; Munteanu, I.-G.; Bulgariu, L. Simple Methods For Quantitative Determination Of Sulphate Ions From Aqueous Media With Industrial Applications. *Bulletin of the Polytechnic Institute of Jassy* **2019**, *65*, 27–37.

(47) Costa, M. C.; Martins, M.; Jesus, C.; Duarte, J. C. Treatment of Acid Mine Drainage by Sulphate-Reducing Bacteria Using Low Cost Matrices. *Water Air Soil Pollut* **2008**, *189* (1–4), 149–162. https://doi.org/10.1007/s11270-007-9563-1.

(48) Mccarthy, J. WHAT IS ARTIFICIAL INTELLIGENCE? Computer Science Department Stanford University 2007. http://www-formal.stanford.edu/jmc/ (accessed 2023-02-07).

(49) *What is Artificial Intelligence (AI) ? | IBM*. https://www.ibm.com/topics/artificial-intelligence (accessed 2023-02-07).

(50) *Artificial intelligence ensuring human rights at the heart of the Sustainable Development Goals*. https://www.ohchr.org/en/stories/2021/03/artificial-intelligence-ensuring-human-rights-heart-sustainable-development-goals (accessed 2023-02-13).

(51) *Bringing Data to Life*; 2022. https://unstats.un.org/sdgs/report/2022/SDG2022_Flipbook_final.pdf (accessed 2023-02-13).

(52) *UN 2023 Water Conference*. https://sdgs.un.org/conferences/water2023 (accessed 2023-02-13).

(53) *What is Machine Learning? | IBM*. https://www.ibm.com/topics/machine-learning (accessed 2023-02-07).

(54) Zhong, S.; Zhang, K.; Bagheri, M.; Burken, J. G.; Gu, A.; Li, B.; Ma, X.; Marrone, B. L.; Ren, Z. J.; Schrier, J.; Shi, W.; Tan, H.; Wang, T.; Wang, X.; Wong, B. M.; Xiao, X.; Yu, X.; Zhu, J.-J.; Zhang, H. Machine Learning: New Ideas and Tools in Environmental Science and Engineering. *Environ Sci Technol* **2021**, *55* (19), 12741–12754. https://doi.org/10.1021/acs.est.1c01339.

(55) Alzubi, J.; Nayyar, A.; Kumar, A. Machine Learning from Theory to Algorithms: An Overview. *J Phys Conf Ser* **2018**, *1142*, 012012. https://doi.org/10.1088/1742-6596/1142/1/012012.

(56) *What is Deep Learning?* | *IBM*. https://www.ibm.com/za-en/topics/deep-learning (accessed 2023-02-07).

(57) *Beginner Resources for AI, ML, and DL – The Data Bistro*. https://databistro.tech/beginner-resources-for-ai-ml-and-dl/ (accessed 2023-02-07).

(58) Perović, M.; Šenk, I.; Tarjan, L.; Obradović, V.; Dimkić, M. Machine Learning Models for Predicting the Ammonium Concentration in Alluvial Groundwaters. *Environmental Modeling & Assessment* **2021**, *26* (2), 187–203. https://doi.org/10.1007/s10666-020-09731-9.

(59) Arora, S.; Keshari, A. K. Implementing Machine Learning Algorithm to Model Reaeration Coefficient of Urbanized Rivers. *Environmental Modeling & Assessment* **2023**. https://doi.org/10.1007/s10666-023-09895-0.

(60) Bordoni, M.; Bittelli, M.; Valentino, R.; Chersich, S.; Persichillo, M. G.; Meisina, C. Soil Water Content Estimated by Support Vector Machine for the Assessment of Shallow Landslides Triggering: The Role of Antecedent Meteorological Conditions. *Environmental Modeling & Assessment* **2018**, *23* (4), 333–352. https://doi.org/10.1007/s10666-017-9586-y.

(61) Wang, J.; Geng, Y.; Zhao, Q.; Zhang, Y.; Miao, Y.; Yuan, X.; Jin, Y.; Zhang, W. Water Quality Prediction of Water Sources Based on Meteorological Factors Using the CA-NARX Approach. *Environmental Modeling and Assessment* **2021**, *26* (4), 529–541. https://doi.org/10.1007/s10666-021-09759-5.

(62) Foroughi, M.; Rahmani, A. R.; Asgari, G.; Nematollahi, D.; Yetilmezsoy, K.; Samarghandi, M. R. Optimization and Modeling of Tetracycline Removal from Wastewater by Three-Dimensional Electrochemical System: Application of Response Surface Methodology and Least Squares Support Vector Machine. *Environmental Modeling & Assessment* **2020**, *25* (3), 327–341. https://doi.org/10.1007/s10666-019-09675-9.

(63) He, H.; Li, W.; Qian, M.; Hu, S. Time Series Clustering and Influencing Factors Analysis on Qinghai-Tibet Plateau Lake Area Change. *Environmental Modeling & Assessment* **2023**. https://doi.org/10.1007/s10666-023-09913-1.

(64) Jabeur, S. Ben; Ballouk, H.; Arfi, W. Ben; Khalfaoui, R. Machine Learning-Based Modeling of the Environmental Degradation, Institutional Quality, and Economic Growth. *Environmental Modeling & Assessment* **2022**, *27* (6), 953–966. https://doi.org/10.1007/s10666-021-09807-0.

(65) Guzman, S. M.; Paz, J. O.; Tagert, M. L. M.; Mercer, A. E. Evaluation of Seasonally Classified Inputs for the Prediction of Daily Groundwater Levels: NARX Networks Vs Support Vector Machines. *Environmental Modeling & Assessment* **2019**, *24* (2), 223–234. https://doi.org/10.1007/s10666-018-9639-x.

(66) Arabgol, R.; Sartaj, M.; Asghari, K. Predicting Nitrate Concentration and Its Spatial Distribution in Groundwater Resources Using Support Vector Machines (SVMs) Model. *Environmental Modeling & Assessment* **2016**, *21* (1), 71–82. https://doi.org/10.1007/s10666-015-9468-0.

(67) *Supervised vs. Unsupervised Learning: What's the Difference?* | *IBM*. https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning (accessed 2023-02-07).

(68) *6 Types of Regression Models in Machine Learning You Should Know About*. https://medium.com/@uma.bollikonda/6-types-of-regression-models-in-machine-learning-you-should-know-about-ed1e83dd0c4d (accessed 2023-02-08).

(69) *6 Types of Regression Models in Machine Learning You Should Know About*. https://www.upgrad.com/blog/types-of-regression-models-in-machine-learning/ (accessed 2023-02-08).

(70) Maulud, D.; Abdulazeez, A. M. A Review on Linear Regression Comprehensive in Machine Learning. *Journal of Applied Science and Technology Trends* **2020**, *1* (4), 140–147. https://doi.org/10.38094/jastt1457.

(71) *Lasso and Ridge Regression in Python Tutorial*. https://www.datacamp.com/tutorial/tutorial-lasso-ridge-regression (accessed 2023-05-01).

(72) Muthukrishnan, R.; Rohini, R. LASSO: A Feature Selection Technique in Predictive Modeling for Machine Learning. In *2016 IEEE International Conference on Advances in Computer Applications (ICACA)*; IEEE, 2016; pp 18–20. https://doi.org/10.1109/ICACA.2016.7887916.

(73) *Lasso Regression: Simple Definition*. https://www.statisticshowto.com/lasso-regression/ (accessed 2023-02-08).

(74) *5.1 - Ridge Regression*. https://online.stat.psu.edu/stat857/node/155/ (accessed 2023-02-08).

(75) *Elastic Net*. https://corporatefinanceinstitute.com/resources/data-science/elastic-net/ (accessed 2023-04-30).

(76) Brownlee, J. *How to Develop Elastic Net Regression Models in Python*. https://machinelearningmastery.com/elastic-net-regression-in-python/ (accessed 2023-04-30).

(77) García-Nieto, P. J.; García-Gonzalo, E.; Paredes-Sánchez, J. P. Prediction of the Critical Temperature of a Superconductor by Using the WOA/MARS, Ridge, Lasso and Elastic-Net Machine Learning Techniques. *Neural Comput Appl* **2021**, *33* (24), 17131–17145. https://doi.org/10.1007/s00521-021-06304-z.

(78) Santos, G. *KNN Regression Model in Python*. https://towardsdatascience.com/knn-regression-model-in-python-9868f21c9fa2 (accessed 2023-05-01).

(79) *What is the k-nearest neighbors algorithm?* https://www.ibm.com/topics/knn (accessed 2023-05-01).

(80) Kramer, O. K-Nearest Neighbors. In *Dimensionality Reduction with Unsupervised Nearest Neighbors*; Springer: Berlin, 2013; Vol. 51, pp 13–23. https://doi.org/10.1007/978-3-642-38652-7_2.

(81) Raj, A. *Unlocking the True Power of Support Vector Regression*. https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0 (accessed 2023-02-09).

(82) *Machine Learning Basics: Support Vector Regression*. https://towardsdatascience.com/machine-learning-basics-support-vector-regression-660306ac5226 (accessed 2023-02-09).

(83) Saini, A. *Support Vector Machine(SVM): A Complete guide for beginners*. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/ (accessed 2023-05-01).

(84) Awad, M.; Khanna, R. Support Vector Regression. In *Efficient Learning Machines*; Apress: Berkeley, CA, 2015; pp 67–80. https://doi.org/10.1007/978-1-4302-5990-9_4.

(85) Shi, A. *Decision Tree Regressor — A Visual Guide with Scikit Learn*. https://towardsdatascience.com/decision-tree-regressor-a-visual-guide-with-scikit-learn-2aa9e01f5d7f (accessed 2023-05-01).

(86) Chouinard, J.-C. *Decision Trees in Machine Learning, with Examples (Python)*. https://www.jcchouinard.com/decision-trees-in-machine-learning/ (accessed 2023-05-01).

(87) Pekel, E. Estimation of Soil Moisture Using Decision Tree Regression. *Theor Appl Climatol* **2020**, *139* (3–4), 1111–1119. https://doi.org/10.1007/s00704-019-03048-8.

(88) Lev, A. *XGBoost versus Random Forest*. https://www.qwak.com/post/xgboost-versus-random-forest (accessed 2023-05-02).

(89) Kavzoglu, T.; Teke, A. Predictive Performances of Ensemble Machine Learning Algorithms in Landslide Susceptibility Mapping Using Random Forest, Extreme Gradient Boosting (XGBoost) and Natural Gradient Boosting (NGBoost). *Arab J Sci Eng* **2022**, *47*, 7367–7385. https://doi.org/10.1007/s13369-022-06560-8.

(90) Singhal, G. *Ensemble Methods in Machine Learning: Bagging Versus Boosting*. https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting (accessed 2023-05-02).

(91) *Random Forest Regression*. https://levelup.gitconnected.com/random-forest-regression-209c0f354c84 (accessed 2023-02-09).

(92) Beheshti, N. *Random Forest Regression*. https://towardsdatascience.com/random-forest-regression-5f605132d19d (accessed 2023-02-09).

(93) Liu, Y.; Wang, Y.; Zhang, J. New Machine Learning Algorithm: Random Forest; Springer: Berlin, 2012; pp 246–252. https://doi.org/10.1007/978-3-642-34062-8_32.

(94) *What is a Neural Network?* https://www.tibco.com/reference-center/what-is-a-neural-network (accessed 2023-02-09).

(95) *What are Neural Networks?* | *IBM*. https://www.ibm.com/topics/neural-networks (accessed 2023-02-09).

(96) Vivanco-Benavides, L. E.; Martínez-González, C. L.; Mercado-Zúñiga, C.; Torres-Torres, C. Machine Learning and Materials Informatics Approaches in the Analysis of Physical Properties of Carbon Nanotubes: A Review. *Comput Mater Sci* **2022**, *201*, 110939. https://doi.org/10.1016/J.COMMATSCI.2021.110939.

(97) Ferreira, B.; Iten, M.; Silva, R. G. Monitoring Sustainable Development by Means of Earth Observation Data and Machine Learning: A Review. *Environ Sci Eur* **2020**, *32* (1), 120. https://doi.org/10.1186/s12302-020-00397-4.

(98) Kou, X.; Han, D.; Cao, Y.; Shang, H.; Li, H.; Zhang, X.; Yang, M. Acid Mine Drainage Discrimination Using Very High Resolution Imagery Obtained by Unmanned Aerial Vehicle in a Stone Coal Mining Area. *Water (Switzerland)* **2023**, *15* (8). https://doi.org/10.3390/w15081613.

(99) Rigueira, X.; Pazo, M.; Araújo, M.; Gerassis, S.; Bocos, E. Bayesian Machine Learning and Functional Data Analysis as a Two-Fold Approach for the Study of Acid Mine Drainage Events. *Water (Switzerland)* **2023**, *15* (8). https://doi.org/10.3390/w15081553.

(100) Trifi, M.; Gasmi, A.; Carbone, C.; Majzlan, J.; Nasri, N.; Dermech, M.; Charef, A.; Elfil, H. Machine Learning-Based Prediction of Toxic Metals Concentration in an Acid Mine Drainage Environment, Northern Tunisia. *Environmental Science and Pollution Research* **2022**, *29* (58), 87490–87508. https://doi.org/10.1007/s11356-022-21890-8.

(101) More, K. S.; Wolkersdorfer, C. Predicting and Forecasting Mine Water Parameters Using a Hybrid Intelligent System. *Water Resources Management* **2022**, *36* (8), 2813–2826. https://doi.org/10.1007/s11269-022-03177-2.

(102) De Jesus, K. L. M.; Senoro, D. B.; Dela Cruz, J. C.; Chan, E. B. A Hybrid Neural Network–Particle Swarm Optimization Informed Spatial Interpolation Technique for Groundwater Quality Mapping in a Small Island Province of the Philippines. *Toxics* **2021**, *9* (11), 273. https://doi.org/10.3390/toxics9110273.

(103) Flores, H.; Lorenz, S.; Jackisch, R.; Tusa, L.; Cecilia Contreras, I.; Zimmermann, R.; Gloaguen, R. Uas-Based Hyperspectral Environmental Monitoring of Acid Mine Drainage Affected Waters. *Minerals* **2021**, *11* (2), 1–25. https://doi.org/10.3390/min11020182.

(104) Ma, L.; Huang, C.; Liu, Z. S.; Morin, K. A.; Aziz, M.; Meints, C. Artificial Neural Network for Prediction of Full-Scale Seepage Flow Rate at the Equity Silver Mine. *Water Air Soil Pollut* **2020**, *231* (4). https://doi.org/10.1007/s11270-020-04541-x.

(105) Betrie, G. D.; Sadiq, R.; Morin, K. A.; Tesfamariam, S. Uncertainty Quantification and Integration of Machine Learning Techniques for Predicting Acid Rock Drainage Chemistry: A Probability Bounds Approach. *Science of the Total Environment* **2014**, *490*, 182–190. https://doi.org/10.1016/j.scitotenv.2014.04.125.

(106) Betrie, G. D.; Tesfamariam, S.; Morin, K. A.; Sadiq, R. Predicting Copper Concentrations in Acid Mine Drainage: A Comparative Analysis of Five Machine

Learning Techniques. *Environ Monit Assess* **2013**, *185* (5), 4171–4182. https://doi.org/10.1007/s10661-012-2859-7.

(107) Brownlee, J. *Stacking Ensemble Machine Learning With Python*. https://machinelearningmastery.com/stacking-ensemble-machine-learning-with-python/ (accessed 2023-02-09).

(108) Qasem, A. G.; Lam, S. S. Prediction of Wart Treatment Response Using a Hybrid GA-Ensemble Learning Approach. *Expert Syst Appl* **2023**, *221*. https://doi.org/10.1016/j.eswa.2023.119737.

(109) Dritsas, E.; Trigka, M. Efficient Data-Driven Machine Learning Models for Water Quality Prediction. *Computation* **2023**, *11* (2), 16. https://doi.org/10.3390/computation11020016.

(110) Moeini, M.; Shojaeizadeh, A.; Geza, M. Supervised Stacking Ensemble Machine Learning Approach for Enhancing Prediction of Total Suspended Solids Concentration in Urban Watersheds. *Journal of Environmental Engineering* **2022**, *148* (6). https://doi.org/10.1061/(asce)ee.1943-7870.0001998.

(111) Li, Z.; Zhang, C.; Liu, H.; Zhang, C.; Zhao, M.; Gong, Q.; Fu, G. Developing Stacking Ensemble Models for Multivariate Contamination Detection in Water Distribution Systems. *Science of The Total Environment* **2022**, *828*, 154284. https://doi.org/10.1016/j.scitotenv.2022.154284.

(112) *Ensemble Stacking for Machine Learning and Deep Learning*. https://www.analyticsvidhya.com/blog/2021/08/ensemble-stacking-for-machine-learning-and-deep-learning/ (accessed 2023-03-17).

(113) Torrey, L.; Shavlik, J. Transfer Learning. In *Handbook of Research on Machine Learning Applications and Trends*; IGI Global, 2010; pp 242–264. https://doi.org/10.4018/978-1-60566-766-9.ch011.

(114) Day, O.; Khoshgoftaar, T. M. A Survey on Heterogeneous Transfer Learning. *J Big Data* **2017**, *4* (1). https://doi.org/10.1186/S40537-017-0089-0.

(115) Jang, Y.; Lee, H.; Hwang, S. J.; Shin, J. Learning What and Where to Transfer. In *Proceedings of the 36th International Conference on Machine Learning*; 2019; pp 1–10.

(116) Qiu, J.; Wu, Q.; Ding, G.; Xu, Y.; Feng, S. A Survey of Machine Learning for Big Data Processing. *EURASIP J Adv Signal Process* **2016**, *2016* (1), 67. https://doi.org/10.1186/s13634-016-0355-x.

(117) Niu, S.; Liu, Y.; Wang, J.; Song, H. A Decade Survey of Transfer Learning (2010–2020). *IEEE Transactions on Artificial Intelligence* **2020**, *1* (2), 151–166. https://doi.org/10.1109/TAI.2021.3053511.

(118) Tariq, S.; Loy-Benitez, J.; Nam, K.; Lee, G.; Kim, M.; Park, D.; Yoo, C. Transfer Learning Driven Sequential Forecasting and Ventilation Control of PM2.5 Associated Health Risk Levels in Underground Public Facilities. *J Hazard Mater* **2021**, *406*, 124753. https://doi.org/10.1016/j.jhazmat.2020.124753.

(119) Himeur, Y.; Rimal, B.; Tiwary, A.; Amira, A. Using Artificial Intelligence and Data Fusion for Environmental Monitoring: A Review and Future Perspectives. *Information Fusion* **2022**, *86–87*, 44–75. https://doi.org/10.1016/j.inffus.2022.06.003.

(120) Hilal, A. M.; Al-Wesabi, F. N.; Alzahrani, K. J.; Al Duhayyim, M.; Ahmed Hamza, M.; Rizwanullah, M.; García Díaz, V. Deep Transfer Learning Based Fusion Model for Environmental Remote Sensing Image Classification Model. *Eur J Remote Sens* **2022**, *55* (sup1), 12–23. https://doi.org/10.1080/22797254.2021.2017799.

(121) Dong, J.; Sitler, K.; Scalia, J.; Ge, Y.; Bireta, P.; Sihota, N.; Hoelen, T. P.; Lowry, G. V. Application of Transfer Learning and Convolutional Neural Networks for Autonomous Oil Sheen Monitoring. *Applied Sciences* **2022**, *12* (17), 8865. https://doi.org/10.3390/app12178865.

(122) Zurowietz, M.; Nattkemper, T. W. Unsupervised Knowledge Transfer for Object Detection in Marine Environmental Monitoring and Exploration. *IEEE Access* **2020**, *8*, 143558–143568. https://doi.org/10.1109/ACCESS.2020.3014441.

(123) *Generative AI Certificate Q&A: What is the difference between generative AI and discriminative AI?* https://pupuweb.com/generative-ai-certificate-what-difference-discriminative/ (accessed 2023-09-12).

(124) Leng, C.; Tang, Z.; Zhou, Y.-G.; Tian, Z.; Huang, W.-Q.; Liu, J.; Li, K.; Li, K. Fifth Paradigm in Science: A Case Study of an Intelligence-Driven Material Design. *Engineering* **2023**. https://doi.org/10.1016/j.eng.2022.06.027.

(125) Zubarev, D. Y.; Pitera, J. W. Cognitive Materials Discovery and Onset of the 5th Discovery Paradigm. In *Machine Learning in Chemistry: Data-Driven Algorithms,*

Learning Systems, and Predictions; 2019; pp 103–120. https://doi.org/10.1021/bk-2019-1326.ch006.

(126) *Introducing ChatGPT*. https://openai.com/blog/chatgpt (accessed 2023-09-07).

(127) Humphry, T.; Fuller, A. L. Potential ChatGPT Use in Undergraduate Chemistry Laboratories. *J Chem Educ* **2023**, *100* (4), 1434–1436. https://doi.org/10.1021/acs.jchemed.3c00006.

(128) Alasadi, E. A.; Baiz, C. R. Generative AI in Education and Research: Opportunities, Concerns, and Solutions. *J Chem Educ* **2023**, *100* (8), 2965–2971. https://doi.org/10.1021/acs.jchemed.3c00323.

(129) Emenike, M. E.; Emenike, B. U. Was This Title Generated by ChatGPT? Considerations for Artificial Intelligence Text-Generation Software Programs for Chemists and Chemistry Educators. *J Chem Educ* **2023**, *100* (4), 1413–1418. https://doi.org/10.1021/acs.jchemed.3c00063.

(130) Tyson, J. Shortcomings of ChatGPT. *J Chem Educ* **2023**, *100* (8), 3098–3101. https://doi.org/10.1021/acs.jchemed.3c00361.

(131) Zhu, J.-J.; Jiang, J.; Yang, M.; Ren, Z. J. ChatGPT and Environmental Research. *Environ Sci Technol* **2023**. https://doi.org/10.1021/acs.est.3c01818.

(132) Clark, T. M. Investigating the Use of an Artificial Intelligence Chatbot with General Chemistry Exam Questions. *J Chem Educ* **2023**, *100* (5), 1905–1916. https://doi.org/10.1021/acs.jchemed.3c00027.

(133) Exintaris, B.; Karunaratne, N.; Yuriev, E. Metacognition and Critical Thinking: Using ChatGPT-Generated Responses as Prompts for Critique in a Problem-Solving Workshop (SMARTCHEMPer). *J Chem Educ* **2023**. https://doi.org/10.1021/acs.jchemed.3c00481.

(134) Yilmaz, R.; Karaoglan Yilmaz, F. G. Augmented Intelligence in Programming Learning: Examining Student Views on the Use of ChatGPT for Programming Learning. *Computers in Human Behavior: Artificial Humans* **2023**, *1* (2), 100005. https://doi.org/10.1016/j.chbah.2023.100005.

(135) Surameery, N. M. S.; Shakor, M. Y. Use Chat GPT to Solve Programming Bugs. *International Journal of Information technology and Computer Engineering* **2023**, No. 31, 17–22. https://doi.org/10.55529/ijitc.31.17.22.

(136) Biswas, S. Role of ChatGPT in Computer Programming. *Mesopotamian Journal of Computer Science* **2023**, 8–16. https://doi.org/10.58496/MJCSC/2023/002.

(137) Chen, E.; Huang, R.; Chen, H.-S.; Tseng, Y.-H.; Li, L.-Y. GPTutor: A ChatGPT-Powered Programming Tool for Code Explanation. **2023**. https://doi.org/https://doi.org/10.48550/arXiv.2305.01863.

(138) Holme, T. A. Can Today's Chemistry Curriculum Actually Produce Tomorrow's Adaptable Chemist? *J Chem Educ* **2019**, *96* (4), 611–612. https://doi.org/10.1021/ACS.JCHEMED.9B00112/ASSET/IMAGES/LARGE/ED-2019-00112B_0001.JPEG.

(139) Luo, Y. Chemistry in the Era of Artificial Intelligence. *Precision Chemistry* **2023**, *1* (2), 127–128. https://doi.org/10.1021/prechem.3c00038.

(140) Zhu, J.-J.; Yang, M.; Ren, Z. J. Machine Learning in Environmental Research: Common Pitfalls and Best Practices. *Environ Sci Technol* **2023**. https://doi.org/10.1021/acs.est.3c00026.

(141) *IODP data portal*. https://iodp.pangaea.de/front_content.php?idcat=119&count=10&q=&maxlat=&minlon=&maxlon=&minlat= (accessed 2023-09-12).

(142) Diepenbroek, M.; Grobe, H.; Reinke, M.; Schindler, U.; Schlitzer, R.; Sieger, R.; Wefer, G. PANGAEA—an Information System for Environmental Sciences. *Comput Geosci* **2002**, *28* (10), 1201–1210. https://doi.org/10.1016/S0098-3004(02)00039-0.

(143) Sharnowski, J. L.; Gannod, G. C.; Cheng, B. H. C. A Distributed, Multimedia Environmental Information System. In *Proceedings of the International Conference on Multimedia Computing and Systems*; IEEE Comput. Soc. Press, 1995; pp 142–149. https://doi.org/10.1109/MMCS.1995.484918.

(144) Tian, H.; Lu, W.; Li, T. O.; Tang, X.; Cheung, S.-C.; Klein, J.; Bissyandé, T. F. Is ChatGPT the Ultimate Programming Assistant -- How Far Is It? **2023**. https://doi.org/https://doi.org/10.48550/arXiv.2304.11938.

(145) Sobania, D.; Briesch, M.; Hanna, C.; Petke, J. An Analysis of the Automatic Bug Fixing Performance of ChatGPT. **2023**. https://doi.org/https://doi.org/10.48550/arXiv.2301.08653.

(146) *GAUTENG MAP*. https://www.freeworldmaps.net/africa/southafrica/gauteng-map.html (accessed 2023-05-07).

(147) Parkhurst, D.; Appelo, C. Description of Input and Examples for PHREEQC Version 3-A Computer Program for Speciation, Batch-Reaction, One-Dimensional Transport, and Inverse Geochemical Calculations. In *Section A, Groundwater Book 6, Modeling Techniques*; U.S. Geological Survey Techniques and Methods, 2013; p 497.

(148) PHREEQC Interactive. United States Geological Survey December 15, 1999.

(149) Chaudhray, S. *Why "1.5" in IQR Method of Outlier Detection?* . https://towardsdatascience.com/why-1-5-in-iqr-method-of-outlier-detection-5d07fdc82097 (accessed 2023-05-09).

(150) Kuhn, Max.; Johnson, Kjell. *Feature Engineering and Selection : A Practical Approach for Predictive Models.*; CRC Press LLC, 2019; Vol. 1.

(151) Witten, I.; Frank, E.; Hall, M.; Pal, C. *Data Mining: Practical Machine Learning Tools and Techniques*; Morgan Kaufmann, Ed.; Elsevier : Morgan Kaufmann, 2017; Vol. 4.

(152) Brownlee, J. *How to Use StandardScaler and MinMaxScaler Transforms in Python*. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/ (accessed 2023-05-09).

(153) Wold, S.; Esbensen, K.; Geladi, P. Principal Component Analysis. *Chemometrics and Intelligent Laboratory Systems* **1987**, *2* (1–3), 37–52. https://doi.org/10.1016/0169-7439(87)80084-9.

(154) Granitto, P. M.; Furlanello, C.; Biasioli, F.; Gasperi, F. Recursive Feature Elimination with Random Forest for PTR-MS Analysis of Agroindustrial Products. *Chemometrics and Intelligent Laboratory Systems* **2006**, *83* (2), 83–90. https://doi.org/10.1016/j.chemolab.2006.01.007.

(155) *K-means Cluster Analysis*. https://afit-r.github.io/kmeans_clustering#fnref:kauf (accessed 2023-10-22).

(156) Brownlee, J. *Regression Metrics for Machine Learning*. https://machinelearningmastery.com/regression-metrics-for-machine-learning/ (accessed 2023-05-10).

(157) Agrawal, R. *Evaluation Metrics for Your Regression Model*. https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/ (accessed 2023-05-10).

(158) *Is there any way to use Tkinter with Google Colaboratory* . https://saturncloud.io/blog/is-there-any-way-to-use-tkinter-with-google-colaboratory/ (accessed 2023-09-07).

(159) Valente, T. M.; Gomes, C. L. Occurrence, Properties and Pollution Potential of Environmental Minerals in Acid Mine Drainage. *Science of the Total Environment, The* **2008**, *407*, 1135–1152. https://doi.org/10.1016/j.scitotenv.2008.09.050.

(160) Chen, C.-J.; Jiang, W.-T. Influence of Waterfall Aeration and Seasonal Temperature Variation on the Iron and Arsenic Attenuation Rates in an Acid Mine Drainage System. *Applied Geochemistry* **2012**, *27* (10), 1966–1978. https://doi.org/10.1016/j.apgeochem.2012.06.003.

(161) Hatar, H.; Rahim, S. A.; Razi, W. M.; Sahrani, F. K. Heavy Metals Content in Acid Mine Drainage at Abandoned and Active Mining Area; 2013; pp 641–646. https://doi.org/10.1063/1.4858727.

(162) Johnson, D. B.; Hallberg, K. B. Acid Mine Drainage Remediation Options: A Review. *Science of The Total Environment* **2005**, *338* (1–2), 3–14. https://doi.org/10.1016/j.scitotenv.2004.09.002.

(163) Förstner, U.; Wittmann, G. T. W. Metal Accumulations in Acidic Waters from Gold Mines in South Africa. *Geoforum* **1976**, *7* (1), 41–49. https://doi.org/10.1016/0016-7185(76)90056-7.

(164) Jain, S. *Lasso & Ridge Regression | A Comprehensive Guide in Python & R*. https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/ (accessed 2023-05-24).

(165) Raj, A. *Unlocking the True Power of Support Vector Regression*. https://towardsdatascience.com/unlocking-the-true-power-of-support-vector-regression-847fd123a4a0 (accessed 2023-05-24).

(166) Dhiraj K. *Top 5 advantages and disadvantages of Decision Tree Algorithm*. https://dhirajkumarblog.medium.com/top-5-advantages-and-disadvantages-of-decision-tree-algorithm-428ebd199d9a (accessed 2023-05-28).

(167) Abyaneh, H. Z. Evaluation of Multivariate Linear Regression and Artificial Neural Networks in Prediction of Water Quality Parameters. **2014**. https://doi.org/10.1186/2052-336X-12-40.

(168) Emamgholizadeh, S.; Kashi, H.; Marofpoor, I.; Zalaghi, E. Prediction of Water Quality Parameters of Karoon River (Iran) by Artificial Intelligence-Based Models. *International Journal of Environmental Science and Technology* **2014**, *11* (3), 645–656. https://doi.org/10.1007/s13762-013-0378-x.

(169) Chapman, B. M.; Jones, D. R.; Jung, R. F. Processes Controlling Metal Ion Attenuation in Acid Mine Drainage Streams. *Geochimica ef Cosmwhimica Ado* **1983**, *47*, 1957–1973.

(170) *Dream by WOMBO*. https://dream.ai/ (accessed 2023-09-07).

(171) Karniadakis, G. E.; Kevrekidis, I. G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-Informed Machine Learning. *Nature Reviews Physics* **2021**, *3* (6), 422–440. https://doi.org/10.1038/s42254-021-00314-5.

APPENDICES

APPENDIX A: DATA PRE-CLEANING PYTHON CODE (CENTRAL RAND: PUMP A)

```
# Load the libraries and packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk

# In excel, change the headings to make them short and remove the units
 row as well as the date column
# For modelling only keep the columns for PUMP A : Temp, pH, sul, Fe, M
n

#Load dataset from excel. Note the read_excel() function

df = pd.read_excel('/content/PUMP A - Central Basin_Water quality 2015
to 2020.xlsx')

# DATA CLEANING

# Convert all empty spaces to NaN
df = df.replace(r'^\s*$', np.nan, regex=True)

#Convert words to NaN
df = df.replace('no distilled water', np.nan, regex=True)
df = df.replace('breakdown', np.nan, regex=True)
df = df.replace('pump 1 off', np.nan, regex=True)
df = df.replace('plant off', np.nan, regex=True)
df = df.replace('electrical maintenance', np.nan, regex=True)
df = df.replace('Pump 1 off', np.nan, regex=True)
df = df.replace('Pump off', np.nan, regex=True)
df = df.replace('RM', np.nan, regex=True)
df = df.replace('pump off', np.nan, regex=True)
df = df.replace('short routine ', np.nan, regex=True)
df = df.replace('vacuum pump damaged', np.nan, regex=True)
df = df.replace('no manganese reagent', np.nan, regex=True)
df = df.replace('no distilled watre', np.nan, regex=True)
df = df.replace('pump 2 off, due to main', np.nan, regex=True)
df = df.replace('Pump 2 off', np.nan, regex=True)
df = df.replace('pump 2 off', np.nan, regex=True)
df = df.replace('PUMP OFF', np.nan, regex=True)
df = df.replace('No evaporation dish', np.nan, regex=True)
df = df.replace('Evaporation dish broken', np.nan, regex=True)
```

```
df = df.replace('no overflow(8:00)', np.nan, regex=True)
df = df.replace('(8:00)', np.nan, regex=True)
df = df.replace('(8:00)', np.nan, regex=True)
df = df.replace('2.2: re-sample 7.090', np.nan, regex=True)
df = df.replace('No manganese reagent', 0 , regex=True)

# Fix numeric value typos
df = df.replace('26. 4', 26.4 , regex=True)
df = df.replace('21.`9', 21.9 , regex=True)
df = df.replace('28,3', 28.3 , regex=True)
df = df.replace('6..19', 6.19 , regex=True)
df = df.replace('0..01', 0.01 , regex=True)
df = df.replace('< 0.05', 0 , regex=True)
df = df.replace('<0.05', 0 , regex=True)

# Replaced the (plus minus 5%) in excel

# Drop all NaN values
df = df.dropna()

# Convert all columns to floats
df = df.astype(float)

df
```

	A - TEMP	A - pH	A-SULP	A-FE	A-MN
3	25.5	5.75	3755.0	1049.40	21.600
4	25.4	5.79	3706.0	888.07	28.500
6	26.3	5.81	4177.0	898.82	20.100
9	26.1	5.40	4039.0	845.48	20.700
10	25.6	5.75	3999.0	711.56	22.800
...	...	...	...	...	...
2130	28.6	5.89	3379.0	496.20	27.000
2131	28.2	5.75	3347.0	683.40	24.000
2132	28.1	5.72	3335.0	524.50	23.977
2133	28.2	5.74	3514.0	649.15	23.218
2134	27.7	5.83	3273.0	497.73	22.541

1272 rows × 5 columns

```
#Save dataframe to excel
df.to_excel ('Central Rand Pump A cleaned for modelling.xlsx')
```

APPENDIX B: PHREEQC INPUT AND OUTPUT FILES (CENTRAL RAND: PUMP A)

AB1: Input File

```
SOLUTION_SPREAD
    -units    mg/l
 Temperature          pH          S(6)             Fe           Mn
                    mg/l charge              mg/l         mg/l
        25.5        5.75          3755         1049.4         21.6
        25.4        5.79          3706         888.07         28.5
        26.3        5.81          4177         898.82         20.1
        26.1         5.4          4039         845.48         20.7
        25.6        5.75          3999         711.56         22.8
          27        5.79          3775         811.08         18.2
        26.1        6.04          3904          838.4         24.4
        25.9        5.87          3706         849.75           23
          26         5.6          3940         1028.5         25.2
        26.3        5.57          3602         796.11         21.3
        26.1        5.64          3438         846.15         23.2
          26        5.91          3357         919.24         25.1
                                     ...
        28.3        5.82          3358         477.19           27
        28.6        5.89          3379          496.2           27
        28.2        5.75          3347          683.4           24
        28.1        5.72          3335          524.5       23.977
        28.2        5.74          3514         649.15       23.218
        27.7        5.83          3273         497.73       22.541

SELECTED_OUTPUT 1
    -file                   Pump A Central Rand Modelling output.xls
    -reset                  false
    -charge_balance         true
    -molalities             SO4-2
```

AB2 : Output File

```
Input file: C:\Users\taske\Documents\PHREEQC\Pump A Central Rand Input.pqi
  Output file: C:\Users\taske\Documents\PHREEQC\Pump A Central Rand
Output.pqo
Database file: C:\Program Files (x86)\USGS\Phreeqc Interactive 3.7.3-
15968\database\phreeqc.dat

------------------
Reading data base.
------------------

      SOLUTION_MASTER_SPECIES
      SOLUTION_SPECIES
      PHASES
      EXCHANGE_MASTER_SPECIES
      EXCHANGE_SPECIES
      SURFACE_MASTER_SPECIES
      SURFACE_SPECIES
      RATES
      END
-----------------------------------
Reading input data for simulation 1.
-----------------------------------
```

```
      DATABASE C:\Program Files (x86)\USGS\Phreeqc Interactive 3.7.3-
15968\database\phreeqc.dat
      SOLUTION_SPREAD
          units    mg/l
       Temperature          pH          S(6)             Fe           Mn
                          mg/l charge          mg/l         mg/l
             25.5          5.75          3755        1049.4         21.6
             25.4          5.79          3706        888.07         28.5
             26.3          5.81          4177        898.82         20.1
             26.1           5.4          4039        845.48         20.7
                                          ...

             28.6          5.89          3379         496.2           27
             28.2          5.75          3347         683.4           24
             28.1          5.72          3335         524.5       23.977
             28.2          5.74          3514        649.15       23.218
             27.7          5.83          3273        497.73       22.541


-------------------------------------------
Beginning of initial solution calculations.
-------------------------------------------

Initial solution 1.

-----------------------------Solution composition------------------------
----

      Elements           Molality       Moles

      Fe                1.888e-02   1.888e-02
      Mn                3.951e-04   3.951e-04
      S(6)              1.928e-02   1.928e-02  Charge balance

----------------------------Description of solution----------------------
----

                                       pH  =   5.750
                                       pe  =   4.000
      Specific Conductance (µS/cm,  26°C)  = 2681
                           Density (g/cm³)  =   0.99979
                                Volume (L)  =   1.00314
                        Activity of water  =   0.999
                  Ionic strength (mol/kgw)  =   5.142e-02
                       Mass of water (kg)  =   1.000e+00
                  Total alkalinity (eq/kg)  =  -1.815e-06
                          Temperature (°C)  =  25.50
                   Electrical balance (eq)  =  -2.042e-14
 Percent error, 100*(Cat-|An|)/(Cat+|An|)  =  -0.00
                                Iterations  =   6
                                   Total H  = 1.110124e+02
                                   Total O  = 5.558333e+01

----------------------------Distribution of species----------------------
----

                                                 Log       Log       Log
mole V
```

Species	Molality	Activity	Molality	Activity	Gamma	cm³/mol
H+	2.086e-06	1.778e-06	-5.681	-5.750	-0.069	0.00
OH-	7.300e-09	5.909e-09	-8.137	-8.228	-0.092	-3.85
H2O	5.551e+01	9.995e-01	1.744	-0.000	0.000	18.07
Fe(2)	1.888e-02					
Fe+2	1.259e-02	6.025e-03	-1.900	-2.220	-0.320	-21.60
FeSO4	6.287e-03	6.362e-03	-2.202	-2.196	0.005	18.45
FeOH+	1.350e-06	1.112e-06	-5.870	-5.954	-0.084	(0)
FeHSO4+	9.092e-08	7.450e-08	-7.041	-7.128	-0.086	(0)
Fe(OH)2	5.488e-12	5.553e-12	-11.261	-11.255	0.005	(0)
Fe(OH)3-	1.415e-16	1.165e-16	-15.849	-15.934	-0.084	(0)
Fe(3)	5.424e-06					
Fe(OH)2+	5.069e-06	4.191e-06	-5.295	-5.378	-0.083	(0)
Fe(OH)3	3.065e-07	3.102e-07	-6.514	-6.508	0.005	(0)
FeOH+2	4.805e-08	2.210e-08	-7.318	-7.656	-0.337	(0)
FeSO4+	4.684e-10	3.858e-10	-9.329	-9.414	-0.084	(0)
Fe(OH)4-	1.962e-10	1.622e-10	-9.707	-9.790	-0.083	(0)
Fe(SO4)2-	6.072e-11	4.976e-11	-10.217	-10.303	-0.086	(0)
Fe+3	2.486e-11	5.914e-12	-10.605	-11.228	-0.624	(0)
Fe2(OH)2+4	3.115e-13	1.288e-14	-12.507	-13.890	-1.384	(0)
FeHSO4+2	4.074e-15	1.837e-15	-14.390	-14.736	-0.346	(0)
Fe3(OH)4+5	1.564e-15	1.077e-17	-14.806	-16.968	-2.162	(0)
H(0)	4.403e-23					
H2	2.201e-23	2.228e-23	-22.657	-22.652	0.005	28.61
Mn(2)	3.951e-04					
Mn+2	2.634e-04	1.261e-04	-3.579	-3.899	-0.320	-18.70
MnSO4	1.316e-04	1.332e-04	-3.881	-3.875	0.005	22.32
MnOH+	2.304e-09	1.897e-09	-8.638	-8.722	-0.084	(0)
Mn(OH)3-	4.309e-22	3.548e-22	-21.366	-21.450	-0.084	(0)
Mn(3)	1.762e-25					
Mn+3	1.762e-25	4.192e-26	-24.754	-25.378	-0.624	(0)
O(0)	0.000e+00					
O2	0.000e+00	0.000e+00	-46.920	-46.915	0.005	30.44

```
S(6)          1.928e-02
   SO4-2           1.285e-02   5.884e-03    -1.891    -2.230    -0.339
15.31
   FeSO4           6.287e-03   6.362e-03    -2.202    -2.196     0.005
18.45
   MnSO4           1.316e-04   1.332e-04    -3.881    -3.875     0.005
22.32
   HSO4-           1.255e-06   1.028e-06    -5.901    -5.988    -0.086
40.49
   FeHSO4+         9.092e-08   7.450e-08    -7.041    -7.128    -0.086
(0)
   FeSO4+          4.684e-10   3.858e-10    -9.329    -9.414    -0.084
(0)
   Fe(SO4)2-       6.072e-11   4.976e-11   -10.217   -10.303    -0.086
(0)
   FeHSO4+2        4.074e-15   1.837e-15   -14.390   -14.736    -0.346
(0)

-------------------------------Saturation indices--------------------------
----

  Phase               SI** log IAP   log K(298 K,   1 atm)

  Fe(OH)3(a)         1.13      6.02     4.89  Fe(OH)3
  Goethite           7.04      6.02    -1.02  FeOOH
  H2(g)            -19.55    -22.65    -3.10  H2
  H2O(g)            -1.49     -0.00     1.49  H2O
  Hausmannite      -18.61     42.30    60.91  Mn3O4
  Hematite          16.09     12.04    -4.05  Fe2O3
  Manganite         -7.99     17.35    25.34  MnOOH
  Melanterite       -2.25     -4.45    -2.20  FeSO4:7H2O
  O2(g)            -44.02    -46.92    -2.90  O2
  Pyrochroite       -7.60      7.60    15.20  Mn(OH)2
  Pyrolusite       -14.20     27.10    41.30  MnO2:H2O

**For a gas, SI = log10(fugacity). Fugacity = pressure * phi / 1 atm.
  For ideal gases, phi = 1.

Initial solution 2.

-------------------------------Solution composition------------------------
----

      Elements           Molality       Moles

      Fe                1.598e-02   1.598e-02
      Mn                5.212e-04   5.212e-04
      S(6)              1.649e-02   1.649e-02  Charge balance

----------------------------Description of solution-------------------------
----

                                       pH  =   5.790
                                       pe  =   4.000
      Specific Conductance (µS/cm,  25°C)  = 2365
                           Density (g/cm³)  =   0.99941
                                Volume (L)  =   1.00310
                        Activity of water  =   1.000
                  Ionic strength (mol/kgw)  =   4.491e-02
                         Mass of water (kg)  =   1.000e+00
                  Total alkalinity (eq/kg)  =  -1.362e-06
```

```
                        Temperature (°C)  =  25.40
                  Electrical balance (eq)  =  -3.175e-15
 Percent error, 100*(Cat-|An|)/(Cat+|An|)  =  -0.00
                               Iterations  =   6 (12 overall)
                                  Total H  = 1.110124e+02
                                  Total O  = 5.557221e+01

----------------------------Distribution of species-------------------------
----

                                                    Log       Log       Log
mole V
   Species          Molality    Activity  Molality  Activity     Gamma
cm³/mol

   H+              1.890e-06   1.622e-06    -5.724    -5.790    -0.066
0.00
   OH-             7.857e-09   6.431e-09    -8.105    -8.192    -0.087
-3.87
   H2O             5.551e+01   9.995e-01     1.744    -0.000     0.000
18.07
Fe(2)         1.597e-02
   Fe+2            1.087e-02   5.384e-03    -1.964    -2.269    -0.305    -
21.64
   FeSO4           5.098e-03   5.151e-03    -2.293    -2.288     0.004
18.55
   FeOH+           1.300e-06   1.081e-06    -5.886    -5.966    -0.080
(0)
   FeHSO4+         6.648e-08   5.499e-08    -7.177    -7.260    -0.082
(0)
   Fe(OH)2         5.811e-12   5.871e-12   -11.236   -11.231     0.004
(0)
   Fe(OH)3-        1.624e-16   1.350e-16   -15.790   -15.870    -0.080
(0)
Fe(3)         5.716e-06
   Fe(OH)2+        5.316e-06   4.436e-06    -5.274    -5.353    -0.079
(0)
   Fe(OH)3         3.547e-07   3.584e-07    -6.450    -6.446     0.004
(0)
   FeOH+2          4.483e-08   2.141e-08    -7.348    -7.669    -0.321
(0)
   FeSO4+          3.735e-10   3.105e-10    -9.428    -9.508    -0.080
(0)
   Fe(OH)4-        2.453e-10   2.047e-10    -9.610    -9.689    -0.079
(0)
   Fe(SO4)2-       4.393e-11   3.634e-11   -10.357   -10.440    -0.082
(0)
   Fe+3            2.085e-11   5.255e-12   -10.681   -11.279    -0.598
(0)
   Fe2(OH)2+4      2.523e-13   1.214e-14   -12.598   -13.916    -1.318
(0)
   FeHSO4+2        2.879e-15   1.348e-15   -14.541   -14.870    -0.329
(0)
   Fe3(OH)4+5      1.243e-15   1.084e-17   -14.906   -16.965    -2.059
(0)
H(0)          3.671e-23
   H2              1.836e-23   1.855e-23   -22.736   -22.732     0.004
28.61
Mn(2)         5.212e-04
   Mn+2            3.548e-04   1.757e-04    -3.450    -3.755    -0.305    -
18.80
```

```
   MnSO4           1.664e-04   1.682e-04    -3.779    -3.774     0.004
22.37
   MnOH+           3.459e-09   2.875e-09    -8.461    -8.541    -0.080
(0)
   Mn(OH)3-        7.841e-22   6.518e-22   -21.106   -21.186    -0.080
(0)
Mn(3)          2.283e-25
   Mn+3            2.283e-25   5.756e-26   -24.641   -25.240    -0.598
(0)
O(0)           0.000e+00
   O2              0.000e+00   0.000e+00   -46.793   -46.788     0.004
30.43
S(6)           1.649e-02
   SO4-2           1.123e-02   5.341e-03    -1.950    -2.272    -0.323
15.25
   FeSO4           5.098e-03   5.151e-03    -2.293    -2.288     0.004
18.55
   MnSO4           1.664e-04   1.682e-04    -3.779    -3.774     0.004
22.37
   HSO4-           1.027e-06   8.496e-07    -5.988    -6.071    -0.082
40.47
   FeHSO4+         6.648e-08   5.499e-08    -7.177    -7.260    -0.082
(0)
   FeSO4+          3.735e-10   3.105e-10    -9.428    -9.508    -0.080
(0)
   Fe(SO4)2-       4.393e-11   3.634e-11   -10.357   -10.440    -0.082
(0)
   FeHSO4+2        2.879e-15   1.348e-15   -14.541   -14.870    -0.329
(0)

-------------------------------Saturation indices--------------------------
----

  Phase               SI** log IAP   log K(298 K,   1 atm)

  Fe(OH)3(a)         1.20      6.09     4.89  Fe(OH)3
  Goethite           7.10      6.09    -1.01  FeOOH
  H2(g)            -19.63    -22.73    -3.10  H2
  H2O(g)            -1.49     -0.00     1.49  H2O
  Hausmannite      -17.88     43.05    60.93  Mn3O4
  Hematite          16.22     12.18    -4.04  Fe2O3
  Manganite         -7.73     17.61    25.34  MnOOH
  Melanterite       -2.34     -4.54    -2.20  FeSO4:7H2O
  O2(g)            -43.89    -46.79    -2.90  O2
  Pyrochroite       -7.38      7.82    15.20  Mn(OH)2
  Pyrolusite       -13.91     27.40    41.32  MnO2:H2O

**For a gas, SI = log10(fugacity). Fugacity = pressure * phi / 1 atm.
  For ideal gases, phi = 1.


...

Initial solution 1272.

-----------------------------Solution composition------------------------
----

      Elements           Molality       Moles
```

```
      Fe                8.946e-03   8.946e-03
      Mn                4.119e-04   4.119e-04
      S(6)              9.355e-03   9.355e-03  Charge balance

----------------------------Description of solution----------------------
----

                                       pH  =   5.830
                                       pe  =   4.000
      Specific Conductance (µS/cm,  28°C)  = 1564
                           Density (g/cm³)  =   0.99773
                                Volume (L)  =   1.00370
                        Activity of water  =   1.000
                  Ionic strength (mol/kgw)  =   2.714e-02
                       Mass of water (kg)  =   1.000e+00
                  Total alkalinity (eq/kg)  =  -8.057e-07
                          Temperature (°C)  =  27.70
                   Electrical balance (eq)  =   5.612e-11
 Percent error, 100*(Cat-|An|)/(Cat+|An|)  =   0.00
                                Iterations  =   5 (6669 overall)
                                   Total H  = 1.110125e+02
                                   Total O  = 5.554365e+01

----------------------------Distribution of species----------------------
----

                                                 Log       Log       Log
mole V
   Species          Molality    Activity  Molality  Activity     Gamma
cm³/mol

   H+             1.686e-06   1.479e-06    -5.773    -5.830    -0.057
0.00
   OH-            9.870e-09   8.382e-09    -8.006    -8.077    -0.071
-3.86
   H2O            5.551e+01   9.997e-01     1.744    -0.000     0.000
18.08
Fe(2)         8.940e-03
   Fe+2           6.483e-03   3.605e-03    -2.188    -2.443    -0.255    -
21.60
   FeSO4          2.456e-03   2.472e-03    -2.610    -2.607     0.003
16.25
   FeOH+          1.097e-06   9.411e-07    -5.960    -6.026    -0.066
(0)
   FeHSO4+        2.842e-08   2.429e-08    -7.546    -7.615    -0.068
(0)
   Fe(OH)2        6.790e-12   6.833e-12   -11.168   -11.165     0.003
(0)
   Fe(OH)3-       2.053e-16   1.762e-16   -15.688   -15.754    -0.066
(0)
Fe(3)         6.392e-06
   Fe(OH)2+       5.864e-06   5.045e-06    -5.232    -5.297    -0.065
(0)
   Fe(OH)3        4.905e-07   4.936e-07    -6.309    -6.307     0.003
(0)
   FeOH+2         3.755e-08   2.037e-08    -7.425    -7.691    -0.266
(0)
   Fe(OH)4-       3.938e-10   3.388e-10    -9.405    -9.470    -0.065
(0)
   FeSO4+         1.984e-10   1.703e-10    -9.702    -9.769    -0.066
(0)
```

```
   Fe(SO4)2-       1.617e-11   1.382e-11   -10.791   -10.860    -0.068
(0)
   Fe+3            1.292e-11   3.987e-12   -10.889   -11.399    -0.511
(0)
   Fe2(OH)2+4      1.238e-13   9.995e-15   -12.907   -14.000    -1.093
(0)
   FeHSO4+2        1.265e-15   6.746e-16   -14.898   -15.171    -0.273
(0)
   Fe3(OH)4+5      4.197e-16   8.231e-18   -15.377   -17.085    -1.708
(0)
H(0)          2.997e-23
   H2              1.499e-23   1.508e-23   -22.824   -22.822     0.003
28.60
Mn(2)         4.119e-04
   Mn+2            2.985e-04   1.660e-04    -3.525    -3.780    -0.255    -
19.15
   MnSO4           1.133e-04   1.141e-04    -3.946    -3.943     0.003
21.39
   MnOH+           4.180e-09   3.587e-09    -8.379    -8.445    -0.066
(0)
   Mn(OH)3-        9.467e-22   8.124e-22   -21.024   -21.090    -0.066
(0)
Mn(3)         2.458e-25
   Mn+3            2.458e-25   7.583e-26   -24.609   -25.120    -0.511
(0)
O(0)          0.000e+00
   O2              0.000e+00   0.000e+00   -45.876   -45.874     0.003
30.61
S(6)          9.355e-03
   SO4-2           6.785e-03   3.671e-03    -2.168    -2.435    -0.267
15.34
   FeSO4           2.456e-03   2.472e-03    -2.610    -2.607     0.003
16.25
   MnSO4           1.133e-04   1.141e-04    -3.946    -3.943     0.003
21.39
   HSO4-           6.558e-07   5.603e-07    -6.183    -6.252    -0.068
40.57
   FeHSO4+         2.842e-08   2.429e-08    -7.546    -7.615    -0.068
(0)
   FeSO4+          1.984e-10   1.703e-10    -9.702    -9.769    -0.066
(0)
   Fe(SO4)2-       1.617e-11   1.382e-11   -10.791   -10.860    -0.068
(0)
   FeHSO4+2        1.265e-15   6.746e-16   -14.898   -15.171    -0.273
(0)

-----------------------------Saturation indices--------------------------
----

  Phase               SI** log IAP   log K(300 K,   1 atm)

  Fe(OH)3(a)         1.20      6.09     4.89  Fe(OH)3
  Goethite           7.19      6.09    -1.10  FeOOH
  H2(g)           -19.71    -22.82    -3.11  H2
  H2O(g)           -1.43     -0.00     1.43  H2O
  Hausmannite     -17.07     43.30    60.37  Mn3O4
  Hematite         16.39     12.18    -4.21  Fe2O3
  Manganite        -7.63     17.71    25.34  MnOOH
  Melanterite      -2.70     -4.88    -2.18  FeSO4:7H2O
  O2(g)           -42.96    -45.87    -2.91  O2
  Pyrochroite      -7.32      7.88    15.20  Mn(OH)2
```

```
  Pyrolusite      -13.41     27.54   40.95  MnO2:H2O

**For a gas, SI = log10(fugacity). Fugacity = pressure * phi / 1 atm.
  For ideal gases, phi = 1.

------------------
End of simulation.
------------------

-----------------------------------
Reading input data for simulation 2.
-----------------------------------

-------------------------------
End of Run after 5.464 Seconds.
-------------------------------
```

APPENDIX C: DATA CLEANING AND VISUALIZATION PYTHON CODE (CENTRAL RAND: PUMP A)

```
# CENTRAL RAND PUMP A REGRESSION (with outlier removal)

# Load the libraries and packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk

# Data for Pump A for Central Rand was pre-cleaned and modelled using
PHREEQC
# Values to be used are Temperature (TEMP), Electrical
Conductivity(EC), pH, Total Dissolved Solids (TDS), Sulphate (SUL),
Iron (FE) and Manganese (MN)

#Load dataset from excel. Note the read_excel() function

df = pd.read_excel('/content/Pump A - Central Rand (Cleaned and
modelled).xlsx')

#Now let's analyze the data using some Descriptive Statistics

#shape
print('The shape is:',df.shape)

#types
print('The types of data in each column are:', df.dtypes)
```

```
The shape is: (1245, 7)
The types of data in each column are: A - TEMP    float64
A-EC          float64
A-TDS         float64
A-pH          float64
A-SUL         float64
A-FE          float64
A-MN          float64
dtype: object
```

```
# All data is floats and we know there are no NaN values since we
modelled the data in PHREEQC already
# Data is squeaky clean and ready !

# Let's view the data to get an overall understanding
df
```

	A - TEMP	A-EC	A-TDS	A-pH	A-SUL	A-FE	A-MN
0	25.5	551.0	3040.00	5.75	0.012855	1049.40	21.600
1	25.4	548.0	3.07	5.79	0.011229	888.07	28.500
2	26.3	538.0	2940.00	5.81	0.011212	898.82	20.100
3	26.1	535.0	2900.00	5.40	0.010660	845.48	20.700
4	25.6	555.0	2990.00	5.75	0.009250	711.56	22.800
...	...	...	...	...	...	...	...
1240	28.6	470.0	5734.00	5.89	0.007910	496.20	27.000
1241	28.2	476.0	5100.00	5.75	0.007049	683.40	24.000
1242	28.1	478.0	5458.00	5.72	0.007683	524.50	23.977
1243	28.2	474.0	4730.00	5.74	0.007110	649.15	23.218
1244	27.7	474.0	5584.00	5.83	0.007228	497.73	22.541

1245 rows × 7 columns

```
# Notice : No null values nor NaN (yay !)

#Let's see the descriptive statistics for each attribute

df.describe()
```

	A - TEMP	A-EC	A-TDS	A-pH	A-SUL	A-FE	A-MN
count	1245.000000	1245.000000	1245.000000	1245.000000	1245.000000	1245.000000	1245.000000
mean	25.917992	511.665863	5960.115317	9.817116	0.008959	674.316817	27.430765
std	1.406147	25.151838	525.178432	143.100601	0.001453	122.965911	2.584595
min	20.000000	416.000000	2.970000	5.090000	0.000877	29.000000	3.549000
25%	25.100000	492.000000	5646.000000	5.710000	0.007767	572.530000	25.719000
50%	26.100000	508.000000	6020.000000	5.780000	0.008717	649.990000	27.735000
75%	26.900000	528.000000	6259.000000	5.830000	0.010107	774.040000	29.397000
max	33.200000	604.000000	7177.000000	5055.000000	0.014954	1130.900000	34.034000

```
#Let's see the correlation between numerical attributes
df.corr()
```

	A - TEMP	A-EC	A-TDS	A-pH	A-SUL	A-FE	A-MN
A - TEMP	1.000000	-0.350065	-0.300798	0.019755	-0.452614	-0.490340	-0.407613
A-EC	-0.350065	1.000000	0.414208	-0.046201	0.715709	0.733054	0.500631
A-TDS	-0.300798	0.414208	1.000000	-0.019319	0.439682	0.417122	0.421508
A-pH	0.019755	-0.046201	-0.019319	1.000000	-0.026605	-0.020364	-0.053243
A-SUL	-0.452614	0.715709	0.439682	-0.026605	1.000000	0.846070	0.546295
A-FE	-0.490340	0.733054	0.417122	-0.020364	0.846070	1.000000	0.595845
A-MN	-0.407613	0.500631	0.421508	-0.053243	0.546295	0.595845	1.000000

```
# Let's view the distributions of the data as is, we will do
normalization afterwards to get rid of the differeing scales and then
re-visualize

# DATA VISUALIZATION

# Unimodel data Visualization - individual attributes

# histograms

df.hist(figsize = (25,25) , grid = False, edgecolor = "k")
plt.tight_layout()
plt.show()

# Now lets look at some density plots

df.plot(kind='density', figsize = (20,20), subplots=True, layout
=(3,3), sharex=False)
plt.tight_layout()
plt.show()

# Now let's see some Box and Whisker plots

df.plot(kind='box', figsize = (20,20), subplots=True , layout=(3,3) ,
sharex=False)
plt.tight_layout()
plt.show()

# Notice we have a decent amount of outliers, especially in pH we have
a very high outlier

# Let's remove all outliers

# Define a function to detect and remove outliers using the IQR (Inter
Quartile Range) method
def drop_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - (1.5 * IQR)
    upper_bound = Q3 + (1.5 * IQR)
    return column[(column >= lower_bound) & (column <= upper_bound)]

# Loop through each column and drop the rows that contain the outliers
for col in df.columns:
    df = df.loc[drop_outliers(df[col]).index]

df
```

	A - TEMP	A-EC	A-TDS	A-pH	A-SUL	A-FE	A-MN
9	26.3	564.0	6711.0	5.57	0.010122	796.11	21.300
10	26.1	564.0	6896.0	5.64	0.010688	846.15	23.200
11	26.0	570.0	6729.0	5.91	0.011489	919.24	25.100
15	24.4	519.0	6652.0	5.67	0.010489	820.28	23.400
16	25.7	530.0	6731.0	5.63	0.010830	853.69	26.900
...	...	...	...	...	...	...	...
1240	28.6	470.0	5734.0	5.89	0.007910	496.20	27.000
1241	28.2	476.0	5100.0	5.75	0.007049	683.40	24.000
1242	28.1	478.0	5458.0	5.72	0.007683	524.50	23.977
1243	28.2	474.0	4730.0	5.74	0.007110	649.15	23.218
1244	27.7	474.0	5584.0	5.83	0.007228	497.73	22.541

1107 rows × 7 columns

```
#Notice the shape changed to 1107 rows

# Now let's see some Box and Whisker plots

df.plot(kind='box', figsize = (20,20), subplots=True , layout=(3,3) ,
sharex=False)
plt.tight_layout()
plt.show()

# Much better let's carry on with visualizing from the start

# RE-Visualize the data without any Normalizations 1st

# DATA VISUALIZATION

# Unimodel data Visualization - individual attributes

# histograms

df.hist(figsize = (25,25) , grid = False, edgecolor = "k")
plt.tight_layout()
plt.show()

# Now lets look at some density plots

df.plot(kind='density', figsize = (20,20), subplots=True, layout
=(3,3), sharex=False)
plt.tight_layout()
plt.show()

# Now let's see some Box and Whisker plots
```

```
df.plot(kind='box', figsize = (20,20), subplots=True , layout=(3,3) ,
sharex=False)
plt.tight_layout()
plt.show()

# MULTIMODAL DATA VISUALIZATIONS

#scatter plot matrix
from pandas.plotting import scatter_matrix

scatter_matrix(df,figsize = (25,25),edgecolor = "k")
plt.show()

# Now lets see the correlation matrix
# Using seaborn

import seaborn as sns

plt.figure(figsize=(25,25))
sns.heatmap(df.corr(), annot=True)

# NORMALIZATION

# Now let's scale the data since they each have different units. So for
regression we need the same scale.
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

# transform data
scaled = scaler.fit_transform(df)

# Replace the existing dataframe with the new one
df = pd.DataFrame(scaled)

# Ensure the old headings remain
df.set_axis(['A-TEMP','A-EC','A-TDS','A-pH','A-SUL','A-FE','A-MN'],
axis='columns', inplace=True),

# REVISUALIZE NORMALIZED DATA

# DATA VISUALIZATION

# Unimodel data Visualization - individual attributes

# histograms

df.hist(figsize = (25,25) , grid = False, edgecolor = "k")
plt.tight_layout()
```

```
plt.show()

# Now lets look at some density plots

df.plot(kind='density', figsize = (20,20), subplots=True, layout
=(3,3), sharex=False)
plt.tight_layout()
plt.show()

# Now let's see some Box and Whisker plots

df.plot(kind='box', figsize = (20,20), subplots=True , layout=(3,3) ,
sharex=False)
plt.tight_layout()
plt.show()

# MULTIMODAL DATA VISUALIZATIONS

#scatter plot matrix
from pandas.plotting import scatter_matrix

scatter_matrix(df,figsize = (25,25),edgecolor = "k")
plt.show()

# Now lets see the correlation matrix
# Using seaborn

import seaborn as sns

plt.figure(figsize=(25,25))
sns.heatmap(df.corr(), annot=True)

# FEATURE SELECTION
# Feature Extraction with PCA

from sklearn.decomposition import PCA

pca = PCA()
pca.fit(df)
exp_var_cumul = np.cumsum(pca.explained_variance_ratio_)

import plotly.express as px
px.area(
    x=range(1, exp_var_cumul.shape[0] + 1),
    y=exp_var_cumul,
    labels={"x": "Number of Components", "y": "Explained Variance"}
)
# Let's visualize the PCA data using a Scree Plot
```

```
PC_values = np.arange(pca.n_components_) + 1
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2,
color='blue')
plt.title('Scree Plot - Pump A Central Rand')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()

# From the above data we can clearly see we actually only need 2
componets to explain 78% of our data

# Let's use RFE to find those components

# Feature Extraction with RFE


from sklearn.feature_selection import RFE
from sklearn.linear_model import LinearRegression # Must use a
regression model

# Create predictor matrix and target
dataframe = df
array = dataframe.values
X = array[:, df.columns != 'A-SUL']
Y = array[:, df.columns == 'A-SUL'].flatten()

# feature extraction
model = LinearRegression()
rfe = RFE(model,  n_features_to_select=2)
fit = rfe.fit(X, Y)
print("Number of Features (i.e. Principal Components): %d" %
fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Number of Features (i.e. Principal Components): 2
Selected Features: [False False  True False  True False]
Feature Ranking: [4 3 1 5 1 2]
```

```
# The main features are the ones with True

# These columns are A-TDS and A-FE

# (remember A-SUL is the target)

# Great ! Now we have the name of the most important columns, let's
create a new dataframe using these two components and the target A-SUL
```

```
# Replace the existing dataframe with the new one
df = df.loc[:, ['A-TDS','A-FE','A-SUL']]

# Ensure the old headings remain
df.set_axis(['A-TDS','A-FE','A-SUL'], axis='columns', inplace=True)

df
```

	A-TDS	A-FE	A-SUL
0	0.853511	0.655717	0.582293
1	0.933218	0.743195	0.661206
2	0.861267	0.870968	0.772883
3	0.828091	0.697970	0.633461
4	0.862128	0.756376	0.681004
...	...	...	...
1102	0.432572	0.131427	0.273893
1103	0.159414	0.458682	0.153796
1104	0.313658	0.180900	0.242273
1105	0.000000	0.398808	0.162384
1106	0.367945	0.134101	0.178864

1107 rows × 3 columns

APPENDIX D: MACHINE LEARNING – INDIVIDUAL REGRESSION MODELS AND SE-ML PYTHON CODE (CENTRAL RAND: PUMP A)

```
# EVALUATE ALGORITHMS

from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import ElasticNet
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor

# a) Split-out test dataset

array = df.values
X = array[:,0:2] #remember, the last column is 2 not 3 cause of how
Python counts
y = array[:,2]

X_train, X_test, y_train, y_test = train_test_split (X , y , test_size
= 0.2 , random_state = 7)

# b) Evaluate baseline models and their performance metrics

# Test options and evaluation metric

num_folds = 10 # 10 fold cross validation
seed = 7
scoring = 'neg_mean_squared_error'  #MSE in descending order . Lowest
MSE (ie. one closest to zero is best)

# Now lets spot-check a couple of linear,non-linear and esemble (RF is
bagged tree, XGBOOST is boosted tree) algorithms to find the best one

models = []
models.append(('LR', LinearRegression()))
models.append(('RD', Ridge()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNNR', KNeighborsRegressor()))
```

```
models.append(('DT', DecisionTreeRegressor()))
models.append(('SVR', SVR()))
models.append(('XG', XGBRegressor()))
models.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
models.append(('MLP', MLPRegressor()))

# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=num_folds)  #Setting a random_state has no
effect since shuffle is False. You should leave random_state to its
default (None), or set shuffle=True.
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
```

```
LR: -0.006047 (0.001000)
RD: -0.006064 (0.001062)
LASSO: -0.032166 (0.005089)
EN: -0.032166 (0.005089)
KNNR: -0.006151 (0.001026)
DT: -0.010715 (0.001448)
SVR: -0.005318 (0.000763)
XG: -0.007309 (0.001230)
RF: -0.006476 (0.000935)
MLP: -0.005971 (0.001033)
```

```
# Let's visualize the above mean (std dev) using box and whisker plots

# Compare Algorithms
fig = plt.figure()
fig.suptitle('Baseline Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.xlabel('Baseline Models')
plt.ylabel('Negative Mean Squared Error (MSE)')
plt.show()

# So far SVR looks to be the best

# STACKING ENSEMBLE TO COMBINE ALL THESE MODELS
from sklearn.ensemble import StackingRegressor

# get a stacking ensemble of models
```

```
def get_stacking():

 # define the base models
 # Let's try all models first
 level0 = list()
 level0.append(('LR', LinearRegression()))
 level0.append(('RD', Ridge()))
 level0.append(('LASSO', Lasso()))
 level0.append(('EN', ElasticNet()))
 level0.append(('KNNR', KNeighborsRegressor()))
 level0.append(('DT', DecisionTreeRegressor()))
 level0.append(('SVR', SVR()))
 level0.append(('XG', XGBRegressor()))
 level0.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
 level0.append(('MLP', MLPRegressor()))

 # define meta learner model
 level1 = LinearRegression()

 # define the stacking ensemble
 model = StackingRegressor(estimators=level0, final_estimator=level1,
cv=10)

 return model

# Let's evaluate the Stacked ensemble with the baseline models

models = []
models.append(('LR', LinearRegression()))
models.append(('RD', Ridge()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNNR', KNeighborsRegressor()))
models.append(('DT', DecisionTreeRegressor()))
models.append(('SVR', SVR()))
models.append(('XG', XGBRegressor()))
models.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
models.append(('MLP', MLPRegressor()))
models.append(('Stacked',get_stacking()))

# evaluate each model in turn
results = []
names = []
for name, model in models:
```

```
    kfold = KFold(n_splits=num_folds)  #Setting a random_state has no
effect since shuffle is False. You should leave random_state to its
default (None), or set shuffle=True.
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

# Let's visualize the above mean (std dev) using box and whisker plots

# Compare Algorithms
fig = plt.figure()
fig.suptitle('Baseline Algorithm and Stacked Ensemble Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.xlabel('Models')
plt.ylabel('Negative Mean Squared Error (MSE)')
plt.show()
```

```
LR: -0.006047 (0.001000)
RD: -0.006064 (0.001062)
LASSO: -0.032166 (0.005089)
EN: -0.032166 (0.005089)
KNNR: -0.006151 (0.001026)
DT: -0.010831 (0.001547)
SVR: -0.005318 (0.000763)
XG: -0.007309 (0.001230)
RF: -0.006476 (0.000935)
MLP: -0.006423 (0.001115)
Stacked: -0.005436 (0.000814)
```

```
# SVR performs the best !

# STACKING ENSEMBLE USING ONLY GOOD MODELS

from sklearn.ensemble import StackingRegressor

# get a stacking ensemble of models

def get_stacking():

 # define the base models
 # Let's try ONLY THE BEST MODELS NOW, remove LASSO and EN
 level0 = list()
 level0.append(('LR', LinearRegression()))
 level0.append(('RD', Ridge()))
 level0.append(('KNNR', KNeighborsRegressor()))
```

```
 level0.append(('DT', DecisionTreeRegressor()))
 level0.append(('SVR', SVR()))
 level0.append(('XG', XGBRegressor()))
 level0.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
 level0.append(('MLP', MLPRegressor()))

 # define meta learner model
 level1 = LinearRegression()

 # define the stacking ensemble
 model = StackingRegressor(estimators=level0, final_estimator=level1,
cv=10)

 return model
# Let's evaluate the Stacked ensemble with the good baseline models

models = []
models.append(('LR', LinearRegression()))
models.append(('RD', Ridge()))
models.append(('KNNR', KNeighborsRegressor()))
models.append(('DT', DecisionTreeRegressor()))
models.append(('SVR', SVR()))
models.append(('XG', XGBRegressor()))
models.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
models.append(('MLP', MLPRegressor()))
models.append(('Stacked',get_stacking()))

# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=num_folds)  #Setting a random_state has no
effect since shuffle is False. You should leave random_state to its
default (None), or set shuffle=True.
    cv_results = cross_val_score(model, X_train, y_train, cv=kfold,
scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

# Let's visualize the above mean (std dev) using box and whisker plots

# Compare Algorithms
fig = plt.figure()
```

```
fig.suptitle('Baseline Algorithm and Stacked Ensemble Comparison (Only
Good Models)')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.xlabel('Models')
plt.ylabel('Negative Mean Squared Error (MSE)')
plt.show()
```

```
LR: -0.006047 (0.001000)
RD: -0.006064 (0.001062)
KNNR: -0.006151 (0.001026)
DT: -0.010816 (0.001389)
SVR: -0.005318 (0.000763)
XG: -0.007309 (0.001230)
RF: -0.006476 (0.000935)
MLP: -0.007165 (0.003117)
Stacked: -0.005458 (0.000858)
# SVR still performs the best!
```

APPENDIX E: MODEL TESTING PYTHON CODE (CENTRAL RAND: PUMP A)

```
# FINALIZE THE MODEL

#First train the selected model
model = SVR()

model.fit(X_train,y_train)

#Now test the model on the test dataset
predictions = model.predict(X_test)

#Now find the accuracy
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score


print('MSE:',mean_squared_error(y_test,predictions))
print('MAE:',mean_absolute_error(y_test,predictions))
print('R squared:', r2_score(y_test,predictions))
MSE: 0.005274958489549416
MAE: 0.055934964525850756
R squared: 0.8413513701129478
```

```
# Now test for all the models

models = []
models.append(('LR', LinearRegression()))
models.append(('RD', Ridge()))
models.append(('LASSO', Lasso()))
models.append(('EN', ElasticNet()))
models.append(('KNNR', KNeighborsRegressor()))
models.append(('DT', DecisionTreeRegressor()))
models.append(('SVR', SVR()))
models.append(('XG', XGBRegressor()))
models.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
models.append(('MLP', MLPRegressor()))
models.append(('Stacked',get_stacking()))

# But still keep stacking for only the good ones
from sklearn.ensemble import StackingRegressor
def get_stacking():
 # Let's try ONLY THE BEST MODELS NOW, remove LASSO and EN
 level0 = list()
 level0.append(('LR', LinearRegression()))
 level0.append(('RD', Ridge()))
```

```
 level0.append(('KNNR', KNeighborsRegressor()))
 level0.append(('DT', DecisionTreeRegressor()))
 level0.append(('SVR', SVR()))
 level0.append(('XG', XGBRegressor()))
 level0.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
 level0.append(('MLP', MLPRegressor()))
 level1 = LinearRegression()
 model = StackingRegressor(estimators=level0, final_estimator=level1,
cv=10)
 return model

print ('The following are the evaluation metrics MSE,MAE and R2:')

MSE = []
MAE = []
R2 = []
names = []
for name, model in models:
    model.fit(X_train,y_train)
    predictions = model.predict(X_test)
    names.append(name)
    msg = "%s: %f %f %f" % (name,
mean_squared_error(y_test,predictions),
mean_absolute_error(y_test,predictions),r2_score(y_test,predictions))
    print(msg)
    MSE.append(mean_squared_error(y_test,predictions))
    MAE.append(mean_absolute_error(y_test,predictions))
    R2.append(r2_score(y_test,predictions))
```

```
The following are the evaluation metrics MSE,MAE and R2:
LR: 0.006296 0.060355 0.810643
RD: 0.006289 0.060255 0.810865
LASSO: 0.033252 0.153985 -0.000069
EN: 0.033252 0.153985 -0.000069
KNNR: 0.006372 0.061412 0.808349
DT: 0.012552 0.089095 0.622486
SVR: 0.005275 0.055935 0.841351
XG: 0.007042 0.065316 0.788193
RF: 0.006457 0.061755 0.805815
MLP: 0.006700 0.061866 0.798483
Stacked: 0.005252 0.055598 0.842041
```

```
# Visualize this using a multi bar graph with a legend

x_axis = np.arange(len(models))

plt.bar(x_axis -0.2, MSE, width=0.4, label = 'MSE')
plt.bar(x_axis +0.2, MAE, width=0.4, label = 'MAE')

plt.xticks(x_axis, names)
```

```
plt.legend()
plt.xlabel('Models')
plt.ylabel('Metric Value')
plt.suptitle('Model Evaluation by Metric Comparison')
plt.show()

# Now visualize the R2 values

plt.bar( x_axis, R2, width=0.4, label = 'R2', color = 'green')

plt.xticks(x_axis, names)
plt.xlabel('Models')
plt.ylabel('R2 Value')
plt.suptitle('Model Evaluation by R2 Comparison')
plt.show()
# Now test for all the models besides LASSO and EN

models = []
models.append(('LR', LinearRegression()))
models.append(('RD', Ridge()))
models.append(('KNNR', KNeighborsRegressor()))
models.append(('DT', DecisionTreeRegressor()))
models.append(('SVR', SVR()))
models.append(('XG', XGBRegressor()))
models.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
models.append(('MLP', MLPRegressor()))
models.append(('Stacked',get_stacking()))

# But still keep stacking for only the good ones
from sklearn.ensemble import StackingRegressor
def get_stacking():
 # Let's try ONLY THE BEST MODELS NOW, remove LASSO and EN
 level0 = list()
 level0.append(('LR', LinearRegression()))
 level0.append(('RD', Ridge()))
 level0.append(('KNNR', KNeighborsRegressor()))
 level0.append(('DT', DecisionTreeRegressor()))
 level0.append(('SVR', SVR()))
 level0.append(('XG', XGBRegressor()))
 level0.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
 level0.append(('MLP', MLPRegressor()))
 level1 = LinearRegression()
 model = StackingRegressor(estimators=level0, final_estimator=level1,
cv=10)
 return model
```

```
print ('The following are the evaluation metrics MSE,MAE and R2:')

MSE = []
MAE = []
R2 = []
names = []
for name, model in models:
    model.fit(X_train,y_train)
    predictions = model.predict(X_test)
    names.append(name)
    msg = "%s: %f %f %f" % (name,
mean_squared_error(y_test,predictions),
mean_absolute_error(y_test,predictions),r2_score(y_test,predictions))
    print(msg)
    MSE.append(mean_squared_error(y_test,predictions))
    MAE.append(mean_absolute_error(y_test,predictions))
    R2.append(r2_score(y_test,predictions))
# Visualize this using a multi bar graph with a legend

x_axis = np.arange(len(models))

plt.bar(x_axis -0.2, MSE, width=0.4, label = 'MSE')
plt.bar(x_axis +0.2, MAE, width=0.4, label = 'MAE')

plt.xticks(x_axis, names)
plt.legend()
plt.xlabel('Models')
plt.ylabel('Metric Value')
plt.suptitle('Model Evaluation by Metric Comparison')
plt.show()
# Now visualize the R2 values

plt.bar( x_axis, R2, width=0.4, label = 'R2', color = 'green')

plt.xticks(x_axis, names)
plt.xlabel('Models')
plt.ylabel('R2 Value')
plt.suptitle('Model Evaluation by R2 Comparison')
plt.show()
```

```
The following are the evaluation metrics MSE,MAE and R2:
LR: 0.006296 0.060355 0.810643
RD: 0.006289 0.060255 0.810865
KNNR: 0.006372 0.061412 0.808349
DT: 0.012215 0.087321 0.632631
SVR: 0.005275 0.055935 0.841351
XG: 0.007042 0.065316 0.788193
RF: 0.006457 0.061755 0.805815
MLP: 0.006224 0.060069 0.812801
Stacked: 0.005335 0.055679 0.839538
```

APPENDIX F : KGR INFLOW (WEST RAND) PYTHON CLEANING CODE

```
# WEST RAND - KGR INFLOW (UNTREATED AMD)

# Load the libraries and packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk


#Load dataset from excel. Note the read_excel() function

df = pd.read_excel('/content/West Rand - KGR inflow (Raw).xlsx')

df
```

	TEMP - KGR inflow	FE - KGR inflow	SUL - KGR inflow
0	19.6	1.6	NaN
1	19.2	0.8	NaN
2	19.7	0.2	NaN
3	18	4	NaN
4	19	9	NaN
...	...	...	...
2463	21.5	0.19	NaN
2464	NaN	NaN	NaN
2465	NaN	NaN	NaN
2466	NaN	NaN	NaN
2467	22.5	0.19	NaN

2468 rows × 3 columns

```
# drop the sul column since it is our target
df.drop(['SUL - KGR inflow'], axis=1, inplace=True)

df
```

	TEMP - KGR inflow	FE - KGR inflow
0	19.6	1.6
1	19.2	0.8
2	19.7	0.2
3	18	4
4	19	9
...	...	...
2463	21.5	0.19
2464	NaN	NaN
2465	NaN	NaN
2466	NaN	NaN
2467	22.5	0.19

2468 rows × 2 columns

```
# DATA CLEANING

# Convert all empty spaces to NaN
df = df.replace(r'^\s*$', np.nan, regex=True)

#Convert words to NaN
df = df.replace('No Electricity', np.nan, regex=True)
df = df.replace('Electr. Off', np.nan, regex=True)
df = df.replace('No Sample', np.nan, regex=True)


# Drop all NaN values
df = df.dropna()

# Convert all columns to floats
df = df.astype(float)

df
```

	TEMP - KGR inflow	FE - KGR inflow
0	19.6	1.60
1	19.2	0.80
2	19.7	0.20
3	18.0	4.00
4	19.0	9.00
...	...	...
2457	19.9	0.21
2460	21.3	0.19
2461	21.6	0.19
2463	21.5	0.19
2467	22.5	0.19

1478 rows × 2 columns

```
# Now save this to a new Excel file

df.to_excel ('West Rand - KGR inflow cleaned.xlsx')
```

APPENDIX G : KGR INFLOW (WEST RAND) PYTHON DATA CLEANING AND VISUALIZATION CODE

```
# West Rand - KGR inflow visualization and cleaning for TL

# Load the libraries and packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk

#Load dataset from excel. Note the read_excel() function

df = pd.read_excel('/content/West Rand - KGR inflow cleaned.xlsx')

#Now let's analyze the data using some Descriptive Statistics

#shape
print('The shape is:',df.shape)

#types
print('The types of data in each column are:', df.dtypes)
```

```
The shape is: (1478, 2)
The types of data in each column are: TEMP - KGR inflow     float64
FE - KGR inflow      float64
dtype: object
```

```
# Let's view the data to get an overall understanding
df
```

	TEMP - KGR inflow	FE - KGR inflow
0	19.6	1.60
1	19.2	0.80
2	19.7	0.20
3	18.0	4.00
4	19.0	9.00
...	...	...
1473	19.9	0.21
1474	21.3	0.19
1475	21.6	0.19
1476	21.5	0.19
1477	22.5	0.19

1478 rows × 2 columns

```
# Let's see the info for each attribute

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1478 entries, 0 to 1477
Data columns (total 2 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   TEMP - KGR inflow   1478 non-null   float64
 1   FE - KGR inflow     1478 non-null   float64
dtypes: float64(2)
memory usage: 23.2 KB
```

```
# Notice : No null values nor NaN (yay !)

#Let's see the descriptive statistics for each attribute

df.describe()
```

	TEMP - KGR inflow	FE - KGR inflow
count	1478.000000	1478.000000
mean	19.570981	0.342423
std	2.665918	1.054001
min	8.400000	0.000000
25%	17.900000	0.100000
50%	19.900000	0.150000
75%	21.500000	0.210750
max	26.500000	13.000000

```
#Let's see the correlation between numerical attributes
df.corr()
```

	TEMP - KGR inflow	FE - KGR inflow
TEMP - KGR inflow	1.000000	0.080258
FE - KGR inflow	0.080258	1.000000

```
# Let's view the distributions of the data as is, we will do
normalization afterwards to get rid of the differeing scales and then
re-visualize

# DATA VISUALIZATION

# Unimodel data Visualization - individual attributes

# histograms

df.hist(figsize = (10,7) , grid = False, edgecolor = "k")
plt.tight_layout()
plt.show()
# Now lets look at some density plots
```

```
df.plot(kind='density', figsize = (10,7), subplots=True, layout =(2,2),
sharex=False)
plt.tight_layout()
plt.show()
# Now let's see some Box and Whisker plots

df.plot(kind='box', figsize = (10,7), subplots=True , layout=(2,2) ,
sharex=False)
plt.tight_layout()
plt.show()
# Lots of outliers esp in FE, so lets remove them
# let's remove all rows that have at least one outlier in each column

# Define a function to detect and remove outliers using the IQR (Inter
Quartile Range) method
def drop_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - (1.5 * IQR)
    upper_bound = Q3 + (1.5 * IQR)
    return column[(column >= lower_bound) & (column <= upper_bound)]

# Loop through each column and drop the rows that contain the outliers
for col in df.columns:
    df = df.loc[drop_outliers(df[col]).index]

df
```

	TEMP - KGR inflow	FE - KGR inflow
2	19.7	0.20
36	23.7	0.01
37	23.4	0.10
38	23.1	0.20
48	20.6	0.01
...	...	...
1473	19.9	0.21
1474	21.3	0.19
1475	21.6	0.19
1476	21.5	0.19
1477	22.5	0.19

1320 rows × 2 columns

```
#Notice the shape changed to 1320 rows

# Now let's see some Box and Whisker plots
```

```
df.plot(kind='box', figsize = (10,7), subplots=True , layout=(2,2) ,
sharex=False)
plt.tight_layout()
plt.show()
# Much better now, less outliers

# RE-Visualize the data without any Normalizations 1st

# DATA VISUALIZATION

# Unimodel data Visualization - individual attributes

# histograms

df.hist(figsize = (10,7) , grid = False, edgecolor = "k")
plt.tight_layout()
plt.show()
# Now lets look at some density plots

df.plot(kind='density', figsize = (10,7), subplots=True, layout =(2,2),
sharex=False)
plt.tight_layout()
plt.show()

# Now let's see some Box and Whisker plots

df.plot(kind='box', figsize = (10,7), subplots=True , layout=(2,2) ,
sharex=False)
plt.tight_layout()
plt.show()

# MULTIMODAL DATA VISUALIZATIONS

#scatter plot matrix
from pandas.plotting import scatter_matrix

scatter_matrix(df,figsize = (10,10),edgecolor = "k")
plt.show()

# Now lets see the correlation matrix
# Using seaborn

import seaborn as sns

plt.figure(figsize=(10,10))
sns.heatmap(df.corr(), annot=True)

# NORMALIZATION
```

```
# Now let's scale the data since they each have different units. So for
regression we need the same scale.
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

# transform data
scaled = scaler.fit_transform(df)

# Replace the existing dataframe with the new one
df = pd.DataFrame(scaled)

# Ensure the old headings remain
df.set_axis(['TEMP - KGR inflow', 'FE - KGR inflow'], axis='columns',
inplace=True),

# REVISUALIZE NORMALIZED DATA
# DATA VISUALIZATION
# Unimodel data Visualization - individual attributes
# histograms

df.hist(figsize = (10,7) , grid = False, edgecolor = "k")
plt.tight_layout()
plt.show()

df.plot(kind='density', figsize = (10,7), subplots=True, layout =(2,2),
sharex=False)
plt.tight_layout()
plt.show()

df.plot(kind='box', figsize = (10,7), subplots=True , layout=(2,2) ,
sharex=False)
plt.tight_layout()
plt.show()

#scatter plot matrix
from pandas.plotting import scatter_matrix

scatter_matrix(df,figsize = (10,10),edgecolor = "k")
plt.show()

plt.figure(figsize=(10,10))
sns.heatmap(df.corr(), annot=True)
# Now save this to a new Excel file

df.to_excel ('West Rand - KGR inflow (no outliers and normalized) for
regression.xlsx')
```

APPENDIX H : TRANSFER LEARNING AND EDA PYTHON CODE FOR KGR INFLOW (WEST RAND)

```
# Start by training Central Rand Pump B

# Load the libraries and packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk

# Data for Pump B for Central Rand was pre-cleaned and modelled using
PHREEQC
# Values to be used are Temperature (TEMP), Electrical
Conductivity(EC), pH, Total Dissolved Solids (TDS), Sulphate (SUL),
Iron (FE) and Manganese (MN)

#Load dataset from excel. Note the read_excel() function

df = pd.read_excel('/content/Pump B - Central Rand (Cleaned and
modelled).xlsx')

# Now remove these outliers

df.drop([387], axis = 'index' , inplace = True)
df.drop([1116], axis = 'index' , inplace = True)

# Define a function to detect and remove outliers using the IQR (Inter
Quartile Range) method
def drop_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - (1.5 * IQR)
    upper_bound = Q3 + (1.5 * IQR)
    return column[(column >= lower_bound) & (column <= upper_bound)]

# Loop through each column and drop the rows that contain the outliers
for col in df.columns:
    df = df.loc[drop_outliers(df[col]).index]

# NORMALIZATION

# Now let's scale the data since they each have different units. So for
regression we need the same scale.
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
# transform data
scaled = scaler.fit_transform(df)

# Replace the existing dataframe with the new one
df = pd.DataFrame(scaled)

# Ensure the old headings remain
df.set_axis(['B-TEMP','B-EC','B-TDS','B-pH','B-SUL','B-FE','B-MN'],
axis='columns', inplace=True),

# Replace the existing dataframe with the new one
df = df.loc[:, ['B-TEMP','B-FE','B-SUL']]

# Ensure the old headings remain
df.set_axis(['B-TEMP','B-FE','B-SUL'], axis='columns', inplace=True)
# TRAIN GERM PUMP B'S BEST MODEL - STACKED USING ONLY GOOD MODELS

from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor

# a) Split-out test dataset

array = df.values
X = array[:,0:2] #remember, the last column is 2 not 3 cause of how
Python counts
y = array[:,2]

X_train, X_test, y_train, y_test = train_test_split (X , y , test_size
= 0.2 , random_state = 7)

# Test options
num_folds = 10 # 10 fold cross validation
seed = 7
scoring = 'neg_mean_squared_error'

# Stacked ensemble

from sklearn.ensemble import StackingRegressor
```

```
# get a stacking ensemble of models

def get_stacking():

 # define the base models
 # Let's try ONLY THE BEST MODELS NOW, remove LASSO,EN and SVR
 level0 = list()
 level0.append(('LR', LinearRegression()))
 level0.append(('RD', Ridge()))
 level0.append(('KNNR', KNeighborsRegressor()))
 level0.append(('DT', DecisionTreeRegressor()))
 level0.append(('XG', XGBRegressor()))
 level0.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
 level0.append(('MLP', MLPRegressor()))

 # define meta learner model
 level1 = LinearRegression()

 # define the stacking ensemble
 model = StackingRegressor(estimators=level0, final_estimator=level1,
cv=10)

 return model


# FINALIZE THE MODEL

#First train the selected model
model = get_stacking()

model.fit(X_train,y_train)

# WEST RAND - KGR INFLOW

#Load dataset from excel.
df_kgr = pd.read_excel('West Rand - KGR inflow (no outliers and
normalized) for regression.xlsx')

# TRANSFER LEARNING
X_test_new = df_kgr

#Now test the model on the test dataset
predictions = model.predict(X_test_new)
#Convert predictions array to dataframe
df_sul = pd.DataFrame(predictions)

df_sul
```

	0
0	0.533243
1	0.025140
2	0.259953
3	0.521234
4	0.037611
...	...
1315	0.557574
1316	0.499661
1317	0.499115
1318	0.499382
1319	0.496543

1320 rows × 1 columns

```
# Now unnormalize by applying x = y(max-min) + min

# From Central Rand pump B :

sul_max = 0.016430
sul_min = 0.000001

# create an unnormalization function in Python
def unnormalize(df):
  df['unnormalized'] = df.apply(lambda row: row[0]*(sul_max - sul_min)
+ sul_min, axis=1)
  return df

df_sul = unnormalize(df_sul)

df_sul.set_axis(['SUL predicted (normalized) - KGR inflow','SUL
predicted (un-normalized) - KGR inflow'],axis='columns', inplace=True)

# understand predicted values via stats and visualization

df_sul.describe()
```

	SUL predicted (normalized) - KGR inflow	SUL predicted (un-normalized) - KGR inflow
count	1320.000000	1320.000000
mean	0.393230	0.006461
std	0.196380	0.003226
min	0.022638	0.000373
25%	0.267927	0.004403
50%	0.358958	0.005898
75%	0.526267	0.008647
max	0.981780	0.016131

```
df_sul.hist(figsize = (10,7) , grid = False, edgecolor = "k")
plt.tight_layout()
plt.show()
# Now lets look at some density plots

df_sul.plot(kind='density', figsize = (14,14), subplots=True, layout
=(2,2), sharex=False)
plt.tight_layout()
plt.show()
# Now let's see some Box and Whisker plots

df_sul.plot(kind='box', figsize = (10,7), subplots=True , layout=(2,2)
, sharex=False)
plt.tight_layout()
plt.show()

# Save to excel
df_sul.to_excel ('West Rand - KGR inflow Sulphate predicted from
TL.xlsx')

df_sul
```

APPENDIX I : CENTRAL RAND DATASET STATISTICAL VALUES

AI 1 – Overall

Table A1 : Mean values of water quality parameters (normalized and unnormalized) for the Central Rand dataset.

		Mean Value						
		TEMP (°C)	**EC (mS/m)**	**TDS (mg/L)**	**pH**	**SUL (mg/L)**	**FE (mg/L)**	**MN (mg/L)**
Prior to normalization	**Pump A**	26.03	508.64	5951.39	5.78	0.01	657.09	27.32
	Pump B	26.16	507.48	5932.77	5.79	0.01	652.05	27.35
	Treated Water	25.24	401.01	3994.29	8.71	0.03	0.02	1.22
After normalization	**Pump A**	0.5600	0.4864	0.5262	0.5130	0.3937	0.4127	0.5118
	Pump B	0.5222	0.4733	0.4504	0.5065	0.4268	0.4243	0.4908
	Treated Water	0.5041	0.4866	0.4942	0.5497	0.4977	0.2586	0.3780

AI 2 – Pump A

Table A2 : Statistics used to plot histograms, density plots and box and whisker diagrams after outlier removal for Pump A.

	A - TEMP	A-EC	A-TDS	A-pH	A-SUL	A-FE	A-MN
count	1107.00	1107.00	1107.00	1107.00	1107.00	1107.00	1107.00
mean	26.03	508.64	5951.39	5.78	0.01	657.09	27.32
std	1.26	23.29	365.57	0.10	0.00	108.64	2.36
min	22.50	442.00	4730.00	5.53	0.01	421.02	20.29
25%	25.20	491.00	5620.00	5.73	0.01	567.35	25.59
50%	26.20	505.00	6002.00	5.78	0.01	631.52	27.44
75%	27.00	523.00	6192.00	5.83	0.01	740.93	29.19
max	28.80	579.00	7051.00	6.01	0.01	993.05	34.03

Table A3 : Model training and testing statistics obtained for Central Rand Pump A (shaded models are the best performing models).

Model	Training		Testing		
	NMSE	Standard deviation	MSE	MAE	R^2
LR	-0.00605	-0.001	0.006296	0.060355	0.810643
RD	-0.00606	-0.001062	0.006289	0.060255	0.810865
LASSO	-0.03217	-0.005089	0.033252	0.153985	-6.9×10^{-5}
EN	-0.03217	-0.005089	0.033252	0.153985	-6.9×10^{-5}
KNNR	-0.00615	-0.001026	0.006372	0.061412	0.808349
DT	-0.01072	-0.001448	0.012552	0.089095	0.622486
SVR	-0.00532	-0.000763	0.005275	0.055935	0.841351
XG	-0.00731	-0.00123	0.007042	0.065316	0.788193
RF	-0.00648	-0.000935	0.006457	0.061755	0.805815
MLP	-0.00597	-0.001033	0.0067	0.061866	0.798483
Stacked (All models)	-0.00544	-0.000814	0.005252	0.055598	0.842041
Stacked (only best models)	-0.00546	-0.000858	0.005335	0.055679	0.839538

AI 3 – Pump B

Table A4 : Statistics used to plot histograms, density plots and box and whisker diagrams after outlier removal for Pump B.

	B-TEMP	B-EC	B-TDS	B-pH	B-SUL	B-FE	B-MN
count	1189.00	1189.00	1189.00	1189.00	1189.00	1189.00	1189.00
mean	26.16	507.48	5932.77	5.79	0.01	652.05	27.35
std	1.29	20.83	349.85	0.09	0.00	103.69	2.23
min	22.40	445.00	5028.00	5.56	0.01	436.91	21.16
25%	25.40	491.00	5618.00	5.75	0.01	572.65	25.84
50%	26.40	506.00	5965.00	5.80	0.01	630.11	27.38
75%	27.20	519.00	6148.00	5.85	0.01	719.43	29.01
max	29.60	577.00	7037.00	6.02	0.01	944.00	33.77

Table A5 : Model training and testing statistics obtained for Central Rand Pump B (shaded model is the best performing model).

Model	Training		Testing		
	NMSE	Standard deviation	MSE	MAE	R^2
LR	-1.8×10^{-5}	-0.000005	0.000013	0.002706	0.99971
RD	-5.7×10^{-5}	-0.000008	0.000047	0.005536	0.998901
LASSO	-0.04149	-0.005781	0.043197	0.169071	-0.0001
EN	-0.04149	-0.005781	0.043197	0.169071	-0.0001
KNNR	-0.00013	-0.000065	0.000086	0.006893	0.99801
DT	-5.1×10^{-5}	-0.000012	0.000048	0.005373	0.998883
SVR	-0.00242	-0.000258	0.002405	0.044632	0.944313
XG	-3.2×10^{-5}	-0.000009	0.000028	0.004133	0.999354
RF	-3.5×10^{-5}	-0.000011	0.000029	0.004299	0.999324
MLP	-0.00045	-0.000232	0.000487	0.016707	0.988725
Stacked (All models)	-1.6×10^{-5}	-0.000004	0.000011	0.002656	0.999741
Stacked (only best models)	-1.6×10^{-5}	-0.000005	0.000011	0.002617	0.999737

AI 4 – Treated Water

Table A6 : Statistics used to plot histograms, density plots and box and whisker diagrams after outlier removal for Treated Water.

	T-TEMP	T-EC	T-pH	T-TDS	T-SUL	T-FE	T-MN
count	1197	1197	1197	1197	1197	1197	1197
mean	25.24	401.01	8.71	3994.29	0.03	0.02	1.22
std	1.550	12.476	0.179	148.330	0.001	0.018	0.681
min	20.8	365	8.18	3594	0.0236	0	0.05
25%	24.2	393	8.59	3902	0.02689	0	0.69
50%	25.4	400	8.73	3998	0.02792	0.01	1.017
75%	26.4	408	8.85	4079	0.0289	0.03	1.616
max	29.6	439	9.15	4404	0.03225	0.07	3.146

Table A7 : Model training and testing statistics obtained for Central Rand Treated Water (shaded model is the best performing model).

Model	Training		Testing		
	NMSE	Standard deviation	MSE	MAE	R^2
LR	-0.02526	-0.003536	0.024533	0.122144	0.137285
RD	-0.02526	-0.003526	0.024502	0.12226	0.138378
LASSO	-0.02947	-0.003407	0.028459	0.134916	-0.00077
EN	-0.02947	-0.003407	0.028459	0.134916	-0.00077
KNNR	-0.02944	-0.004281	0.029781	0.133188	-0.04724
DT	-0.04845	-0.00349	0.055957	0.191006	-0.96773
SVR	-0.02518	-0.004379	0.024902	0.123406	0.124322
XG	-0.03296	-0.005712	0.033683	0.14398	-0.18448
RF	-0.02794	-0.004542	0.029278	0.133407	-0.02958
MLP	-0.02576	-0.003418	0.025086	0.123928	0.117861
Stacked (All models)	-0.02495	-0.003778	0.024362	0.121571	0.143315
Stacked (only best models)	-0.02501	-0.00388	0.024403	0.121403	0.141863

APPENDIX J : DESCRIPTIVE STATISTICS PERTAINING TO THE EAST RAND DATASET STATISTICS

Table A8 : Descriptive statistics for the East Rand AMD feed.

ER AMD-	TEMP	EC	TDS	pH	SUL	FE	MN
count	1492	1492	1492	1492	1492	1492	1492
mean	23.12	294.29	2631.13	6.45	0.001592	95.34	5.99
std	1.95	14.02	195.42	0.19	0.00018	11.04	2.51
min	18.00	253.00	1953.00	6.00	0.001133	66.56	2.01
25%	22	285	2452	6.31	0.001458	87.34	4.107
50%	23.4	296	2598.5	6.40	0.00158	95.69	4.838
75%	24.5	306	2830	6.58	0.001721	103	7.70
max	28.4	321	3375	6.98	0.002107	127	13.8

Table A9 : Descriptive statistic comparison between true sulphate values and predicted sulphate values using TL.

	SUL True	SUL Predicted
count	1492	1492
mean	0.001592	0.001599
std	0.00018	0.000176
min	0.001133	0.001149
25%	0.001458	0.00147
50%	0.00158	0.001601
75%	0.00172	0.00172
max	0.00211	0.00209

APPENDIX K : DESCRIPTIVE STATISTICS PERTAINING TO THE WEST RAND DATASET

Table A10 : Statistics pertaining to the untreated AMD ("KGR inflow") of the West Rand dataset.

	TEMP - KGR inflow	FE - KGR inflow	SUL Predicted - KGR inflow
count	1320	1320	1320
mean	19.618	0.146	0.393
std	2.532	0.076	0.196
min	12.500	0.000	0.023
25%	17.900	0.100	0.268
50%	19.900	0.130	0.359
75%	21.500	0.200	0.526
max	26.500	0.376	0.982

Table A11 : Statistics pertaining to the Treated AMD ("Treated flow into KGR") of the West Rand dataset.

Treated flow into KGR	TEMP	FE	SUL Predicted
count	1407	1407	1407
mean	19.6	0.130	0.00611
std	2.52	0.075	0.00338
min	12.6	0.000	0.000343
25%	17.9	0.0900	0.00436
50%	19.9	0.110	0.00507
75%	21.5	0.185	0.00856
max	26.2	0.355	0.0162

APPENDIX L : SULFIND GUI CODE (Tkinter)

```
# Install the necessary packages
!pip install xgboost
!pip install pandas
!pip install matplotlib
!pip install scikit-learn
!pip install openpyxl

# Start by training Central Rand Pump B

# Load the libraries and packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn as sk

# Data for Pump B for Central Rand was pre-cleaned and modelled using
PHREEQC
# Values to be used are Temperature (TEMP), Electrical
Conductivity(EC), pH, Total Dissolved Solids (TDS), Sulphate (SUL),
Iron (FE) and Manganese (MN)

#Load dataset from excel. Note the read_excel() function

df = pd.read_excel(r'C:\Users\taske\Desktop\Masters - ML and AMD
(sulphate)\Data\Split dataframes\Pump B\Pump B - Central Rand (Cleaned
and modelled).xlsx')

# Now remove these outliers

df.drop([387], axis = 'index' , inplace = True)
df.drop([1116], axis = 'index' , inplace = True)

# Define a function to detect and remove outliers using the IQR (Inter
Quartile Range) method
def drop_outliers(column):
    Q1 = column.quantile(0.25)
    Q3 = column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - (1.5 * IQR)
    upper_bound = Q3 + (1.5 * IQR)
    return column[(column >= lower_bound) & (column <= upper_bound)]

# Loop through each column and drop the rows that contain the outliers
for col in df.columns:
    df = df.loc[drop_outliers(df[col]).index]
```

```
# NORMALIZATION

# Now let's scale the data since they each have different units. So for
regression we need the same scale.
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()

# transform data
scaled = scaler.fit_transform(df)

# Replace the existing dataframe with the new one
df = pd.DataFrame(scaled)

# Ensure the old headings remain
df = pd.DataFrame(scaled, columns=['B-TEMP', 'B-EC', 'B-TDS', 'B-pH',
'B-SUL', 'B-FE', 'B-MN'])

# Replace the existing dataframe with the new one
df = df.loc[:, ['B-TEMP','B-FE','B-SUL']]

# Ensure the old headings remain
df = df[['B-TEMP', 'B-FE', 'B-SUL']]

# TRAIN CR PUMP B'S BEST MODEL - STACKED USING ONLY GOOD MODELS

from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor

# a) Split-out test dataset

array = df.values
X = array[:,0:2] #remember, the last column is 2 not 3 cause of how
Python counts
y = array[:,2]

X_train, X_test, y_train, y_test = train_test_split (X , y , test_size
= 0.2 , random_state = 7)

# Test options
```

```
num_folds = 10 # 10 fold cross validation
seed = 7
scoring = 'neg_mean_squared_error'

# Stacked ensemble

from sklearn.ensemble import StackingRegressor

# get a stacking ensemble of models

def get_stacking():

 # define the base models
 # Let's try ONLY THE BEST MODELS NOW, remove LASSO,EN and SVR
 level0 = list()
 level0.append(('LR', LinearRegression()))
 level0.append(('RD', Ridge()))
 level0.append(('KNNR', KNeighborsRegressor()))
 level0.append(('DT', DecisionTreeRegressor()))
 level0.append(('XG', XGBRegressor()))
 level0.append(('RF', RandomForestRegressor(n_estimators = 100,
random_state = 0)))
 level0.append(('MLP', MLPRegressor()))

 # define meta learner model
 level1 = LinearRegression()

 # define the stacking ensemble
 model = StackingRegressor(estimators=level0, final_estimator=level1,
cv=10)

 return model


# FINALIZE THE MODEL

#First train the selected model
model = get_stacking()

model.fit(X_train,y_train)

# Now export the model
import joblib

# Save the model using joblib.dump
model_path = r"\Users\taske\Desktop\Masters - ML and AMD (sulphate)\SE
Model.pkl"
joblib.dump(model, model_path)
```

```
# Now create the GUI

import tkinter as tk
from tkinter import ttk
import joblib
import numpy as np

class RegressionModelGUI:
    def __init__(self, model_path):
        self.root = tk.Tk()
        self.root.title("Sulfind GUI")

        # Set the background color to yellow
        self.root.configure(bg="yellow")

        self.model = joblib.load(model_path)

        # Min and Max values for normalization (Temperature and Iron)
        self.min_temperature = 0.0
        self.max_temperature = 100.0
        self.min_iron = 0.0
        self.max_iron = 10.0

        # Min and Max values for unnormalization (Sulphate)
        self.min_sulfate = 0.000001
        self.max_sulfate = 0.002107

        # Load the logo image
        self.logo_image =
tk.PhotoImage(file=r"C:\Users\taske\Desktop\Masters - ML and AMD
(sulphate)\Sulfind\Sulfind logo.png").subsample(5, 5)
        self.root.update_idletasks()

        self.create_widgets()

    def create_widgets(self):

        # Create a label to display the logo
        logo_label = tk.Label(self.root, image=self.logo_image,
bg="yellow")
        logo_label.grid(row=0, column=4, rowspan=7)

        ttk.Label(self.root, text="Temperature (\u00B0C)",
background="yellow").grid(row=0, column=0)
        ttk.Label(self.root, text="Min",
background="yellow").grid(row=0, column=1)
        ttk.Label(self.root, text="Max",
background="yellow").grid(row=0, column=2)
```

```
        ttk.Label(self.root, text="Value",
background="yellow").grid(row=0, column=3)

        ttk.Label(self.root, text="Iron (mg/L)",
background="yellow").grid(row=3, column=0)
        ttk.Label(self.root, text="Min",
background="yellow").grid(row=3, column=1)
        ttk.Label(self.root, text="Max",
background="yellow").grid(row=3, column=2)
        ttk.Label(self.root, text="Value",
background="yellow").grid(row=3, column=3)

        self.entry_temp_min = ttk.Entry(self.root, width=10)
        self.entry_temp_min.grid(row=1, column=1)
        self.entry_temp_max = ttk.Entry(self.root, width=10)
        self.entry_temp_max.grid(row=1, column=2)
        self.entry_temp_input = ttk.Entry(self.root, width=10)
        self.entry_temp_input.grid(row=1, column=3)

        self.entry_iron_min = ttk.Entry(self.root, width=10)
        self.entry_iron_min.grid(row=4, column=1)
        self.entry_iron_max = ttk.Entry(self.root, width=10)
        self.entry_iron_max.grid(row=4, column=2)
        self.entry_iron_input = ttk.Entry(self.root, width=10)
        self.entry_iron_input.grid(row=4, column=3)

        style = ttk.Style()
        style.configure("Black.TButton", foreground="black",
background="black")

        self.btn_predict = ttk.Button(self.root, text="Predict Sulphate
(mg/L)", command=self.predict, style="Black.TButton")
        self.btn_predict.grid(row=5, column=0, columnspan=4)

        self.label_result = ttk.Label(self.root, text="",
foreground="black", background="yellow")
        self.label_result.grid(row=6, column=0, columnspan=4)

    def normalize_input(self, temp_min, temp_max, temp_input, iron_min,
iron_max, iron_input):
        # Normalize the input using MinMaxScaler
        temp_input_normalized = (temp_input - temp_min) / (temp_max -
temp_min)
        iron_input_normalized = (iron_input - iron_min) / (iron_max -
iron_min)
        return temp_input_normalized, iron_input_normalized

    def unnormalize_output(self, prediction):
```

```
        # Unnormalize the sulfate prediction
        return prediction * (self.max_sulfate - self.min_sulfate) +
self.min_sulfate

    def predict(self):
        try:
            # Extract user input values
            temp_min = float(self.entry_temp_min.get())
            temp_max = float(self.entry_temp_max.get())
            temp_input = float(self.entry_temp_input.get())

            iron_min = float(self.entry_iron_min.get())
            iron_max = float(self.entry_iron_max.get())
            iron_input = float(self.entry_iron_input.get())

            # Normalize the input features
            temp_input_normalized, iron_input_normalized =
self.normalize_input(temp_min, temp_max, temp_input,

        iron_min, iron_max, iron_input)

            # Create input feature array
            input_features = np.array([[temp_input_normalized,
iron_input_normalized]])

            # Make prediction using the model
            prediction_normalized =
self.model.predict(input_features)[0]

            # Unnormalize the sulfate prediction
            prediction_unnormalized =
self.unnormalize_output(prediction_normalized)

            self.label_result.config(text=f"Predicted Sulphate Value:
{prediction_unnormalized:.6f}")
        except Exception as e:
            self.label_result.config(text="Error: Invalid input")

if __name__ == "__main__":
    model_path = r"C:\Users\taske\Desktop\Masters - ML and AMD
(sulphate)\SE Model.pkl"
    app = RegressionModelGUI(model_path)
    app.root.mainloop()
```

APPENDIX M: SULFIND GUI CODE (Flask)

```
from flask import Flask, render_template, request

import joblib
import numpy as np

app = Flask(__name__)

# Define the file path for the model
model_path = "SE_Model.pkl"

# Load your machine learning model
model = joblib.load(model_path)

@app.route('/')
def index():
    return """
        <html>
            <head>
                <title>Sulfind GUI</title>
            </head>
            <body>
                <h1>Sulfind GUI</h1>
                <form method="POST" action="/predict">
                    <label for="temp-min">Min Temperature (°C)</label>
                       <input type="text" id="temp-min" name="temp-min"
placeholder="Min Temperature (°C)"><br>
                    <label for="temp-max">Max Temperature (°C)</label>
                       <input type="text" id="temp-max" name="temp-max"
placeholder="Max Temperature (°C)"><br>
                  <label for="temp-input">Temperature Input (°C)</label>
                    <input type="text" id="temp-input" name="temp-input"
placeholder="Temperature Input (°C)"><br>
                    <label for="iron-min">Min Iron (mg/L)</label>
                       <input type="text" id="iron-min" name="iron-min"
placeholder="Min Iron (mg/L)"><br>
                    <label for="iron-max">Max Iron (mg/L)</label>
                       <input type="text" id="iron-max" name="iron-max"
placeholder="Max Iron (mg/L)"><br>
                    <label for="iron-input">Iron Input (mg/L)</label>
                    <input type="text" id="iron-input" name="iron-input"
placeholder="Iron Input (mg/L)"><br>
                   <button type="submit">Predict Sulphate (mg/L)</button>
                </form>
                <p id="prediction-output"></p>
            </body>
        </html>
    """
```

```
@app.route('/predict', methods=['POST'])
def predict():
    try:
        temp_min = float(request.form['temp-min'])
        temp_max = float(request.form['temp-max'])
        temp_input = float(request.form['temp-input'])
        iron_min = float(request.form['iron-min'])
        iron_max = float(request.form['iron-max'])
        iron_input = float(request.form['iron-input'])

        # Normalize the input features
         temp_input_normalized = (temp_input - temp_min) / (temp_max -
temp_min)
         iron_input_normalized = (iron_input - iron_min) / (iron_max -
iron_min)

        # Create input feature array
                 input_features    =    np.array([[temp_input_normalized,
iron_input_normalized]])

        # Make prediction using the model
        prediction_normalized = model.predict(input_features)[0]

        # Unnormalize the sulfate prediction
        min_sulfate = 0.000001
        max_sulfate = 0.002107
        prediction_unnormalized = prediction_normalized * (max_sulfate -
min_sulfate) + min_sulfate

                    return    f"Predicted    Sulphate    Value    (mg/L):
{prediction_unnormalized:.6f}"
    except Exception as e:
        return "Error: Invalid input"

if __name__ == '__main__':
    app.run(debug=True)
```

APPENDIX N: SELECTED ChatGPT CONVERSTAION SCREENSHOTS

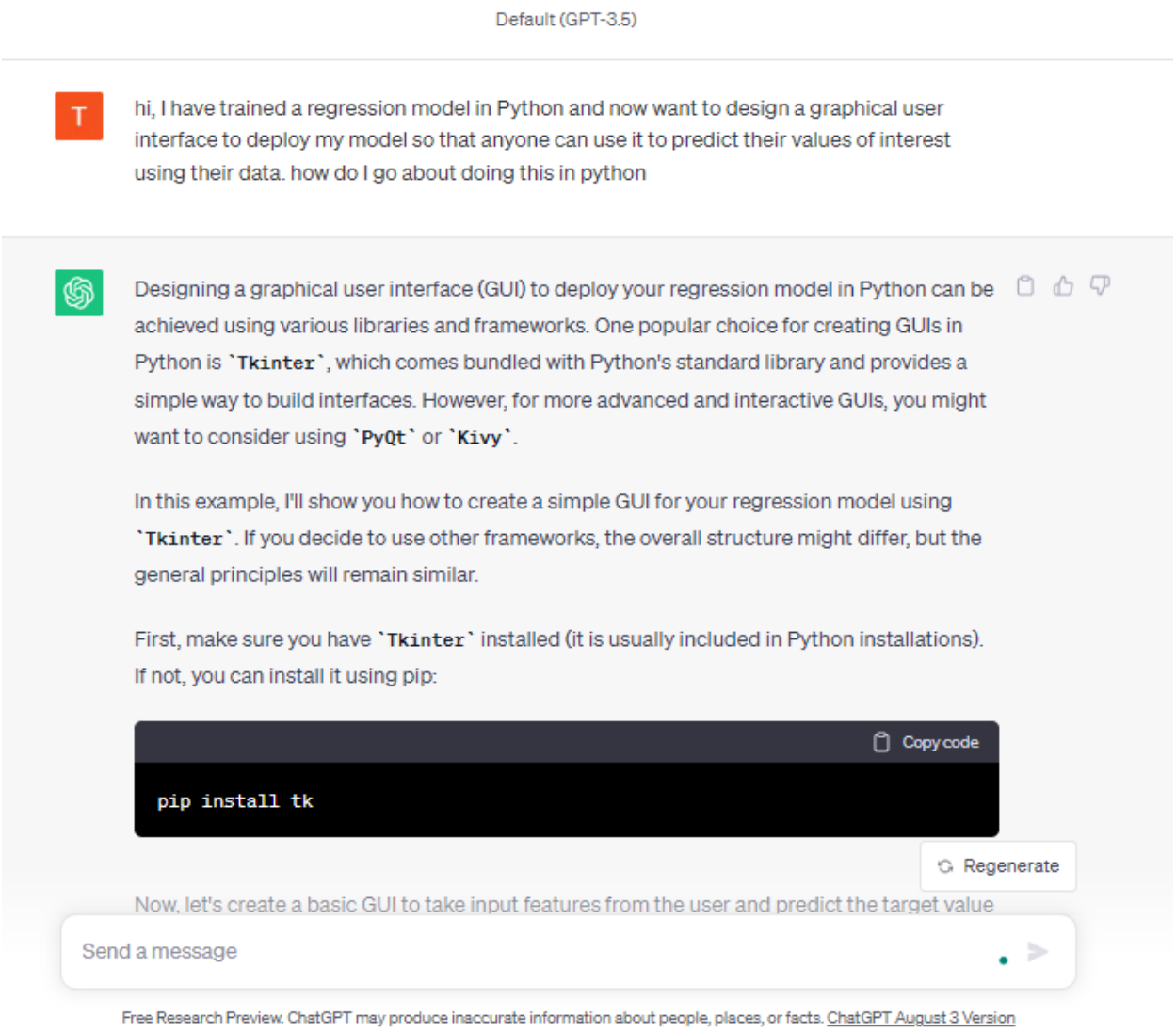


Figure A1: First prompt given to ChatGPT to create a GUI for the trained regression model (Central Rand Pump B SE-ML).

Now, let's create a basic GUI to take input features from the user and predict the target value using your trained regression model:

python Copy code

```
import tkinter as tk
from tkinter import ttk
import joblib
import numpy as np

class RegressionModelGUI:
    def __init__(self, model_path):
        self.root = tk.Tk()
        self.root.title("Regression Model GUI")

        self.model = joblib.load(model_path)

        self.create_widgets()

    def create_widgets(self):
        self.label_features = ttk.Label(self.root, text="Enter features (com
        self.label_features.pack()

        self.entry_features = ttk.Entry(self.root)
        self.entry_features.pack()

        self.btn_predict = ttk.Button(self.root, text="Predict", command=sel
```

Regene

Send a message

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT August 3 Version

Figure A2: Screenshot of initial ChatGPT generated Python code to create the Sulfind GUI.

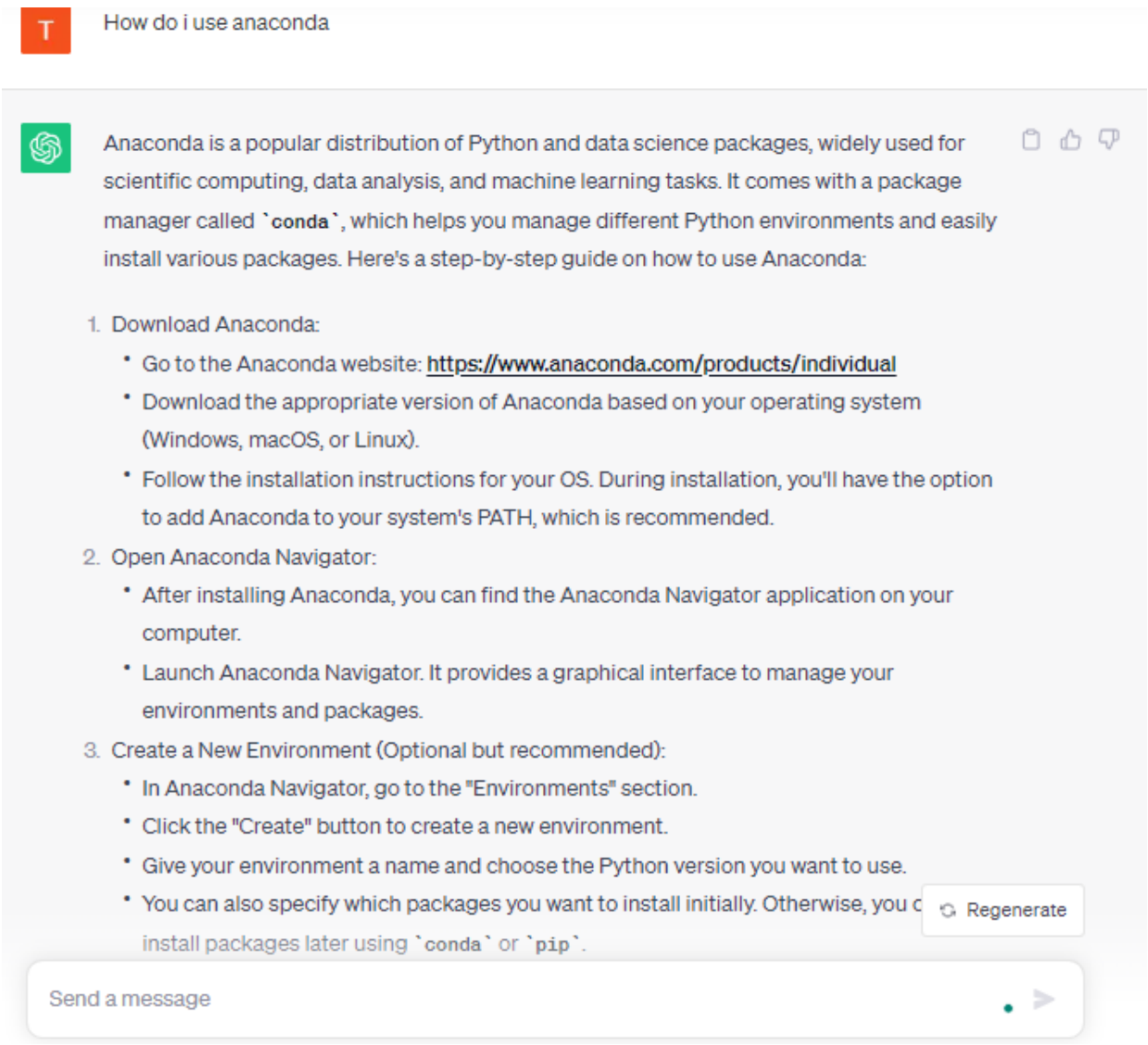
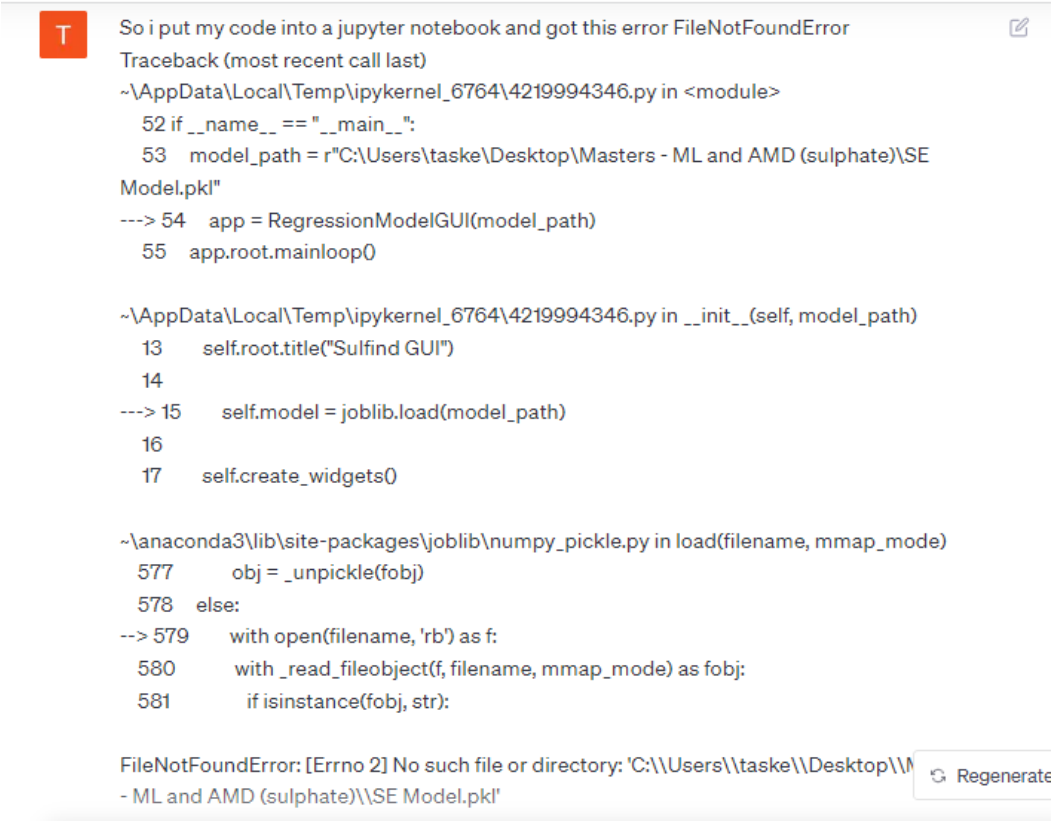


Figure A3: clear and concise instructions given by ChatGPT on how to install and use Anaconda Navigator.



So i put my code into a jupyter notebook and got this error FileNotFoundError
Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_6764\4219994346.py in <module>
52 if __name__ == "__main__":
53 model_path = r"C:\Users\taske\Desktop\Masters - ML and AMD (sulphate)\SE Model.pkl"
---> 54 app = RegressionModelGUI(model_path)
55 app.root.mainloop()

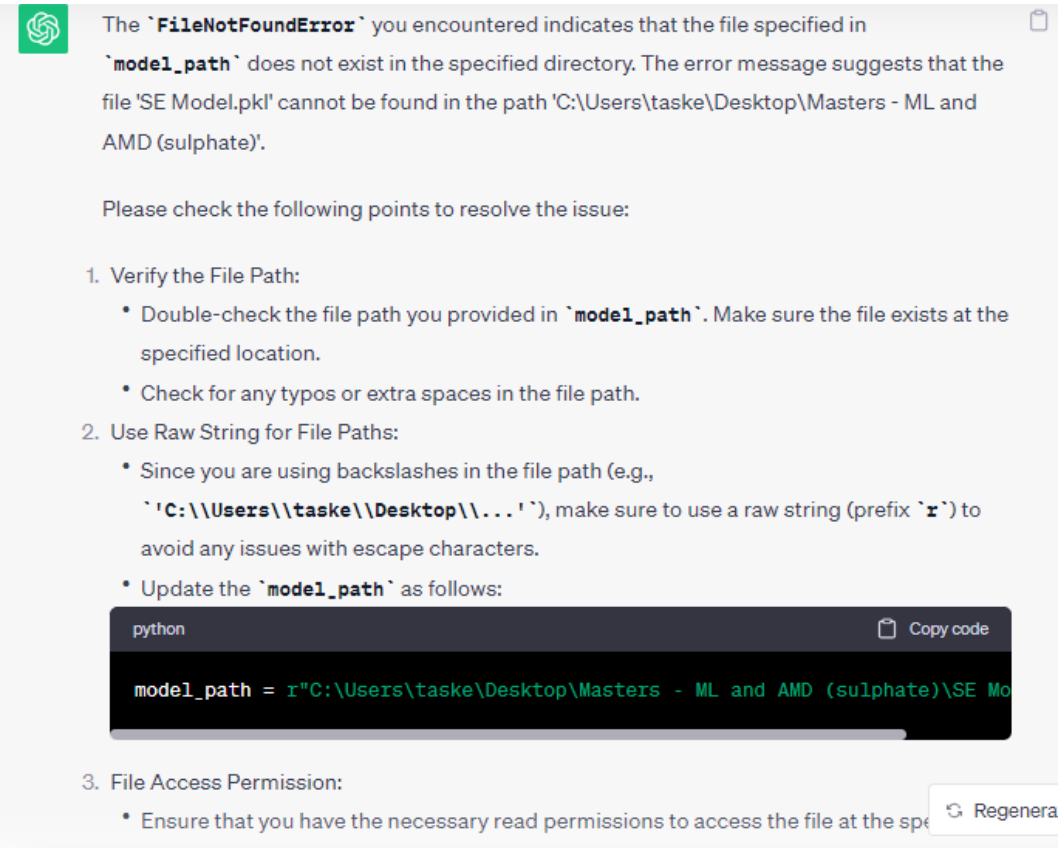
~\AppData\Local\Temp\ipykernel_6764\4219994346.py in __init__(self, model_path)
13 self.root.title("Sulfind GUI")
14
---> 15 self.model = joblib.load(model_path)
16
17 self.create_widgets()

~\anaconda3\lib\site-packages\joblib\numpy_pickle.py in load(filename, mmap_mode)
577 obj = _unpickle(fobj)
578 else:
--> 579 with open(filename, 'rb') as f:
580 with _read_fileobject(f, filename, mmap_mode) as fobj:
581 if isinstance(fobj, str):

FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\taske\\Desktop\\M - ML and AMD (sulphate)\\SE Model.pkl'

Regenerate

Figure A4: Error message fed into ChatGPT in order to debug the code (human negative error message feedback loop creation).



The `FileNotFoundError` you encountered indicates that the file specified in `model_path` does not exist in the specified directory. The error message suggests that the file 'SE Model.pkl' cannot be found in the path 'C:\Users\taske\Desktop\Masters - ML and AMD (sulphate)'.

Please check the following points to resolve the issue:

1. Verify the File Path:
   - Double-check the file path you provided in `model_path`. Make sure the file exists at the specified location.
   - Check for any typos or extra spaces in the file path.
2. Use Raw String for File Paths:
   - Since you are using backslashes in the file path (e.g., `'C:\\Users\\taske\\Desktop\\...'`), make sure to use a raw string (prefix `r`) to avoid any issues with escape characters.
   - Update the `model_path` as follows:

python Copy code

```
model_path = r"C:\Users\taske\Desktop\Masters - ML and AMD (sulphate)\SE Mo
```

3. File Access Permission:
   - Ensure that you have the necessary read permissions to access the file at the spe

Regenera

Figure A5: ChatGPT debugging the error message fed to it along with an explanation.

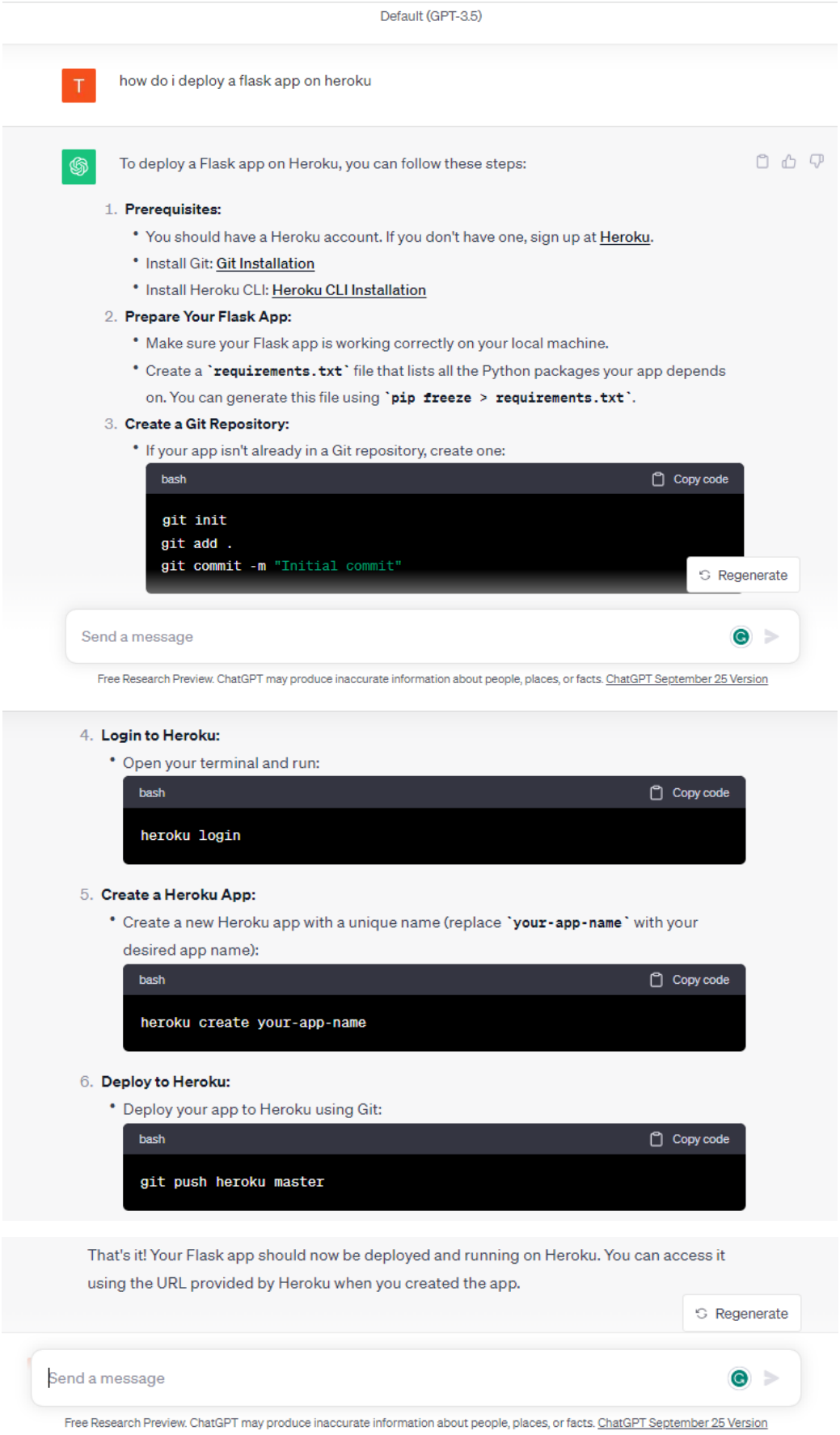


Figure A6: Portion of ChatGPT conversation on how to deploy a Flask app to the web using Heroku.