DEGREE PROJECT IN NUCLEAR ENERGY ENGINEERING,
SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2020

A Machine Learning Approach to the Nuclear Fuel Fabrication Process

ELINA CHARATSIDOU

KTH Royal Institute of Technology
School of Engineering Sciences
Department of Physics
MSc in Nuclear Energy Engineering

Elina Charatsidou, MSc
elinach@kth.se

Elina Charatsidou, MSc
elinach@kth.se

A Machine Learning Approach to the Nuclear Fuel Fabrication Process

Elina Charatsidou
elinach@kth.se

Master Thesis Project
Westinghouse Electric Sweden AB
Västerås, Sweden
Supervisor: Denise Adorno Lopes
KTH Supervisor: Pär Olsson
June 2020

Elina Charatsidou, MSc
elinach@kth.se

Elina Charatsidou, MSc
elinach@kth.se

Abstract

The nuclear fuel fabrication is a complex process which requires precise control of many process parameters to obtain a final product conforming to stringent specifications. Hence, the yield improvement is one of the most important topics in nuclear fuel fabrication. The yield is driven down by demands on the final quality of the pellet: density, impurities, surface defects, etc., forcing to recycle some of the pellets, with the associated cost.

Over time, the process was able to reach a significant yield thanks to the use of traditional statistical methods on the various sub-processes. However, traditional approaches have limits in extracting the full benefits of the data, since trends can be hard to find. Therefore, the manufacturing data is currently poorly explored even in the most sophisticated process.

In this work, data from the nuclear fuel factory of Westinghouse Electric AB in Västerås, Sweden, were manually collected, organized, and structured in a way useful for data analysis and machine learning implementations. Afterwards, machine learning algorithms, namely neural network and gradient boosting, were applied to build models, feature weights of the parameter process, and understand correlations: the trained models were later used to predict output label values based on new datasets, and an evaluation of these predictions was performed, alongside with the comparison of the performance of both neural networks and gradient boosting for such problems. Through this method, it will be possible to model the fabrication process and use this tool to pursue improvements based on data.

Hence, the aim of this work is to facilitate the preprocessing of the fabrication data, creating an automated, high performance, accurate algorithm which will minimize human error as well as improve fabrication time and lower the cost of the fabrication process.

Keywords: nuclear fuel, fuel fabrication, Westinghouse, data analysis, machine learning, neural network, gradient boosting

Acknowledgments

During this project, several people assisted by providing informative inputs and valuable knowledge, comments, long discussions and suggestions on improvements. Therefore, I would like to thank them all, especially, Juan José Casal and Magnus Limbäck.

I would also like to thank my supervisor at KTH, Professor Pär Olsson, who introduced me to this work and the team at Westinghouse, went with me through the first steps, familiarized me with the topic of the master thesis and offered his support and his constructive supervision.

A person that deserves the deepest gratitude is my supervisor at Westinghouse, Denise Adorno Lopes, who not only worked closely with me, tirelessly, supporting me in every step of the way, but also offered to me a lot more than her guidance regarding this project. I was able to acquire treasured knowledge and firsthand work experience, all provided by her, as well as hours of discussions on diverse topics, that opened new opportunities, horizons, and ways of looking at my career. I consider Denise more than my supervisor, I consider her my mentor, who's role in this project and the direction of my career path goes beyond the acknowledgements of this thesis dissertation.

Table of Contents

Abstract... 5
Acknowledgments 6
Introduction 9
1. Literature Review 11
 1.1 Fuel Fabrication 11
 1.2 Statistical Process in Fuel Fabrication (Quality Control) 15
2. Machine Learning 18
 2.1 Correlation Analysis 18
 2.2 Libraries 20
 2.3 Neural Network 22
 2.4 Gradient Boosting Machine 23
 2.5 Cross Validation 24
3. Methodology 26
 3.1 Data Analysis 26
  3.1.1 Data Collection 26
  3.1.2 Correlation Analysis for USA 26
  3.1.3 Correlation Analysis for Sweden 28
 3.2 Neural Network built and parameters 29
 3.3. Gradient Boosting used and parameters 33
4. Results and discussion 36
 4.1 Data Analysis 36
  4.1.1 Data Analysis for USA 36
  4.1.2 Data Analysis for Sweden 37
 4.2 Neural Network Model Evaluation 39
 4.3 XGBoost Model Evaluation 41
5. Conclusion 44
Appendix 45
 A1. Model Flowchart 45
References 46

Elina Charatsidou, MSc
elinach@kth.se

Introduction

Nuclear Energy is widely used by many countries and its benefits to the energy production industry are prevalent. It is more reliable and more energy dense than other energy production processes (i.e. coal, renewable energies), and therefore, it is highly profitable. However, as an industrial organization, one of the key factors for nuclear fabrication plants is aiming at reducing the cost and time for nuclear fuel manufacturing. Nevertheless, the safety requirements as well as the imposed margins for the nuclear fuel parameters ought to be in place before the fuel is distributed and used for commercial purposes. Therefore, it is important to develop processes which comply with the safety and margins required, producing safe and reliable fuel, while performing at an optimum pace for the whole fabrication processes to be rendedred finacially profitable.

A process, which both reduces the cost while promotes nuclear material recycling, is the reuse of old powders, the addback, by blending them into new ones, producing, thus, a mixture of two or more uranium powders. During this process, and in order to decide the amount of addback to be mixed, several processes are employed, taking up to several days to be completed, which hinders the fabrication process both financially and timely. This work aims at enhancing these processes by intorducing machine learning applications that will perform the calcucations and predictions now done by the manufacturers, reducing thus the human error factor, as well as the time to perform the procedures, hence, increasing the overall efficiency of the nuclear fuel manufacturing process.

This work is dealing with optimization of the nuclear fuel fabrication process. It comprises a literature study and the employment of machine learning approaches on the data obtained from the nuclear fuel fabrication factory for further data analysis and correlation assessment. Therefore, the path that was followed during this project was firstly understanding the process of the nuclear fuel manufacturing through literature review and visits to the fabrication factory in Västerås, Sweden. Secondly, a code was developed with Python, to perform machine learning and deep learning data analysis of the data at hand and to make predictions on various process parameters, that the factory desires to optimize. Hence, in the first part, the sub-processes of the nuclear fuel manufacturing were analyzed and presented in the following chapters, and in the second part the data analysis and machine learning algorithm is introduced and explained as implemented for the given data. Therefore, chapter 1 refers to the fuel fabrication process and statistical processes employed by the factory, and chapter 2 to the machine learning approach of the topic and an introduction to the two learning approaches used; neural networks and gradient boosting. To better understand the parameters implemented in the model, chapter 3 described how the correlation analyses were performed and the implementation of the aforementioned algorithms to the data at hand is demonstrated, while chapter 4 contains a discussion on the results obtained from both algorithms and comparison of them, as well as further work that can be done based on this project. Finally, chapter 5 refers to the conclusion of the presented work.

It is important here, before diving into the technical explanations, to discuss why such an approach is attractive to the fabrication process and the factory itself and how it is innovatively different compared to the qualified tests performed today in the factory during the fabrication process. For this, we will take a step back and describe the situation as it is from the fuel fabrication factory's perspective, giving to the reader as much realistic representation as possible, of the processes through which the nuclear fuel is manufactured, which are the control and process parameters, as determined by the work performed here, and which are the difficulties the factory is currently facing, thus describing the stages during the manufacturing process that this work is focusing on.

1. Literature Review

1.1 Fuel Fabrication

Figure 1 demonstrates in a schematic way, the stages and their sub-processes employed during the fabrication process. Since this work is mainly centered around the fuel pelletization process (where the addback step takes place), the rest of the stages are shown for consistency and in order to create a holistic understanding for the reader.

The input here (fig. 1) refers to enriched UF_6 in gaseous form. Particularly for Westinghouse Electric Sweden AB, the UF_6 is received already enriched from the customers who are responsible to supply the UF_6 they wish to be converted and manufactured into nuclear fuel complying with their strict specifications. The reason why UF_6 is chosen out of other chemical compounds, is due to the fact that it demonstrates physical and chemical properties which make for manageable conditions [1]. In the next stage, UF_6 is heated up with steam in an autoclave to make it gaseous for transportation to the precipitation tank. There UF_6 must be converted to UO_2, which is the chemical formation of the nuclear ceramic fuel elements. Here, three routes can be employed depending on the choice that each company opts for. It is important to state here that the properties of the converted UO_2 powder are, therefore, dependent on the chemical process employed. Hence, the selection of the conversion method influences powder specifications. The three alternative chemical processes are Indirect Dry Route (IDR), Ammonium Uranium Carbonate (AUC), and Ammonium Diuranate (ADU). The main difference between IDR and the other two processes is that the first is a dry conversion method, where decomposition and reduction of the UF_6 happens by steam and hydrogen. This method in a later stage is divided into two methods. ADU and AUC, on the other hand, are wet processes, focused on uranium compounds being precipitated from solutions in a liquid phase. [2]

Since this work is based on the manufacturing process employed at Westinghouse Electric Sweden AB, the AUC will be presented in more depth while the other two can be found in the literature for further information [1] [2]. Although as it was mentioned before, UF_6 is easier to manage, the fuel pellets are made from UO_2 powder. Therefore, AUC is a necessary intermediate step to achieve the final ceramic UO_2 form of the nuclear fuel pellets. In the AUC process, the gaseous UF_6 is dissolved into water, before undergoing a reaction with ammonium carbonate, from the precipitation the sediment is removed, dried, and subjected to hydrogen and steam treatment for successfully reducing it to UO_2. [3]

The AUC method holds several advantages over ADU and the IDR methods, including the following. Compared to IDR aqueous conversions, both ADU and AUC can process UF_6 and uranium nitrate hexahydrate (UNH) rather easily, whilst IDR can only be performed using UF_6, requiring an additional line for nitrate conversion. This demonstrates how wet processes are more efficient in treating and recovering internal scrap. Another advantage of powders converted by AUC is that they can easily be pressed to green pellets without the need for a binder, only by a slight oxidation to adjust its sintering behavior, in order to adjust the oxidation stability. Pure UO_2 is pyrophoric and will spontaneously oxidize to U_3O_8. The controlled oxidation stabilizes the powder by forming a passivating thin layer of U_3O_8 on the powder particle surfaces. [3] As it was stated earlier, the powders produced by the three different methods will end up with different powder specifications, regarding specific surface area, powder density, powder morphology, O/U ratio, U content, particle size, impurities etc.

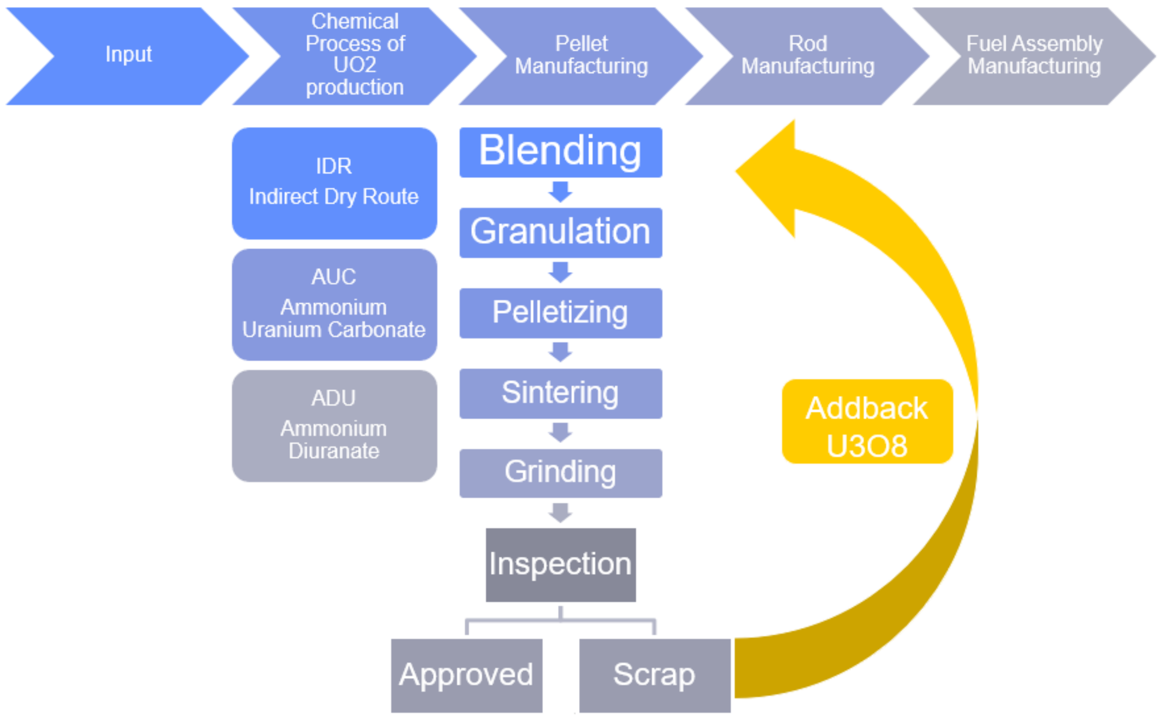


Figure 1: Schematics of the nuclear fuel fabrication process.

In the next phase, the pellet manufacturing takes place. It is this stage of the pelletization that will be adjusted according to the powder properties. After the UO_2 powder has been extracted from the AUC process, it needs to undergo several test procedures in order to be approved and moved on to the next stage. It is important to state here that ex-AUC powders are nearly unchanged in particle size and have a spheroidal particle type, therefore, being free-flowing, and do not need to pass through the pre-pressing and granulation stages, unless they are milled first in order to be mixed with Gd-sesquioxide (for burnup control), but rather after being blended they pass to the pelletization process. [4] However, for matters of consistency all the stages will be briefly discussed.

Addback: We are going to start from the last step in order to make the first step more understandable. After the pellet manufacturing process is finished, there are UO_2 ceramic pellets being rejected for not meeting the desired criteria, as well as residuals from grinding process, and therefore, regarded as scrap materials. This scrapped UO_2 is oxidized to U_3O_8 and can easily be incorporated into a UO_2 powder mix. The U_3O_8 is blended to the UO_2, not only for recycling purposes, minimizing the waste and therefore the financial losses, but also decreasing density and increasing the mechanical strength of the green pellet. Another benefit of blending the addback to the fresh powder is that is introduces porosity to the mixture. The benefit of porosity is that the pores are able to hold the fission gases reducing the fission gas release. [5]

Blending: In this stage, fresh UO_2 powder is blended with U_3O_8 powder coming from the scrap material of the nuclear fuel as described above. Additionally, other powders are added into the mix with different densities, enrichments and microstructures in order to adjust these values in the newly formed powder to improve the pellet specifications. Moreover, regarding non-AUC powders organic pore-forming agents are used to reduce the fuel density. The mixture is pre-pressed and softly blended in order to achieve homogeneity, leaving the powder properties unaffected. It is here that the material usually gets a batch identity and the blend "recipe", receives its own number for traceability. [5]

Granulation: This step applies to the IDR, ADU and pre-milled AUC-powders, since as it was explained above it is not needed for ex-AUC powders. The powder coming from the aforementioned methods has insufficient flowability and compressibility. It is easy to think of an example where one is trying to form homogeneous pellets using powdered sugar (ex-AUC powder), compared to regular crystal sugar (ex-IDR, ADU powders). Therefore, the powder needs to be pre-pressed into pellets, either with or without a binding agent. The pellets are then forced through a coarse mesh fracturing it in granules. In a process called conditioning, the granules size is unified, and a lubricant is added. [5]

Pelletizing: The UO_2 powder blends are pressed here into final shape pellets. They are fed into dies and pressed uniaxially into cylindrical pellet form. The geometry used depends on the geometry requested by the customers, and diameters can vary according to the reactor specifications the fuel is to be used in. The pressure is around 600 MPa (3-4 ton/cm^2). There are several presses available, with the most common being the rotary pellet press, typically producing up to 80 pellets per minute for AUC-powders. At these pressures, friction is high, thus causing wear in machinery, and therefore die wall lubrication is used for AUC-powders. For the rest of the powders, lubricant between granules is usually used. The pressed pellet is now called a green pellet. Green pellets usually have a density of around 6 g/cm^3. It is important to state here that wet (AUC) route green pellet density defines the final density, which is not the case for IDR. There are several control parameters when it comes to the pelletizing process and these are: the depth to which the die is filled, the intensity and duration of applied pressure and the gradual removal of the hold-down force. When pellets are placed inside the reactor core and are being irradiated, they end up being hotter in the central part therefore observing more swelling along the central axis. For this reason, dishes are introduced to facilitate the swelling issue. The dish is basically a bowl like formation along the pellet axis, as a result of the shape of the pressing punch. [5] [6]

Sintering: Sintering is just a different name for baking. The green pellets undergo sintering during which solid-state diffusion processes take place. Sintering is necessary to increase the mechanical integrity of the pellet, to increase its density and to achieve true solid ceramic form. The pellets are placed into molybdenum sintering boats and sintered in pusher-type furnaces under an argon and hydrogen atmosphere at temperature range of 1700-1800°C. The hydrogen gas is used to create a reductive atmosphere and transform U_3O_8 (from the addback) in to UO_2. A small oxygen partial pressure is beneficial. It can be accomplished by moisturizing the hydrogen or feeding with e.g. CO_2. Nitrogen can also be used to dilute the expensive hydrogen. The sintering time of the standard process in the hot zone, is around 4 hours. The whole sintering process, including debinding, reduction sintering and cooling takes considerably more time. In the furnaces used by Westinghouse Electric Sweden AB, approximately 14 hours. The final density achieved is around 95% T.D. In principle, the sintered density can be manipulated and controlled by: (a) control of the specific powder surface area (BET), (b) control of the density of the green pellet by varying the compacting pressure, (c) different amounts of U_3O_8 addback, (d) pore formers, and finally (e) control of sintering process parameters: temperature, time and atmosphere. The inherent powder characteristics are the ones affecting majorly the final pellet density and porosity. The sintering control parameters (time, temperature, atmosphere), are to be kept intact, so that the constant modification of those parameters for different powder batches will not result in loss of time and efficiency in the fuel fabrication process which will also be more prone to human errors. Therefore, it is desirable that the powder parameters variation is to be minimized so that the sintering process parameters can be applied in all batches resulting in the optimum final pellet form. Therefore, the pellet characteristics are adjusted by modifying the powder parameters. However, not all powder parameters are easily adjusted, and the easiest way to control them is by readjusting the addback amounts. [5] [6]

Grinding: Since pellet specifications give little tolerance to dimensional variations, centerless grinding is employed to produce high precision cylindrical shape pellets. Usually grinding uses a spray of water as a lubricant, which introduces moisture to the pellet. By exposing the pellets to water, any open porosity will reveal itself in the subsequent equivalent moisture test. If there is open porosity, which is not allowed in the pellet specification, the pellets will not pass the equivalent moisture test. Few manufacturing plants utilize dry grinding. [5]

Inspection: Although there are inspections and quality assurance tests performed at various stages, here, the final pellet is put under inspection in order to be approved and therefore moved to the fuel rod process or rejected and moved to the scrapped material which is later being treated in order to be reused in the blending process of a new powder mixture. Quality controls are performed regarding composition, density, geometry, micro- and macrostructure, and thermal stability to check if the produced pellets meet the requirement specifications. Regarding chemical and isotopic composition the pellets are tested for: O/U ratio and U content, enrichment in ^{235}U, additives (such as Gd), metallic and non-metallic impurities (Ca, Fe, Ni, S & Cl, F, N, C), and equivalent moisture. The pellets are also tested regarding their dimensions, and visual aspects. Different testing procedures as well as more details can be found in the next chapter and the literature. [5]

Addback treatment: As it was briefly stated above, there are grinding residuals as well as defective pellets produced during the manufacturing process which do not meet the specifications required. The UNH-route is employed for the grinding scrap since this can be mixed with alumina from the regulating wheel in the centerless grinding machine. These UO_2 pellets contain costly enriched Uranium, and from a financial standpoint it is desired to be reused into new UO_2 powders. Commonly, the scrapped UO_2 is oxidized, producing U_3O_8, and the U_3O_8 is mixed into new UO_2 powders. The pelletizing and sintering processes followed for UO_2 powders mixed with U_3O_8 is the same as if only UO_2 was to be treated. Even though the afore-mentioned method is advantageous from an economical perspective, it lacks when it comes to the powder properties, as U_3O_8 degrades the powder's sinterability and lowers its density. Therefore, it becomes apparent that the amount of U_3O_8 that can be mixed to UO_2 powders is limited due to sinterability problems. [7], However, adding U_3O_8 into the UO_2 powder has an advantage of increasing the green pellet's strength. Therefore, there needs to be a compromise when choosing the amount of addback to be mixed with the UO_2 powder, which is typically not higher than 10 wt%. [8]

The amount of addback, at Westinghouse Electric Sweden AB, is currently chosen based on numerous tests, following the trial and error procedure, where the new powder undergoes three stages, each introducing a new variable to be adjusted. Hence, **Test A**, examines the pressure-density correlation of the pure UO_2 powder. Once the pressure has been adjusted to reach the desired density, **Test B** is put into action. Here, 5% of addback is blended with the UO_2 powder and the test varies the pressure until the density reaches the required value. Lastly, in **Test C**, the amount of addback is adjusted, more is added if permissible, and pressure tests are conducted having the final amount of addback set. Although these tests produce desired results, with relatively small scrap waste, they are time consuming, therefore costly, and are interfering with the manufacturing process. If we take a rough example into account, every step test requires a day to be performed, therefore the fabrication process is postponed for at least 3 to 4 days. In the past, attempts to develop models based mainly on a few numbers of variables were not efficient in predicting with good accuracy the optimal amount of addback.

Here, it is exactly where we introduce the purpose of this work. The aim of this project is to create an artificial intelligence model, which will be able to predict the optimum amount of addback given particular powder specifications. Making such a model easily accessible, understandable, and operable by the operators at the fabrication plant, will result in saving both financial resources and time while employing a generic model that can adjust itself to different powders much easier than the tests currently performed, as mentioned above. Additionally, another advantage is that the model will always suggest the maximum amount of addback to be used while keeping the powder specifications undisturbed, which will bring about more scrap material being recycled during the addback addition. The main idea is to use the historical data of the factory to develop such a model making use of years of fabrication data that records the final density achieved for a specific powder and addback.

After the pellets have been manufactured, they move to the next fuel fabrication steps, the fuel rod and fuel assembly manufacturing. During the rod manufacturing, there are four stages describing the process, these are cladding fabrication, pellet insertion, welding, and gas injection. Here, we will not refer further to them nor to the assembly fabrication stages, since these topics are outside of the scope of this work, however, more information can be found in the literature [1]. After the rods are constructed the fuel assembly manufacturing takes place, where the skeleton of the assembly is constructed, and the rods are inserted in it, after which the top nozzle is mounted. Westinghouse Electric Sweden AB follows the schematics found in fig.1, starting by treating and converting UF_6, to assembling rods and transporting them to their destination.

1.2 Statistical Process in Fuel Fabrication (Quality Control)

Not only from an economical perspective, but from a safety standpoint as well, the reliability on the performance of the nuclear fuel is a key element in the nuclear industry. Hence, to achieve reliability and consistency, Quality Control (QC) tests are in place, operating under a Quality Assurance (QA) system. These tests are in place for a variety of procedures related to the nuclear industry. Here we are focusing on presenting the QC applied in the nuclear fuel fabrication process solely. Quality Control is performed during the fabrication process, thus actively participating, and directly influencing process parameters, in order to achieve a desired pellet, as well as in the final fabrication stage as a verification technique of the already achieved quality status. [5]

Quality Control is vital not only to the approval of a newly manufactured pellet batch, but to a wider extent affecting and influencing the whole fabrication process. QC is necessarily performed to be able to identify and compare different attributes belonging to the same product, the same attribute in different lots or products, the same process stages for different lots and or different operators handling them, and the history or trend of the product or process properties. [5]

Since the number of pellets produced for a certain costumer is large, the performance of the QC lays on statistical evaluation. Here we are only going to refer to that, stating the process used without going in depth in explaining the process itself, since this would be outside of the scope of the work. However, the process descriptions can be found in the literature [5].

The QC follows the logical flow of the fabrication process, hence, there are tests in place to be performed on the powder in its preparation and final stage, as well as to the pellet production and to the final product for verification of the obtained properties.

According to IAEA, the QC of the UO_2 fuel regards chemical properties (composition, impurities, moisture and hydrogen), physical properties (particle size, density, porosity, grain size, flowability), enrichment, dimensions and surface appearance (roughness, chipping, cracking) as well as performance tests (thermal stability, sinterability). [5] Here we will focus particularly to the processes employed by Westinghouse Electric AB Sweden and present them in a prescriptive way. The table below, describes the method used to perform the test of the specific feature, the requirements of it, the method of testing and the number of pellets tested each time.

Table 1. Quality Control for UO_2 pellets

Characteristic	Requirement	Method	Sample Size
1.Chemical Analysis			
Ag	≤ 1.0 µg/gU	ICP-MS	1 pellet/main lot
Al	≤ 250 µg/gU	ICP-MS	1 pellet/main lot
B	≤ 0.5 µg/gU	ICP-MS	1 pellet/main lot
Bi	≤ 2.0 µg/gU	ICP-MS	1 pellet/main lot
Ca	≤ 100 µg/gU	ICP-MS	1 pellet/main lot
Cd	≤ 0.5 µg/gU	ICP-MS	1 pellet/main lot
Co	≤ 6 µg/gU	ICP-MS	1 pellet/main lot
Cr	≤ 250 µg/gU	ICP-MS	1 pellet/main lot
Cu	≤ 25 µg/gU	ICP-MS	1 pellet/main lot
Dy	≤ 0.5 µg/gU	ICP-MS	1 pellet/main lot
Eu	≤ 0.5 µg/gU	ICP-MS	1 pellet/main lot
Fe	≤ 250 µg/gU	ICP-MS	1 pellet/main lot
Gd	≤ 1.0 µg/gU	ICP-MS	1 pellet/main lot
In	≤ 3.0 µg/gU	ICP-MS	1 pellet/main lot
Li	≤ 2.0 µg/gU	ICP-MS	1 pellet/main lot
Mg	≤ 100 µg/gU	ICP-MS	1 pellet/main lot
Mn	≤ 10 µg/gU	ICP-MS	1 pellet/main lot
Mo	≤ 100 µg/gU	ICP-MS	1 pellet/main lot

Ni	≤ 100 μg/gU	ICP-MS	1 pellet/main lot
Pb	≤ 20 μg/gU	ICP-MS	1 pellet/main lot
Si	≤ 250 μg/gU	ICP-MS	1 pellet/main lot
Sm	≤ 2 μg/gU	ICP-MS	1 pellet/main lot
Sn	≤ 25 μg/gU	ICP-MS	1 pellet/main lot
Ti	≤ 40 μg/gU	ICP-MS	1 pellet/main lot
V	≤ 1.0 μg/gU	ICP-MS	1 pellet/main lot
W	≤ 50 μg/gU	ICP-MS	1 pellet/main lot
Zn	≤ 20 μg/gU	ICP-MS	1 pellet/main lot
U	> 88.0 w/o	Gravimetrical	1 pellet/main lot
O/U	$2.000^{+0.010}_{-0.000}$	Gravimetrical	1 pellet/main lot
C	≤ 100 μg/gU	IR Detection	1 pellet/main lot
Cl	≤ 25 μg/gU	Ion Selection	1 pellet/main lot
N	≤ 50 μg/gU	Hot Extraction	1 pellet/main lot
F	≤ 15 μg/gU	Ion Selection	1 pellet/main lot
EBC	≤ 1.0 μg/gU	Calculated	1 pellet/main lot
H_2 content H_2O content	x ≤ 4 ppm H_2O x+ks ≤ 8 ppm H_2O	Hot Extraction	15 pellets/main lot & sub-lot
2. Isotope Analysis			
^{235}U	3.70 ± 0.05 w/o/U	Mass Spectroscopy	1 pellet/main lot
^{234}U	≤ $10*10^3$ μg/g^{235}U		1 pellet/main lot
^{236}U	≤ 250 μg/gU		1 pellet/main lot
3. Dimensions			
Diameter	8.192 ± 0.012 mm	Profilograph or Profile Projector	1 pellet/main lot & sub-lot (200)
Height	9.83 ± 0.8 mm	Profilograph/ Profile Projector/ Dial Indicator/ Sighting	2 pellets per pressing tool/ main lot & sub-lot (32)
4. Structure			
Density	10.5 ± 0.1 g/cm^3	Archimedes Method	1 pellet/main lot & sub-lot (200)
Roughness	Ra ≤ 2	Profilometer	3 pellets/main lot & sub-lot
Surface Imperfection		Visual Examination	200 pellets
Grain Size	6-25 μm	Optical Microscopy (ASTM E112)	1 pellet/main lot and pressed or sintered sub-lot
5. Mechanical Integrity			
Thermal Stability Test	x: +0.11/-0.00 g/cm3 x+ks: ≤ 0.15 g/cm3	Archimedes Method	10 ground pellets/main lot & sub-lot

The pellets that meet all the requirements within the acceptable margins are approved and are moving on to the rod and pellet assembly. The ones that do not get approved, are scrapped, and reused as U_3O_8 in new powder batches.

The parameters that describe the powder characteristic are to be used to build the machine learning model as described in the methodology, in the 3rd chapter.

2. Machine Learning

Machine learning (ML) is the framework chosen to accomplish the desired goal of building a model to predict the optimal amount of addback. Machine learning (ML) is dealing with knowledge and information one can obtain from data analysis. It merges various scientific fields like statistics, artificial intelligence, and computer science. The reason that the popularity of machine learning continues to rise is since machine learning is not as complex in its understanding and implementation as other methods used in the past, trying to tackle similar problems. Another important factor in its favor is applicability. Machine learning deals with a wide variety of data-based problems, which means it can be implemented in numerous scientific fields, computing and predicting values in a more sophisticated way than before its predominance. [9]

The goal of this chapter is to introduce the theoretical background of the algorithms employed in this work so they can be understandable by the reader in the following chapter where they will be implemented on the data at hand. Therefore, here we will present the definition of these algorithms and their principles, as well as additional methods that were used by the constructed machines in the next chapter.

The programming language used to code and create the models was Python, through the dynamic environment of Jupyter Notebook. This is an interactive environment that supports a variety of programming languages and tools, while offering a multi-user interface. In the following section the specific libraries used are described.

2.1 Correlation Analysis

Before diving into the world of gradient boosting and neural networks, it is important to understand the fabrication data, what trends it represents, the correlations and what valuable information can be obtained before using the correlations as a tool for our models. Therefore, correlation algorithms are the way to go in order to find those feature correlations in our data. Three correlation methods were considered for implementation and they will be presented here, stating their performance, pros, and cons, as well as if they were chosen or not for data analysis.

Ridge Regression: Ridge regression is a linear regression model. However, additionally to its linearity, in Ridge the coefficients-weights (w) are chosen to have the smallest possible magnitude; namely, all weights should be as close to zero as possible. This type of restriction to the coefficients is known as regularization and it prevents the model from overfitting. The Ridge Regression Model is known as L2 regularization. After a model has been created, the data at hand can be used to produce correlations and understand the features, all based on the linear Ridge model. [10]

Ridge regression is implemented in Python as: **`linear_model.Ridge`**

Lasso Regression: Similarly to Ridge Regression, the Lasso model, also known as L1 regularization, is penalizing weights to go as low as possible, however in contrast with Ridge, some coefficients are exactly zero, meaning that certain features are entirely ignored by the model. Due to this property of the model, L1 regularization can be used as a form of automatic feature selection. Having certain weights be exactly zero often makes a linear model easier to interpret and can reveal the most important features of it. [10]

Lasso regression is implemented as: **`linear_model.Lasso`**

Even though both models seem to be of use in data analysis and correlation understanding, it is important to point out that both L1 and L2 regularizations, build a linear model based on the data provided, and only then present feature parameters and correlations. This means that if a linear model is not suitable for the data at hand, both linear regression models will perform poorly. This is what happened when the data, from the Westinghouse fuel fabrication factory both in the USA and Sweden, were used to construct such regularization models and try to understand feature correlations. The linear regression models performed poorly since they both assume nothing more than linear relationships among the features, this is in agreement with the difficulty to previously built a simple model in the factory. That is to say that Lasso and Ridge models are not suitable for non-linearly correlated data as they are not good in understanding complex correlations. However, although it might seem intuitive now, and one can question the selection of these models in the first place, for data that have never been used before by any machine learning algorithm before, it is not always straightforward to know what kind of correlations one is dealing with and which are the appropriate machines one should implement in order to understand the feature relationships. Hence, it is a good practice to begin by examining the simplest models available, since in case they do perform well enough, there is no need to use complex ones, moving their way to more advanced machines with enhanced complexity that understand non-linear feature correlations as well. Therefore, through trial and error one implements different models to the data at hand, comparing the efficiency of the model as well as the complexity they are gradually introducing in order to find the optimum choice that fits best their data. This practice follows the same methodology one would implement to choose an optimum predictive model as well.

However, in this chapter the goal is only to present the correlations of the feature parameters at hand and not to predict on them, hence, creating a model in order to do that is unnecessary and introduces non-essential complexity to our code. Thus, a different way of understanding correlations should be implemented.

Pandas Correlation Method: This method computes pairwise correlation of columns, of the data existing in the DataFrame pandas format, excluding any NA or null values. [11] As opposed to the two methods mentioned above, the advantage of the Pandas Correlation Method is in its simplicity. Namely, there so no model assumed based on which correlations are produced, but rather, the correlations are representing the true relationship of the raw data. Although this method is very simple and vitally useful in understanding data before applying any kind of machine learning algorithms on them, it might me harder to use and visualize if the datapoint and features are very large in size. However, this method showed to be the optimum choice for the data produced by the fuel factory, since the number of features and datapoints is not tremendously big.

Pandas Correlation Method is implemented as: **`pandas.DataFrame.corr()`**

The pandas.DataFrame.corr() method supports several parameters, one being the method to be employed for the generation of the correlations. There are three available choices: standard Pearson correlation coefficient, Kendall Tau correlation coefficient and Spearman rank correlation. Here, we will not explain the differences of each option as they can be found in the literature [11], however, we mention that for our data the best results were obtained by the Spearman method which was used to understand the correlations as it will be shown in the next chapter.

2.2 Libraries

Scikit-learn: Firstly, we will present sci-kit learn, one of the most popular, and user-friendly libraries for machine learning algorithms. Scikit-learn is an open source Python library that is implementing a wide variety of machine learning, preprocessing, cross-validation, and visualization algorithms using a unified interface. Scikit-learn provides simple linear models as Lasso and Ridge, as well as data handling and manipulation methods as `train_test_split` which separates the data at hand in a training and validation part. The training data are used to fit to the model, so it can learn from them, and the validation data are used to perform validation analysis on the accuracy of the model's predictions. The `mean_squared_error` as a loss function in this regression model, depicting how well the model performs while training and predicting. More information on the methods and functions used with scikit-learn can be found in the literature. [12]

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import scale
from sklearn.linear_model import Lasso
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
```

Keras: Secondly, we will introduce Keras, which is the framework used to develop the deep machine learning model. The advantage of Keras over other frameworks is its simplicity in understanding and implementation, as well as its versatility to adapt to many different fields. Keras is an easy to use yet powerful deep learning library for Theano and TensorFlow, providing a high-level neural networks API in order to develop and evaluate deep learning models. [13] The simplest most used type of model for numerical data manipulation is the Sequential model, with Dense layers. ***Sequential*** is a model with linearly stack layers, and ***Dense*** are the layers in which each node receives inputs from all the nodes in the previous layer, hence connected in a densely manner. `BatchNormalization` is used to normalize and scale inputs and activations, reducing random error fluctuations. The plot function is to produce a schematic figure of the model as will be shown in the next chapter. Optimizers are methods used to alter the specifications of a neural network such as weights and learning rate in order to reduce the losses. In this project the Adam optimizer was used. More information can be found in the literature [14]

```
from keras.layers import Dense
from keras.models import Sequential
from keras.layers import BatchNormalization
from keras.utils import plot_model
from keras import optimizers
from keras.callbacks import EarlyStopping
```

NumPy: Thirdly, NumPy is the fundamental tool for scientific calculations with Python. Consisting among others of N-dimensional array objects, sophisticated functions, tools for integrating C/C++ and Fortran code, linear algebra, Fourier transform, and random number capabilities etc. Therefore, NumPy is more than a mathematical package, and more information can be found in the literature. [14] In order to use NumPy in Python, one should import the library in the code: `import NumPy as np`

Pandas: Pandas is an open source package used for data analysis and manipulation. It is a powerful and versatile tool integrated in Python. Pandas need to be installed and imported to the code in order to be implemented. More about the installation process can be found in the literature [11]. Pandas can be imported as: `import pandas as pd`

An important feature of the pandas framework, worth discussing in this section is the `pd.get_dummies` function. This function treats categorical features existing on datasets, converting them to numerical, binary, format. For example, if there are three types of furnaces employed during the fuel fabrication process, namely, A, B and C, then the dataset will have a feature column with the name FURNACE_TYPE and datapoints the values A, B, C recording which furnace was used for each powder. However, these kinds of feature representation cannot be comprehended by machine learning algorithms, which bring up the need to convert them into an understandable format for the models. Therefore, by employing `pd.get_dummies` onto the FURNACE_TYPE data, Python will create three new columns FURNACE_TYPE_A, FURNACE_TYPE_B and FURNACE_TYPE_C consisting of zeros where the particular furnace is not employed and 1 in cases which the powder was sintered using that furnace. Therefore, a dataset of the categorical feature FURNACE_TYPE, will be converted to

POWDERS	FURNACE_TYPE
POWDER_1	A
POWDER_2	C
POWDER_3	B

a dataset of dummy variables as shown below.

POWDERS	FURNACE_TYPE_A	FURNACE_TYPE_B	FURNACE_TYPE_C
POWDER_1	1	0	0
POWDER_2	0	0	1
POWDER_3	0	1	0

Matplotlib / Seaborn: Both Matplotlib and Seaborn are libraries for data visualization, creation of static or animated figures and plots, offering a high-level interface. [16] [17] They are usually imported and renamed, using an abbreviation, to speed up the process of using them while writing the code.

```
import matplotlib.pyplot as plt
import seaborn as sns
```

2.3 Neural Network

Artificial neural networks as their name suggests are inspired by real neural networks in our brain and they follow the same logic when it comes to decision making. The idea is that a neural network (NN), accepts information, the input, as knowledge it takes the weight values, and as a prediction, the output variable. Hence, all NNs do is interpret the information gained from the weights to make a sound prediction. [18]

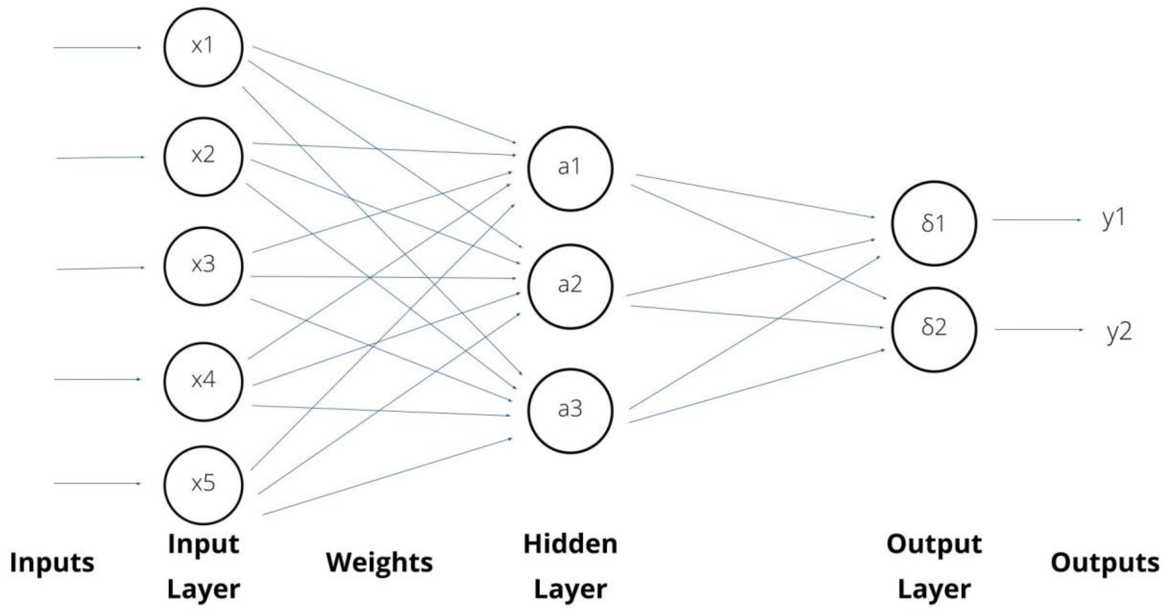


Figure 2. Schematic model of a neural network.

In order to make the description more understandable, we will focus on the basic functions of a NN avoiding in explaining details not vital for this work. For this we will use figure 2, a simplified model schematic of a neural network, without showing the equations and processes running behind. This figure is an example of a NN with 5 inputs, creating a 5-node input layer, one hidden layer consisting of 3 nodes, and finally a 2-output layer. Starting from the left side and the input layer, it consists of 5 (generally m) values, that represent the data points and are the information fed to the NN. In this work, the input data comprises the UO_2 powder and pellet specifications as well as several process parameters. This information is connected to the nodes of the hidden layer through weights, which are the values that are being adjusted while the NN is training, in order to achieve as high accuracy and precision in the output predictions as possible. Namely, the x input values are multiplied by the weights (represented here by the arrows from the input to the hidden layer) and summed up together, along with the bias, a predefined number. In the next step there is the ***activation function*** which can be a choice among a sigmoid, a RELU, and a tanh function, all simulating a variation of a step function based on which decisions are being made in order to get the output result. In case of a classification model, the output would either be a binary result, 0 or 1, or a prediction from a predefined set of output options. Since we are using a regression model (predicting addback), the output is not limited by an activation function and the result can be any real number. [19]

There are many reasons to why one should consider employing NN as a machine learning algorithm to approach data analysis problems. NNs offer many degrees of freedom meaning that they can fit with great precision to almost any data set which gives them the advantage of being a realistic representation of the behaviour of the data at hand. Additionally, NNs support multi input and output representation within the same model, which makes the code concise and the model coherent. However, with the great features of NNs come its disadvantages too. Due to the many degrees of freedom, NNs might lack in generalization compared to other machines, as they might be more prone to overfitting on the training data. Furthermore, optimizing a NN is not a straightforward nor an easy process. There are many hyperparameters that require tuning in order to find the optimum values for them, so the NN performs best. However, there are hardly any generic instructions one should follow during the hyperparameter tuning process, and most of the values ought to be adjusted following a trial and error process in which one opts from a variety of values, tries all the possible combinations and uses the one that results in the smallest error.

2.4 Gradient Boosting Machine

Like the NN, Gradient Boosting Machine (GBM) also offers the feature of performing supervised machine learning tasks, such as regression and classification. There are different variations of GBM in the way they are reaching the predictions, but they are similar in their core.

GBM produce prediction models in the form of an ensemble of weak prediction models, usually employing decision trees, by constructing the model stage-wise and generalizing them by optimizing an arbitrary differentiable loss function. An example of a GBM, can be seen in figure 3. *Boosting* builds models from individual so called “weak learners” in an iterative fashion. These individual models are built sequentially by adding more weight on instances with wrong predictions and high errors. The idea behind it is that these “difficult” instances, will be focused on during learning, so that the model learns from past mistakes. The *gradient* is employed for the minimization of the loss function, similarly to what NNs do to optimize their weights. After every training round, the weak learner is produced, predicting a result which is subsequently compared to the actual expected value. The difference between prediction and actual value constitutes the error rate of the model. These errors are employed to calculate the gradient, which is the partial derivative of the loss function, describing how steep the error function is. The gradient is used to find the direction in which the model parameters need to be modified so that the error reduces as much as possible in the subsequent training round.

The machine that was employed in this was is called XGBoost (xgb), the most popular and widely used GBM technique. XGBoost is a particular type of the Gradient Boosting strategy which utilizes more accurate approximations to locate the best tree model. While regular GBM, as it was described above, employs the loss function as the parameter to minimize the error, xgb computes second partial derivatives of the loss function, providing additional information regarding the direction of the gradients and way to achieve error minimization. Moreover, xgb employs improved regularizations (L1, L2), which enhance the generalization of the model. In order to use xgb in our model we should firstly import:

```
import xgboost as xgb
```

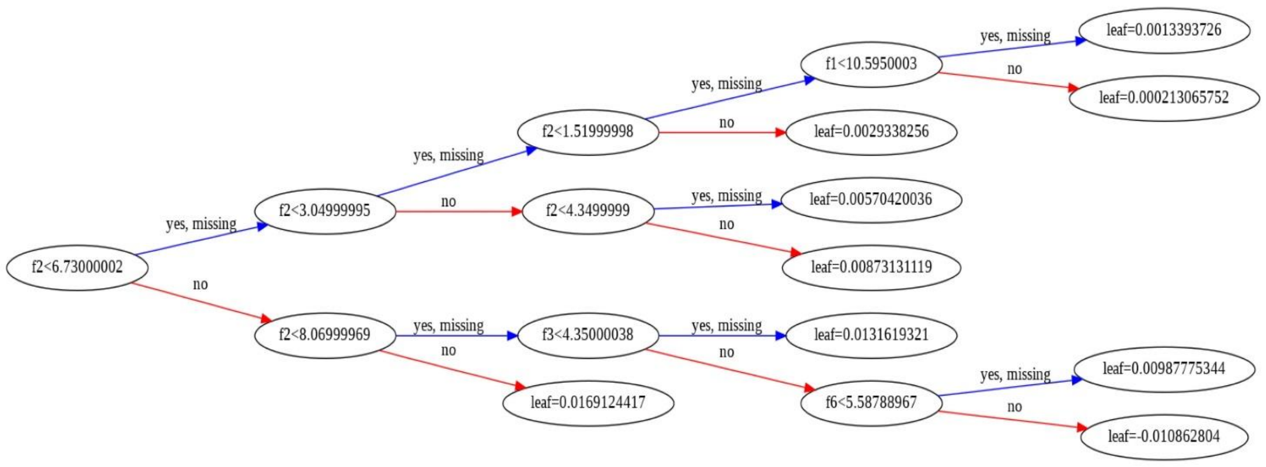


Figure 3. Schematic representation of an XGBoost model.

As it can be seen from the above descriptions, NNs and GBM have both similarities as well as differences, which should be considered when choosing the optimum model for one's data analysis. GBM is simpler in comprehension as well as in coding compared to a NN, which makes the computing time smaller while the model is running. Because of its simplicity GBM usually describes the behavior of the data in a more general approach. Lastly, an important advantage of GBM over NN models, is the fewer number of hyperparameters that need to be tuned in order to find the optimum values that perform with the minimum error. This feature makes cross-validation on GBM a much easier and more straightforward process. However, NNs are popular for a reason, as they outperform GBM in other fields such as in image or natural language processing, since GBM can only be applied to numerical data. Additionally, NNs offer a multiple output option within the same mode, which is not the case for GBM, which can only predict one output at a time. There are surely methods to employ in order to receive multiple outputs from a GBM model but these make the model less comprehensive compared to a NN, and the easier way is to duplicate the model and only change the target, getting in this way a prediction for a different output.

Therefore, it is apparent that there is a tradeoff between the model features and its performance, and one should consider this when choosing and justifying a model for a certain machine learning problem. In this work both NN and xgb were employed, using their optimized hyperparameters, and the results will be presented, compared, analyzed, and discussed in the following chapters.

2.5 Cross Validation

The way the above hyperparameter optimization and selection is performed in Python is called ***cross-validation (CV)***.

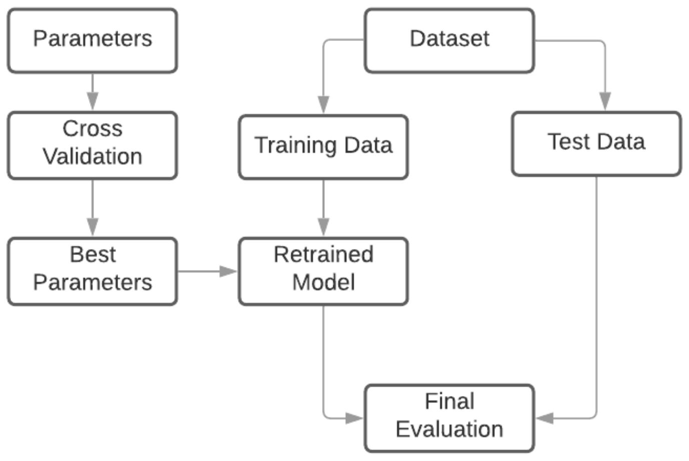


Figure 4. Cross-validation process flowchart.

CV is employed to perform tests to measure the effectiveness of a machine learning model through a re-sampling procedure. It is a technique that is used to evaluate results of statistical data analysis and how they generalize to an independent, unseen data set. The main reason for the use of cross-validation rather than conventional validation techniques such as train_test_split, is that there is only a limited amount of data available for separation into training and test sets, which would result in a loss of precision and accuracy during the prediction by increasing the bias.

For machine learning algorithms that aim at predicting results, such as a NN, a model has usually a data set of known data, the training set, based on which the model is getting trained, learning the patterns of the data, and an unseen data set, the test set, on which the evaluation of the effectiveness of the model is performed, estimating how well the model can predict results based on unseen data. CV would partition the data into subsets (k-folds), and perform validation analysis, on one set after another. Many rounds of cross-validation are performed using many different partitions and then an average of the results is taken. Cross-validation is a powerful technique in the estimation of model performance. [12]

However, along with evaluating the performance of a predefined model, CV offers the capability to run the evaluation process for different hyperparameter values. Therefore, one can use either ***GridSearchCV*** or ***RandomizedSearchCV*** to create a set of values through which the CV will run and evaluate the model (not once but k-fold times as explained above), giving the result each time and finally suggesting the optimum values for the hyperparameters. The difference between GridSearchCV and RandomizedSearchCV is that the first one offers exhaustive search over the specified parameters values, whilst the second one performs a randomized search within a specified interval. [12]

3. Methodology

3.1 Data Analysis

3.1.1 Data Collection

The data to be collected and treated are protected by an intranet within the company's intranet, and therefore, unapproachably hard to reach. The data during this work was collected manually, after access was granted to each intranet and computer separately, and the collection process was time consuming and inefficient for data processing. Hence, the work that was done and described below, has the potential of becoming automated, by building an algorithm which will collect the necessary data, append it to the dataset created to train the model, hence, constantly training and fitting the model to the newest data keeping it up to date.

The main data is spread over three different excel reports. The first one is a summary of all the pellet fabrication process parameters (i.e. press used, furnace temperature, etc.). The second one carries the batch number as a name and contains pellet specifications (i.e. final density etc.). The final document is a chemical report of the powder (i.e. impurities etc.).

Lastly, in order to have all the necessary information collected, the blend recipe is needed. Blend recipe is the amount and type of powders mixed to prepare a certain pellet batch. This information is not documented in any report, but rather in a system that had to be accessed separately.

Thus, it is apparent that the data collection currently is a tedious and a time-consuming process that requires automation in order to be rendered efficient and methodical.

3.1.2 Correlation Analysis for USA

The data provided by the Columbia Westinghouse factory in the USA, was used to understand the function of different methods such as the .corr(), while cleaning up and sorting the data from the Swedish factory. The USA data were given in a clean form ready to be used for data analysis and machine learning purposes, therefore, it was an educational path to choose, examining and optimizing the models and methods using that data, and finally employing the data from the Swedish factory and making the necessary adjustments.

Starting by using the data collected by Westinghouse in the USA, the .corr() function with a Spearman method was applied and with the help of a heatmap plot from the seaborn library, figure 3 was generated. As it can be seen from the right side of the figure, the intensity of the correlations is color coded, with the darker blue shades representing highly positively correlated values, and the lighter colors, on the opposite side of the scale, representing highly negatively correlated features.

Even though such plot contains much information and many details, it is hard to comprehend and evaluate the correlations numerically, due to the large size of features and the dense representation of the correlations. However, such figures have the ability to give a general idea to the programmer, regarding where to look for highly correlated values, and which features to focus on. Therefore, after examining figure 5, two features where selected, addback, and density, and similar correlations were drawn for them with regard to the rest of the parameters.

The actual names of the values have been masked and letters are used instead to represent the feature parameters for security reasons and information protection. The code was implemented in Python via Jupyter Notebook.

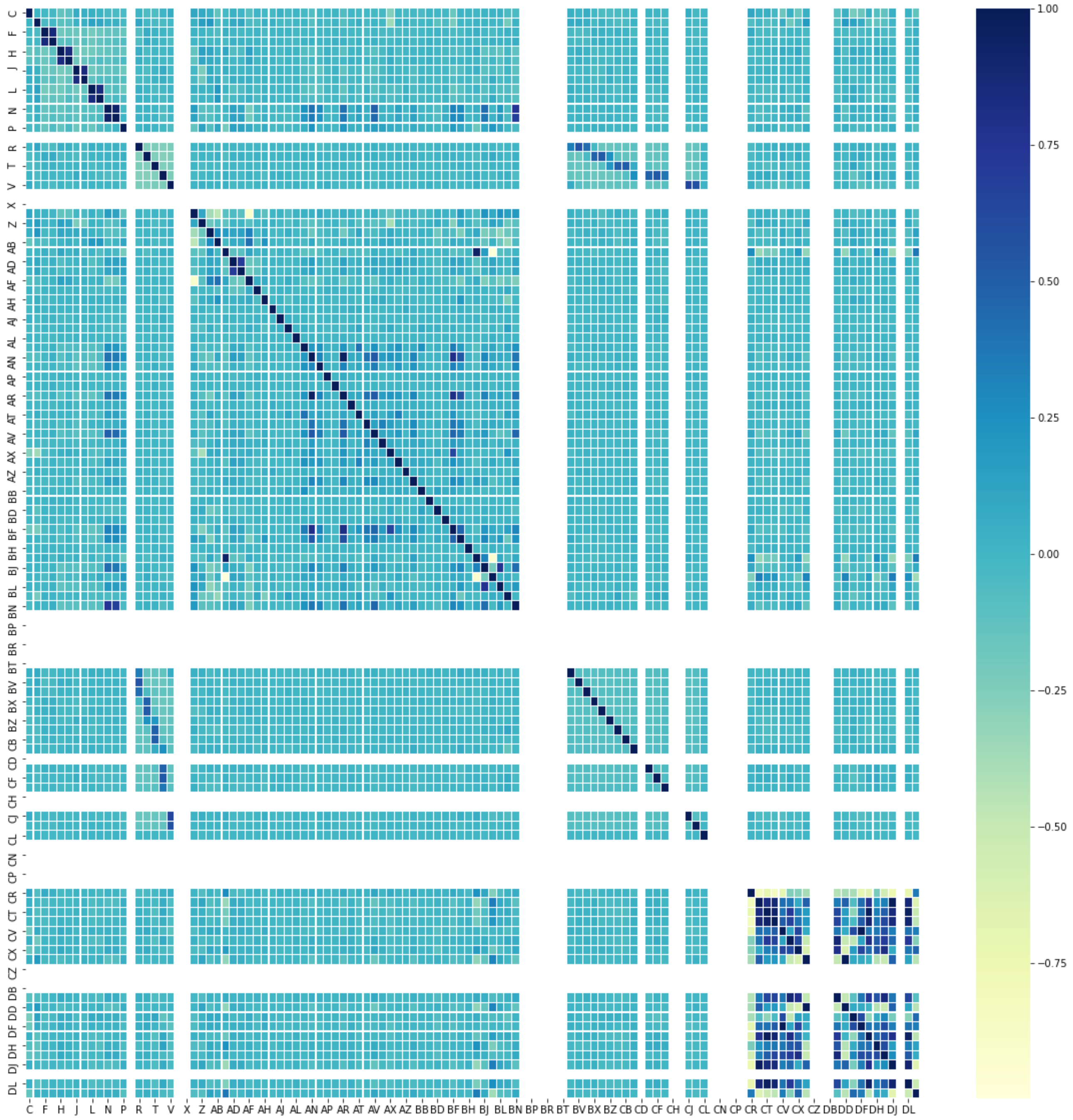


Figure 5. Heatmap plot for the USA data.

3.1.3 Correlation Analysis for Sweden

Moving to the correlation analysis performed using the data from Westinghouse Electric Sweden AB in Västerås, a similar heatmap figure (figure 6) was produced to understand the general behavior of the features based on the correlations shown in the figure. However, here the focus is mainly on the addback, density, and pressure, the bar plots of which are presented below in the Results and Discussion chapter. The symmetry depicted in both graphs stems from the fact that the correlations below the diagonal are reversed to the ones above it. This is one of the disadvantages of the heatmap plot, especially when examining a large dataset. However, more comprehensive graphs of the correlations at interest, will be presented below.

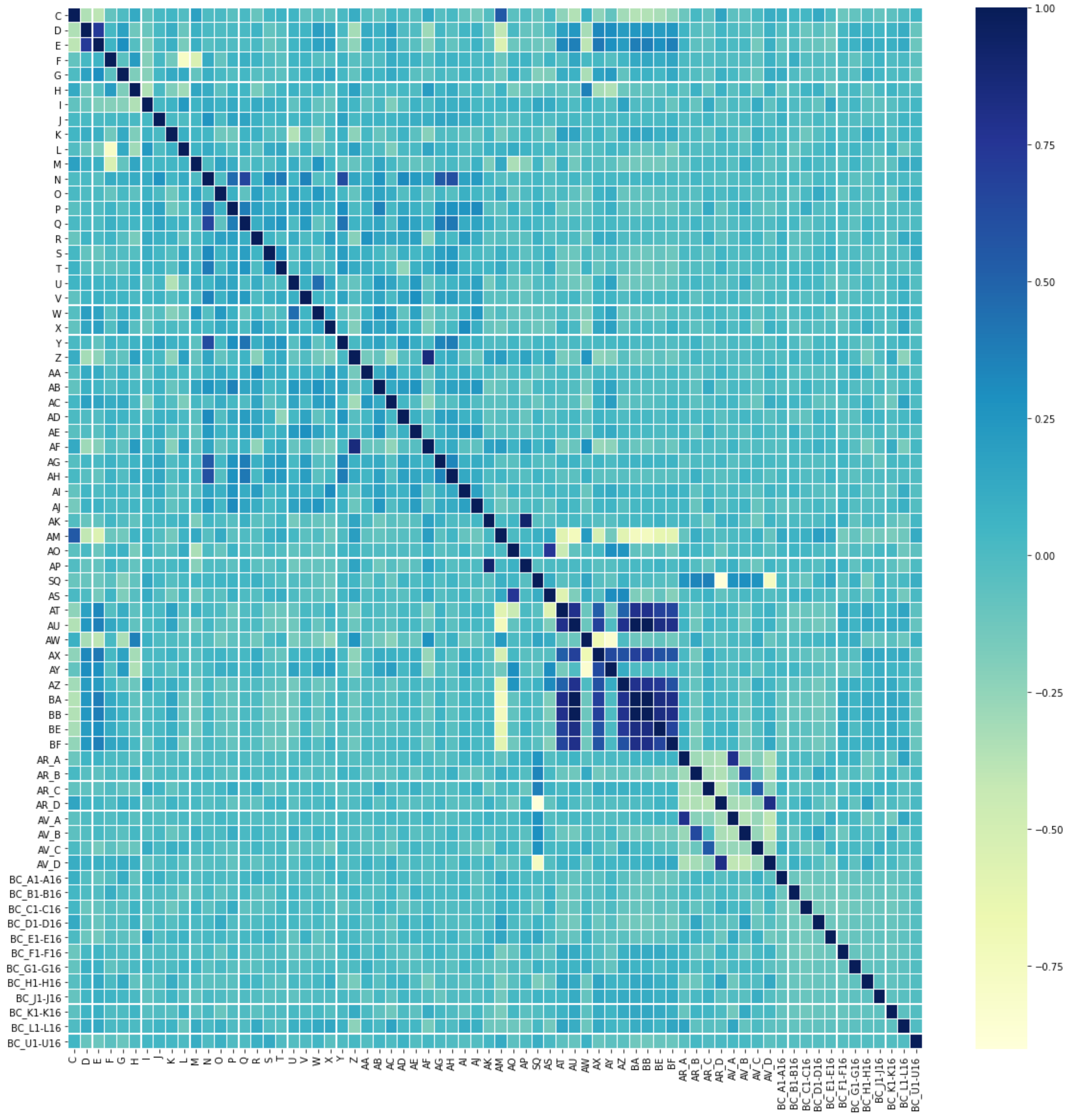


Figure 6. Heatmap plot for the data from Westinghouse in Västerås, Sweden.

3.2 Neural Network built and parameters

After the correlation analysis was performed and information was gathered about the feature relationships, the work in progressing to the next stage, where the data collected by the Westinghouse Electric Sweden AB, will be employed to train a neural network in order for it to learn the feature interactions and to predict accurate results. In this chapter a step by step explanation will be given on how the data was treated, how the model was built, as well as the functions and methods employed.

To use a dataset as an input to a neural network it needs to have a specific format. Therefore, before the data were imported in Python. They were collected in an excel sheet, with rows representing datapoints for different processes, different batches and powders, and the columns represent the features such as powder and pellet densities, pressure, addback, etc. Next, the rows containing null values are not interpreted correctly by a NN. Therefore, those rows need to be removed altogether. Lastly, some final adjustments were made, and mistakes were corrected, as while filling the excel file with the data, human factor errors were hard to eliminate. After the data was cleaned up it was ready to be used by Python. Note that the data cleaning process could have also been performed using Python, however, it was simpler and more intuitive to handle the data in the excel file before inputting it to the code, making sure it is in the shape and format required. The final data size was 882 rows (datapoints), and 57 columns (features).

The features of the data are numerical values except for three features (furnace type, press, and punch design) which are categorical type, representing different paths followed throughout the pelletizing process. For example, there are 4 types of press A, B, C, and D and the PRESS feature column is filled with these 4 letters representing which press was used for each batch. However, text values cannot be comprehended by the NN and one should modify them in a way it is understandable by the model. In this case, the features are modified to a binary format. For this purpose, we employ the pandas.get_dummies function, described in the previous chapter. Finally, after altering the categorical features into binary ones, the dataset comprises 70 features.

In the next step, the data needs to be separated into input and output values, where input values are the features of the dataset, and output are the variables to be predicted. Convention dictates, the input features be named X and the output y. Therefore, X and y arrays were constructed having 882x68 and 882x2 dimensions, respectively. The output variables are addback and pressure.

Following, the X features need to be scaled, in order to accommodate, not allowing features with high range of values to dominate over other features while predicting the results. Scaling here means that the data are centered to the mean and component wise scaled to unity variance. After the data has been preprocessed, they are split into train and test arrays, in order to keep a set of data unseen by the model, to be able to accurately evaluate its predictive capabilities. This is done by using the train_test_split function. In this work 80% of the data are used for training, while 20% is kept for evaluation.

The next step in the code, is to construct the neural network model, deciding the number and type of layers, the activation function, etc. A schematic of the model, produced in Python, is shown in figure 7 below. As it can be seen in the figure, the model comprises a 68-

dimensional input in the dense input layer, which consists of 150 nodes, followed by two 100-node hidden layers, and the output layer, which consists of 2 outputs (addback, pressure). All layers use a sigmoid as activation function, except for the output layer, which does not support an activation function option in case of a regression model. After each layer described above, there is a batch normalization of the appropriated layers, the function of which is to transform the inputs in a standardized manner, similar to what the scale function does to the X array, so they have a mean of zero and a variance of unity. They do that by keeping track of the statistics of each input throughout the training process. By taking into account the number of input and output variables, the number of weights connecting the features and nodes of each consequent layer, the total number of parameters in the model amounts to a little over 37,000.

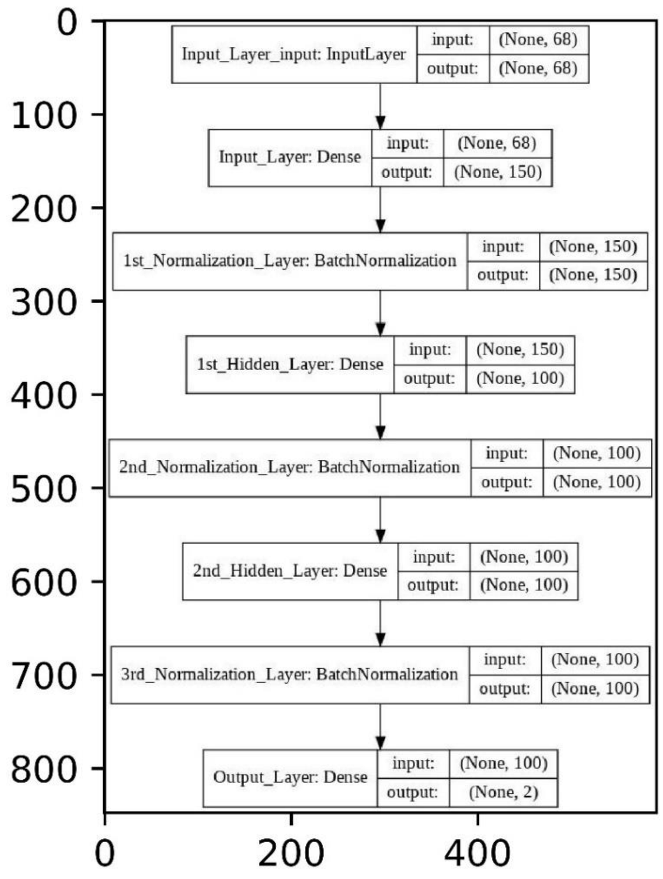

Input_Layer_input: InputLayer | input: (None, 68) | output: (None, 68)
Input_Layer: Dense | input: (None, 68) | output: (None, 150)
1st_Normalization_Layer: BatchNormalization | input: (None, 150) | output: (None, 150)
1st_Hidden_Layer: Dense | input: (None, 150) | output: (None, 100)
2nd_Normalization_Layer: BatchNormalization | input: (None, 100) | output: (None, 100)
2nd_Hidden_Layer: Dense | input: (None, 100) | output: (None, 100)
3rd_Normalization_Layer: BatchNormalization | input: (None, 100) | output: (None, 100)
Output_Layer: Dense | input: (None, 100) | output: (None, 2)
0
100
200
300
400
500
600
700
800
0
200
400

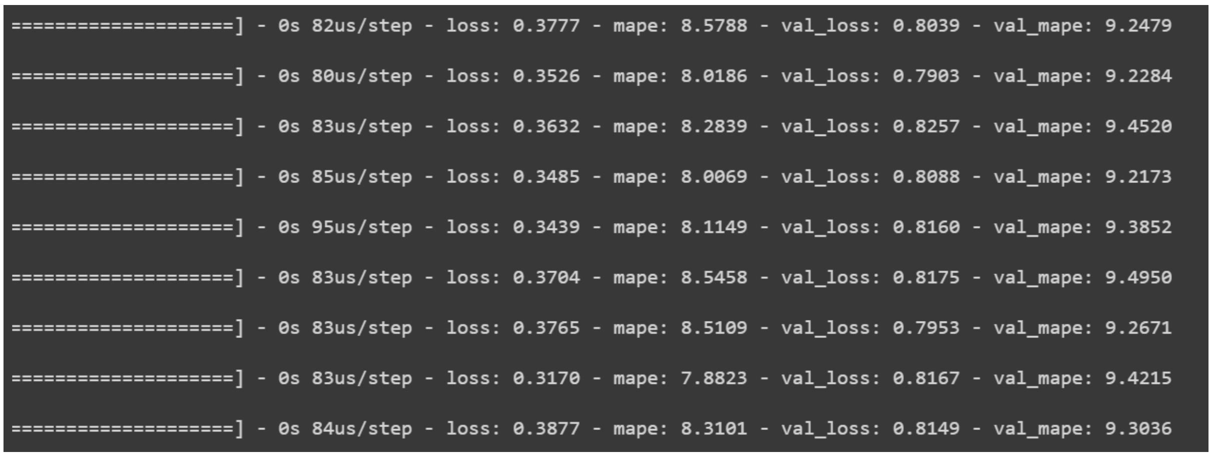
Figure 7. Neural Network Model Schematic

After the model is created, the cross-validation process takes place, in which, the parameters to be examined are defined, as well as the desired range of evaluation. Therefore, in this model, the parameters to be evaluated are the learning rate with a range of [0.01, 0.001, 0.0001], the activation function: sigmoid, tanh or RELU, the batch size: [32, 128, 256], and the number of epochs: [50, 100, 200]. After running the cross-validation, one obtains the optimum values for the above-mentioned parameters, which produces the smallest error.

The most optimum parameters were chosen to be learning rate: 0.0001, activation function: sigmoid, batch size: 32, epochs: 200.

Once the hyperparameters have been decided and chosen properly, the training of the model takes place, along with the training evaluation. For the evaluation of the training the mean absolute percentage error (MAPE) metric was used. Even though this is not the optimum choice for a metric in a regression problem of this nature, it is nevertheless the most intuitive one, giving a result similar to the accuracy of the model in a classification problem, and therefore, easier to comprehend and interpret. However, later a second evaluation metric will be used for comparison of the xgb model and the NN.

The following capture, figure 8, depicts the training process, showing the MAPE error, the loss function, and the same variables for the validation, therefore four parameters in total. As long as the MAPE and loss function reduce in value, the model ought to keep training, and when the MAPE or loss function start to increase, the training should stop, and the result before the increase should be the ones to be used.



```
===================] - 0s 82us/step - loss: 0.3777 - mape: 8.5788 - val_loss: 0.8039 - val_mape: 9.2479

===================] - 0s 80us/step - loss: 0.3526 - mape: 8.0186 - val_loss: 0.7903 - val_mape: 9.2284

===================] - 0s 83us/step - loss: 0.3632 - mape: 8.2839 - val_loss: 0.8257 - val_mape: 9.4520

===================] - 0s 85us/step - loss: 0.3485 - mape: 8.0069 - val_loss: 0.8088 - val_mape: 9.2173

===================] - 0s 95us/step - loss: 0.3439 - mape: 8.1149 - val_loss: 0.8160 - val_mape: 9.3852

===================] - 0s 83us/step - loss: 0.3704 - mape: 8.5458 - val_loss: 0.8175 - val_mape: 9.4950

===================] - 0s 83us/step - loss: 0.3765 - mape: 8.5109 - val_loss: 0.7953 - val_mape: 9.2671

===================] - 0s 83us/step - loss: 0.3170 - mape: 7.8823 - val_loss: 0.8167 - val_mape: 9.4215

===================] - 0s 84us/step - loss: 0.3877 - mape: 8.3101 - val_loss: 0.8149 - val_mape: 9.3036
```

Figure 8. Training process of the neural network.

When the values of either MAPE of the loss function start increasing, the model starts to overfit on the training data, and possibly perform worse on the prediction process due to lack of generalization. In order to prevent overfitting, the metric

```
early_stopping_monitor=EarlyStopping(patience=5)
```

was introduced, controlling the progress of the MAPE and loss function over the course of training, and in case the results were worse for 5 consecutive epochs the training will stop, and the results before the increase would be used for evaluation. One can check the training process manually adjusting the epochs; however, the early stopping monitor is an automated way of avoiding overfitting. It can be seen from figure 8, that the final error of the training process is around 8-9%, meaning that the accuracy of the model is around 91-92%.

The figures below, figures 9 & 10, depict the MAPE and loss function throughout the 200 epochs for training and validation (test) process, and it can be seen that the more the model is trained the lower the error and the higher the accuracy of it.

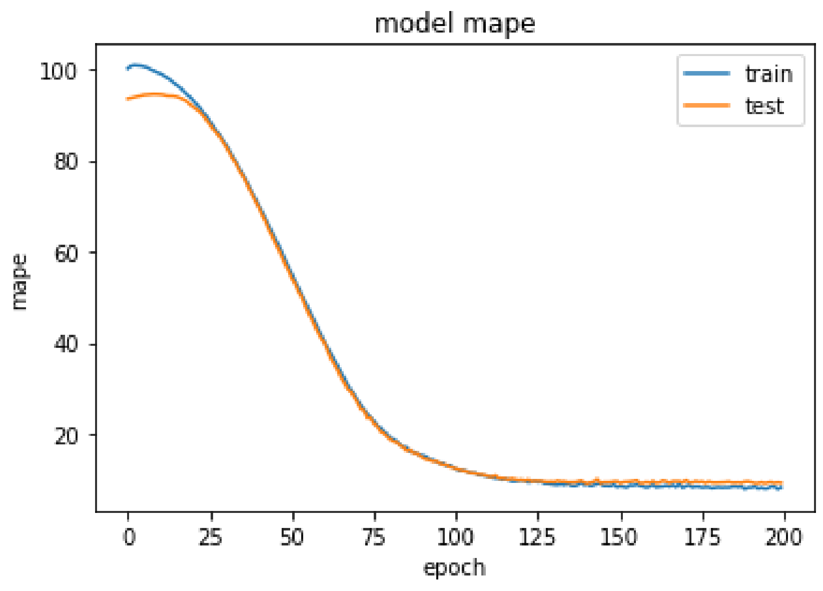


Figure 9. The progress of the mean absolute percentage error during the training and validation of the model.

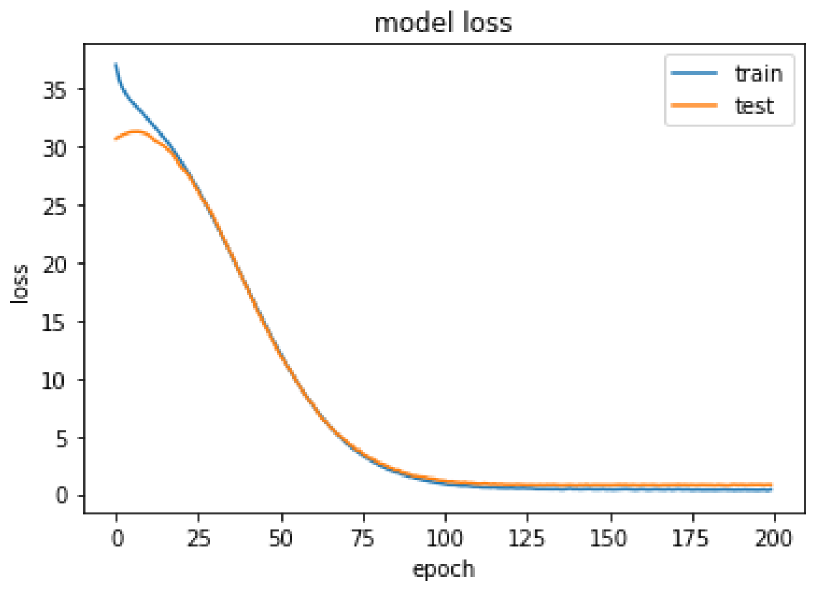


Figure 10. The progress of the loss function during the training and validation of the model.

It is important to note that the validation (test) data in both MAPE and loss function are of higher values than the training results, and this is logical since the prediction of the model on unseen data is expected to be of lower accuracy than the training process of the model itself.

In the following chapter, the prediction process will be presented, along with the results obtained for addback and pressure outputs, and an evaluation and discussion of these results will be performed.

Additionally, a flowchart demonstrating the stages employed by the model in order to predict its outputs, can be found in the Appendix (A1).

3.3. Gradient Boosting used and parameters

Since, as it was stated before, a NN is not easy to tune it terms of its hyperparameters, due to its sensitivity, and fluctuations that occur because of it, it was decided to implement a simpler yet seemingly more robust model, for our dataset: the XGBoost machine (xgb). The dataset for the xgb is the same as for the NN, having undergone the same data manipulation as well as the implementation of the get_dummies variables on it. However, one main difference between the two models is the fact that and xgb machine predicts only one result (output) at a time. Hence, the creation of two similar xgb models, one for predicting the addback and one for the pressure. The only difference in these two models is the output array, which is the datapoints for addback for the first model, and pressure for the second, respectively, and the hyperparameter tuning, which had to be slightly modified in order to perform in the optimum range for each dataset.

Once the data are in the desirable format and before moving on predicting the results, cross-validation is applied to check and choose the optimum parameters for the xgb models. For this purpose GridSearchCV was implemented and the following parameters where checked: colsample_bytree: [0.3, 0.5, 0.8], n_estimators: 40, max_depth: [2, 3, 4, 5, 10] in a 4-fold validation process. The n_estimators variable refers to the number of trees (rounds) in the model. Colsample_bytree is a family of parameters for subsampling of columns, and max_depth refers to the maximum depth of a tree and by increasing the number, one introduces more complexity to the model.

Since there are two models, one for each output value (addback and pressure), the CV was run in both and the results for the addback model were optimized for 40 estimators, colsample_bytree: 0.8, and max_depth: 10, whereas for the pressure simulating model for n_estimators: 40, colsample_bytree: 0.8, and max_depth: 3 gave the smallest error.

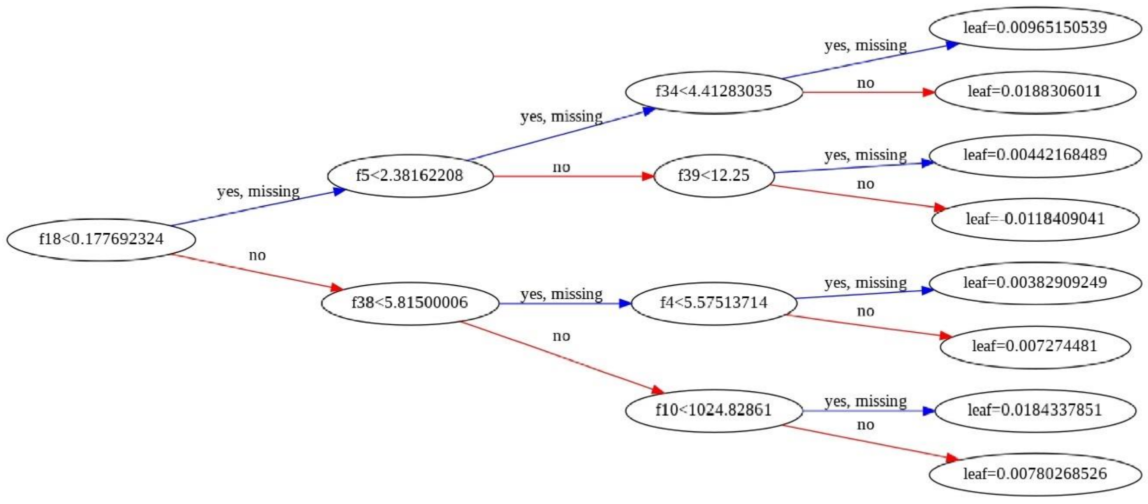


Figure 11. Gradient Boosting model (XGBoost) for pressure as an output.

Elina Charatsidou, MSc
elinach@kth.se

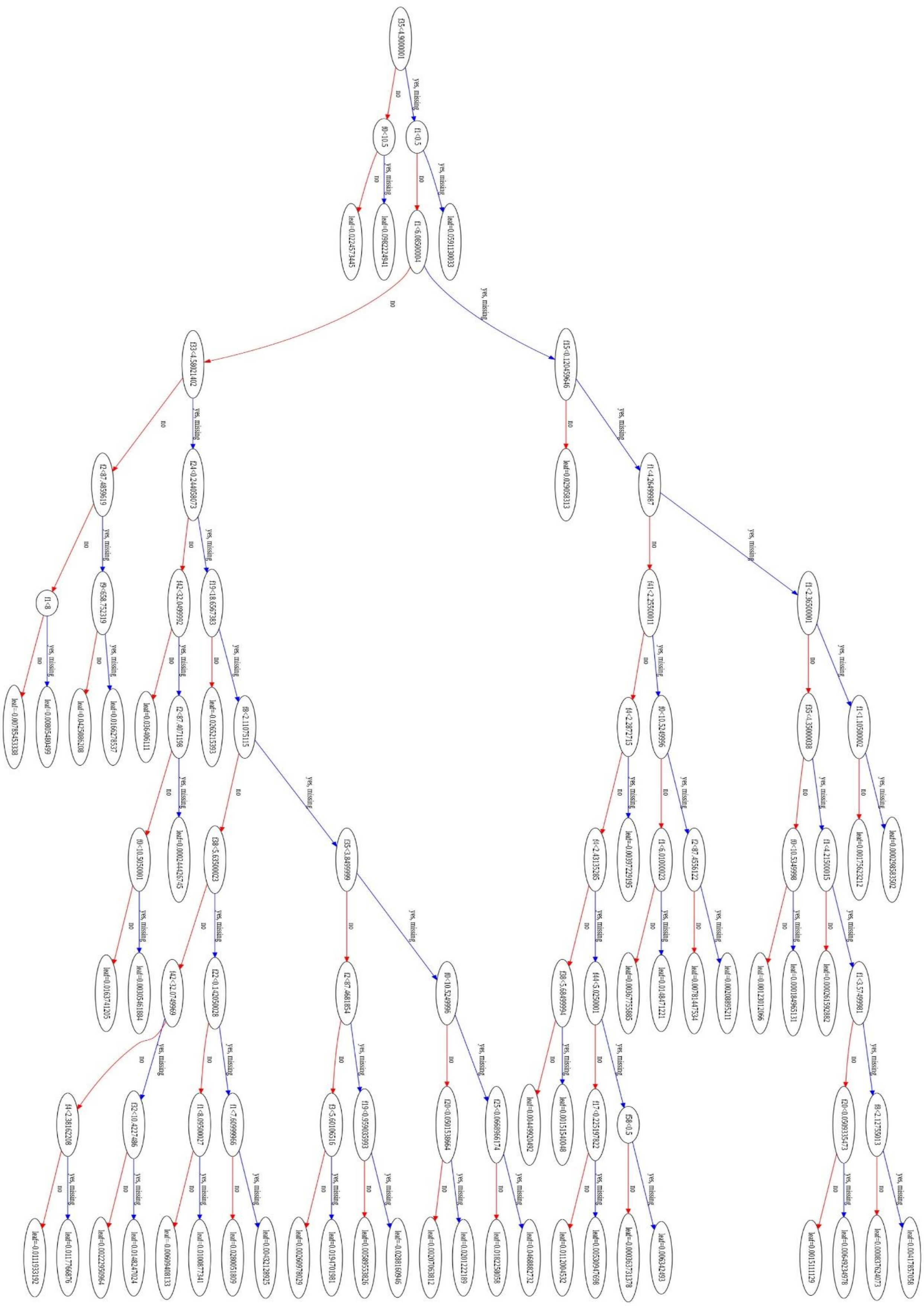


Figure 12. Gradient Boosting model (XGBoost) for addback as an output.

Figures 11 and 12 are representing the xgb models for both outputs. The max depth of the addback is 10 and this why it is “deeper”, more sophisticated in the decision-making process than an xgb model with a smaller max_depth value (pressure xgb model). However, a max_depth of only 3 resulted in a minimal error for the model with pressure as output.

In the following chapter, the prediction process will be presented, along with the results obtained for addback and pressure outputs, and an evaluation and discussion of these results will be performed together with a comparison between the two models (NN and XGBoost). It is important to note, as it was mentioned above, that although the NN produces predictions for both outputs using the same model, the XGBoost models is a single output one, meaning that the predictions that will be described below refer to each individual model, with its own individual structure, prediction pattern, and errors.

Furthermore, a flowchart demonstrating the stages employed by the model in order to predict its outputs, can be found in the Appendix (A1).

4. Results and discussion

4.1 Data Analysis

As it was described above, the collection of the data from the intranet of the company was a tiresome manual process, and as such it is difficult for it to be incorporated into the machine learning approach of the nuclear fuel fabrication process. Therefore, as a suggestion for further improvement of the efficiency, the process ought to be rendered automatic, perhaps via a code which will collect the data, transform them into the desired format, and feed them to the model. Additionally, a further suggestion would be to create a graphic user interface in order to accommodate the operators in using the code to perform the prediction tasks in an intuitive way without prior knowledge of machine learning or coding.

4.1.1 Data Analysis for USA

The two following graphs represent the relationship between addback and density, as well as among the rest of the features (figure 13, 14). As it can be seen addback and density are positively correlated with each other while almost all other features are correlated with both addback and density. Exactly this characteristic, many features contributing in the correlations without any significantly high or low values, is what makes feature selection hard. Meaning, that it is not easy nor trivial to be able to determine a few features advocating that they are the main ones contributing into the process of the nuclear fuel fabrication, since there are numerous features that have slight contributions individually.

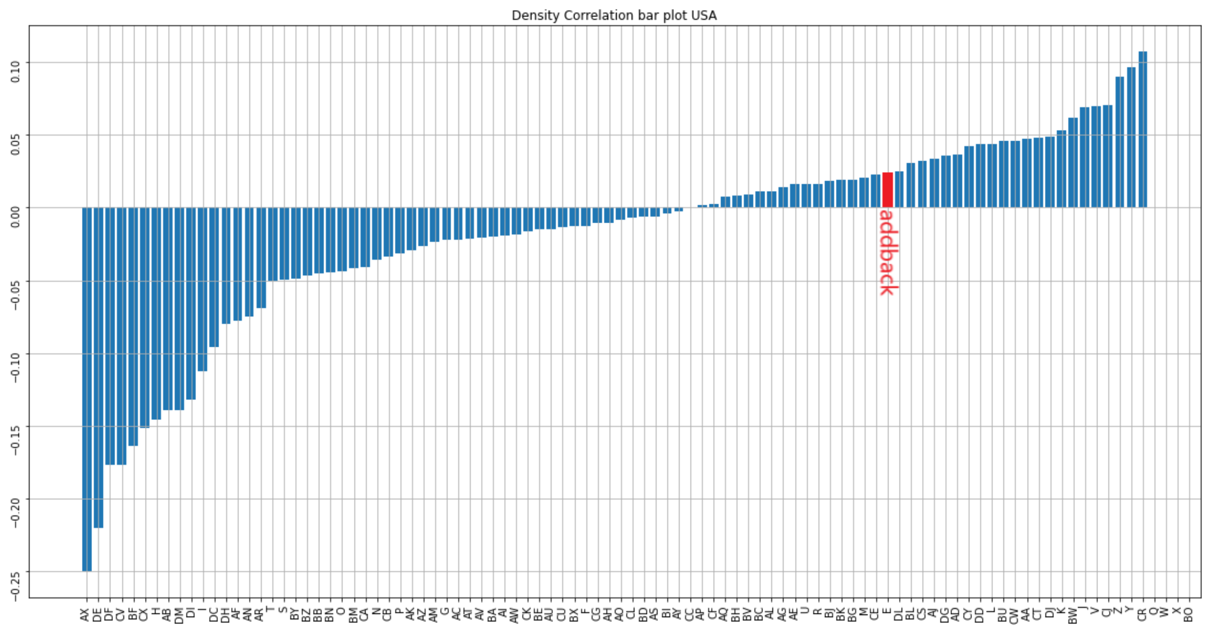


Figure 13. Correlation graph for density from the USA data.

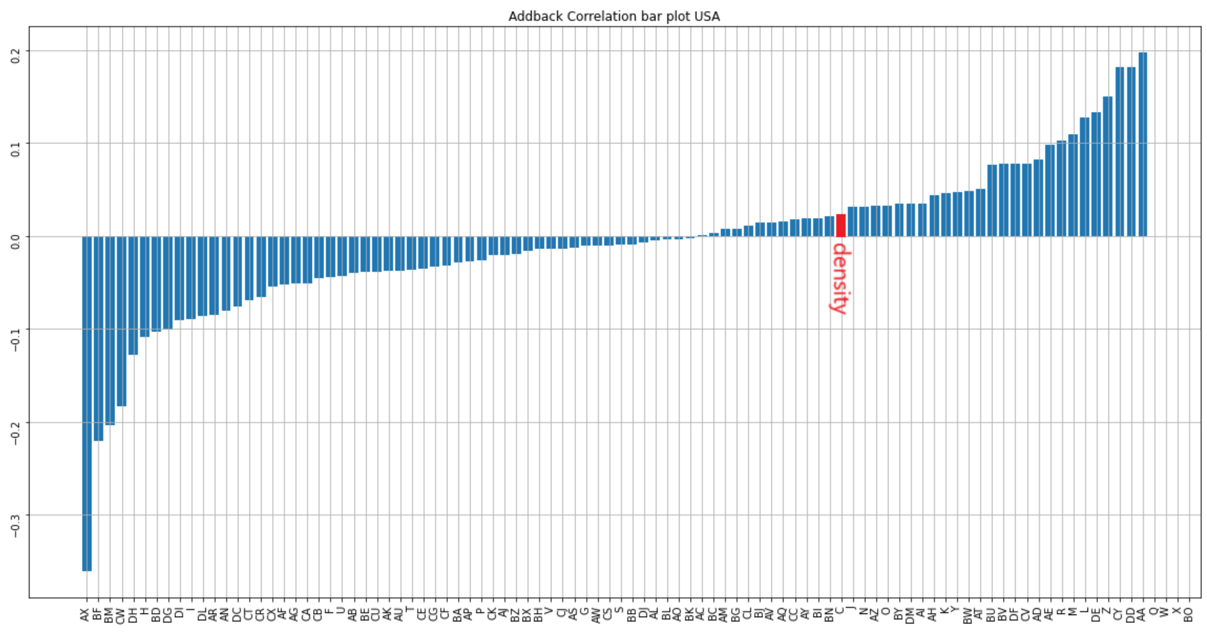


Figure 14. Correlation graph for addback from the USA data.

4.1.2 Data Analysis for Sweden

Similar figures were made for the Swedish factory, focusing on the two outcome parameters, addback, and pressure, as well as the density, which is one of the major features of the nuclear fuel pellet.

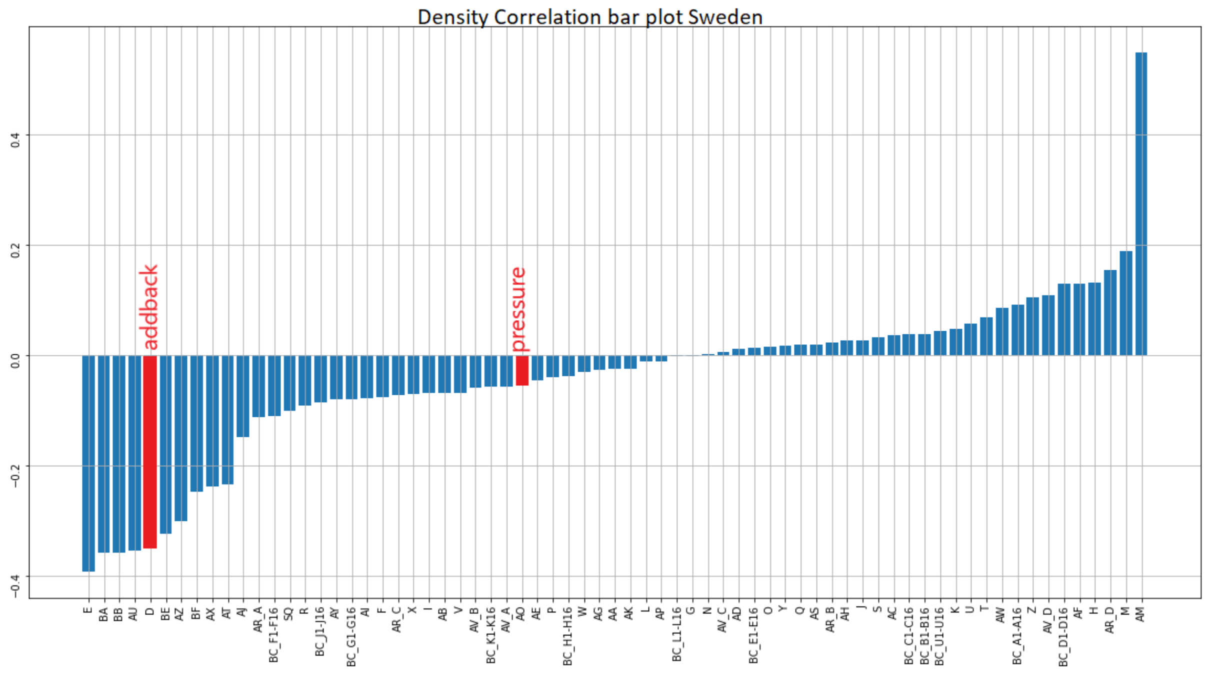


Figure 15. Correlation graph for density from the Västerås data.

From figure 15, we get the information that final pellet density is negatively correlated with addback (correlation around -35%), and negatively correlated with pressure as well (around -5%). However, figure 16 shows that addback is positively correlated with pressure (correlation around 14%), while the addback-density correlation is intuitively the same.

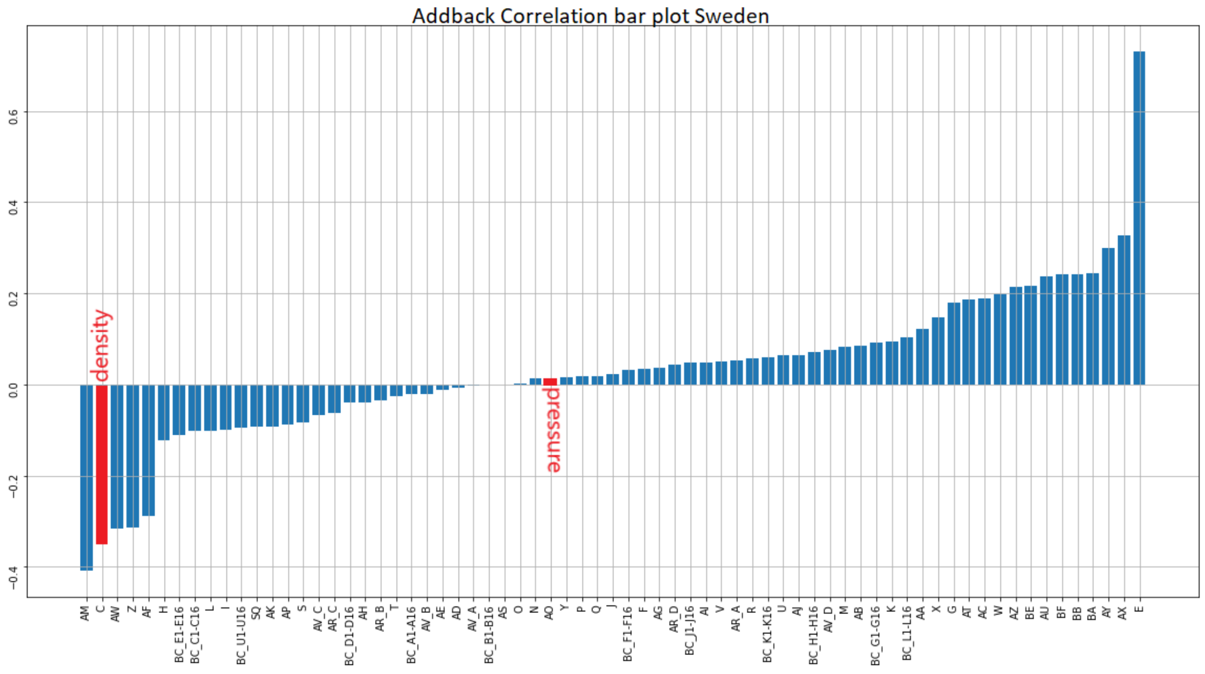


Figure 16. Correlation graph for addback from the Västerås data.

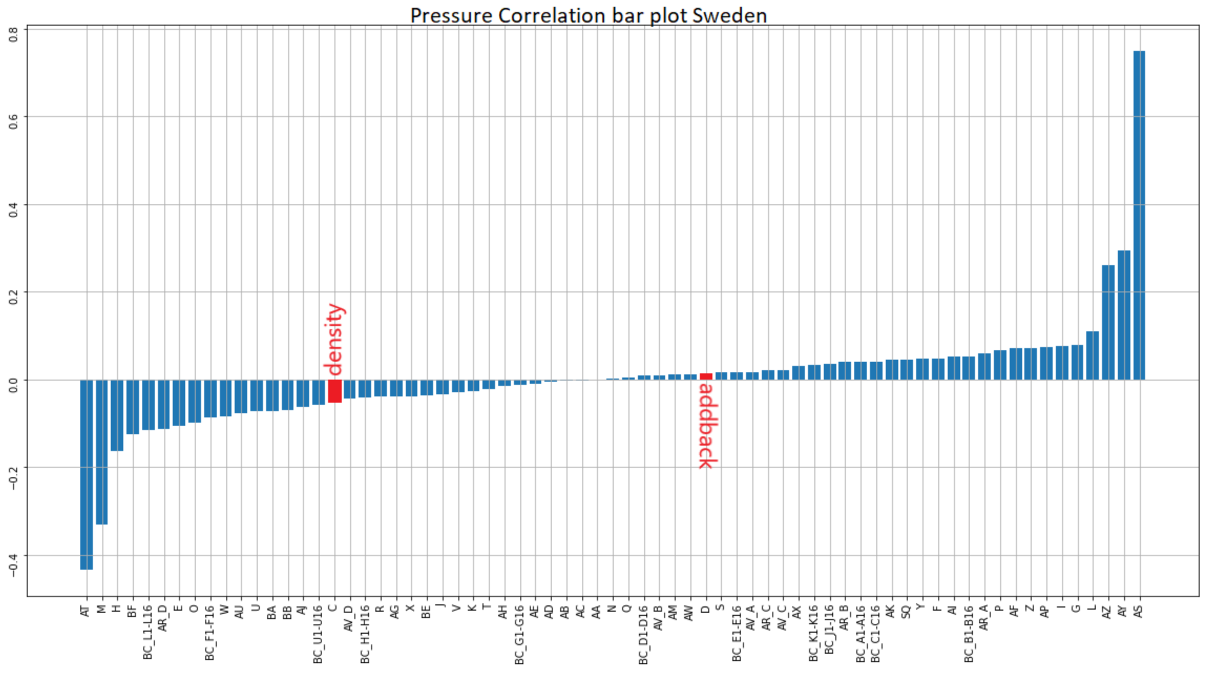


Figure 17. Correlation graph for pressure from the Västerås data.

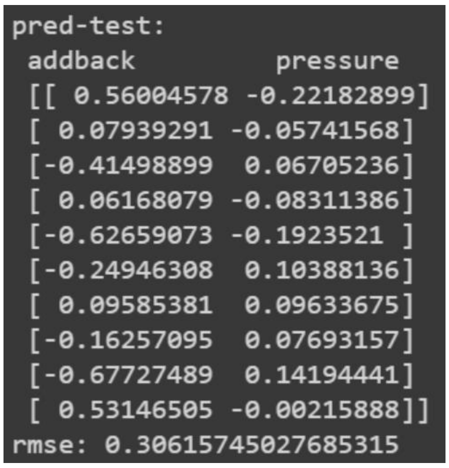
Finally, figure 17, depicts the correlations of pressure between density and addback, and the values of the correlations are stated above.

From a physics standpoint, evaluating these results is of utmost importance. The fact that addback and density are negatively correlated, verifies that the more U_3O_8 is added to the fresh UO_2 powder the more the density is lowered. As a result, because of the lower density, lower pressure needs to be applied to give the same outcome (since density is also depicted of having negative correlation with pressure). However, pressure is positively correlated to the addback, meaning the more U_3O_8 added to the powder mixture, the more pressure needs to be applied, but this correlation is a weak one, as it can be seen from figures 16 and 17.

The data analysis of both datasets provided valuable information on how both models will perform, as well as differences and similarities. Although, the similarities in the correlations are supported by the physics behind them, the differences stem mainly from the different fabrication processes that the two factories are employing. Therefore, it was expected that the two datasets will not produce similar results, and therefore, the same NN or xgb model cannot be implemented for both, rather model adjustments are necessary to accommodate the datasets.

4.2 Neural Network Model Evaluation

After training and evaluating the NN model, it is time for it to perform the task it was designed to do, that of predicting the addback and pressure values for unseen data. For this purpose, a list of values was created in such a way that the code chooses 10 random values among the test values not used throughout the training process. Hence, 10 random values were chosen for addback and 10 for pressure. Afterwards, the `.predict()` method was used on the model and 10 addback and pressure predictions were generated by the neural network. Although for security reasons the actual data will not be presented here, the difference between the prediction and the real value is shown below (figure 18), and the error used to represent this difference is root mean square error (rmse), which is the standard deviation from the residual. This error metric is used here in order to perform sound comparison between NN and xgb models, since XGBoost employs rmse for its error evaluation.



```
pred-test:
 addback         pressure
 [[ 0.56004578 -0.22182899]
 [ 0.07939291 -0.05741568]
 [-0.41498899  0.06705236]
 [ 0.06168079 -0.08311386]
 [-0.62659073 -0.1923521 ]
 [-0.24946308  0.10388136]
 [ 0.09585381  0.09633675]
 [-0.16257095  0.07693157]
 [-0.67727489  0.14194441]
 [ 0.53146505 -0.00215888]]
rmse: 0.30615745027685315
```

Figure 18. Residual of prediction and real value and rmse.

The root mean square error for the neural network model is around **$rmse_{NN}$ = 0.31**. The use and this value will be more apparent in the following paragraph where the comparison of it and the rmse from the xgb model will be performed.

After predicting the output variables, the code is designed to plot the prediction compared to the real values for both addback and pressure, and the figures are presented below (figure 19, 20).

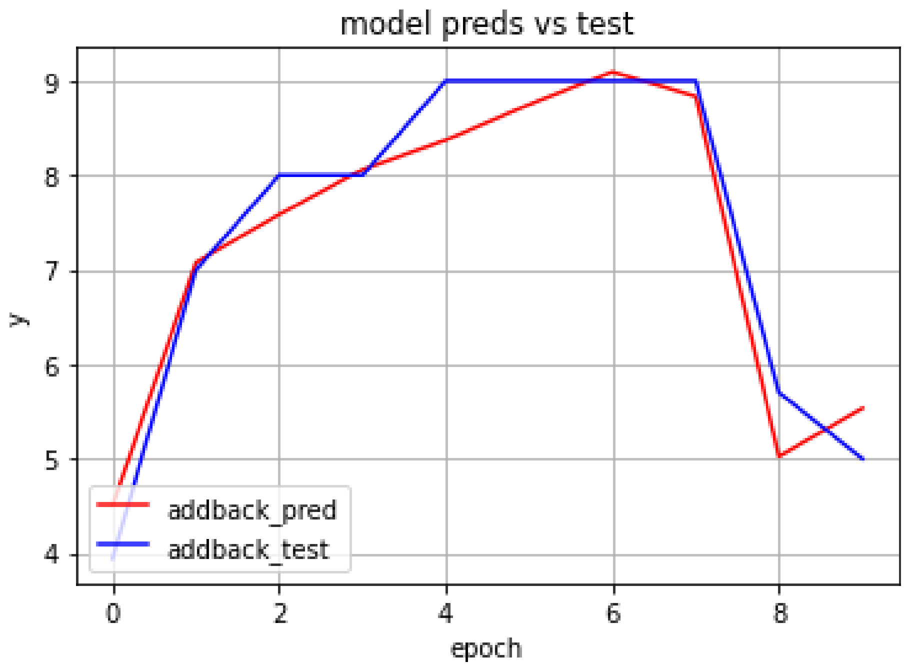


Figure 19. Addback prediction results of the NN model comapred to the actual test value.

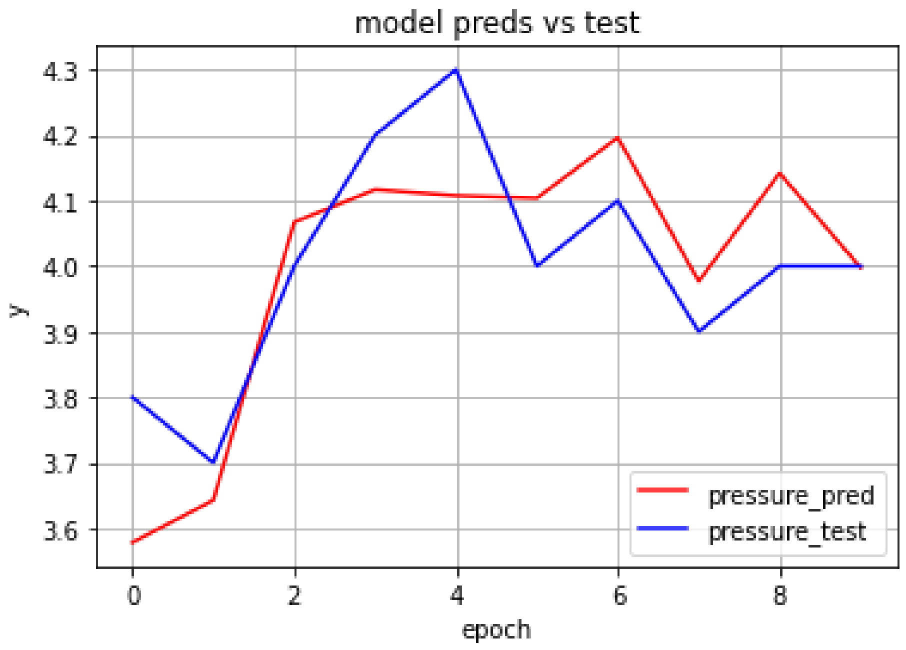


Figure 20. Pressure prediction results of the NN model comapred to the actual test value.

As it can be seen from the above graphs, the model performs accurately on predicting both outputs, with an accuracy reaching above 90%. The preditions for addback appeare to be more accurate, relatively speaking, than the predictions generated for pressure, which is most probably assosiated with random noise and fluctuations of the datapoints. However, both predicitions supersede 90% accuracy. As a comparison the USA model used for the dataset from the Columbia factory, has an accuracy which amounts at 85%. The high accuracy and performance of the NN model is a result of numerous CV runs and trial and error processes, during which an abundance of hyperparameter values were tested and compared for their model performance before the optimum ones were chosen. However, it is important to state that since the model predicts the addback, which is a parameters that fluctuates a lot, compared to final density for example, the random fluctuations of the model are hard to avoid and can only be minimized to an extent by using the same random seed, or running the model multiple times. Another solution to the fluctuations would be to use as an outcome parameter a more stable feature, such as the final pellet density, and reaching to the desired amount of addback through mathematical formulas connecting the two variables. This is a further enhancement that can be performed to the current work.

4.3 XGBoost Model Evaluation

As it was stated above, since the data handled by this work are mainly numerical, apart from several categorical features that had to be forced into numerical values, the idea arose of creating a second, rather simplified model to a NN, in order to compare the two performances and evaluate the pros and cons of each model and code, as well as suggest the most efficient method to be implemented by the factory.

Hence, the decision on employing XGBoost, a promising gradient boosting model, to perform the same task as the designed NN does. It is important to remind the reader that an xgb is fundamentally a 1-output model, therefore, two similar models were created to predict addback and pressure, respectively. Since this is a regression problem, the xgb model that was used is the `xgboost.XGBRegressor()`.

Similarly to how the data was treated to be used for a NN, after importing all the necessary libraries, and the dataset in a pandas format, and applying the .get_dummies function on them, the feature array was designed with all the features, except the one we want to predict, and the output array was a single column matrix, of the feature to be predicted. Next, the X and y arrays were split into train (80%) and test data (20% of the total database), exactly as in the NN. After the data handling was complete, the arrays were transformed into `xgboost.DMatrix()` arrays, having X as the data parameter and y as label, a specific type of matrix, that xgb models are employing for further data manipulation. Cross-validation was employed afterwards to find the optimum values of the hyperparameters, for which the error is the minimum one. Hence, after running the CV, it was decided to employ, for the addback model, n_estimators: 40, which is the number of trees to be built, and max_depth: 10, which represents the depth of the trees, and the colsample_bytree_vals: 0.8, which is the subsample ratio of columns when constructing each tree. For the pressure model, the values are the same except for the max_depth which was set to 3, as this resulted in the minimum error. After the hyperparameter tuning was complete, the xgb regression models were trained, by fitting to them the training features and labels, before selecting 10 unseen feature datapoints and predicting on them, while evaluating the predictions. Again here, the residual between prediction and actual test value is presented in the table below (table 2).

Table 2. Residual of the prediction and the true test value, from the xgb models.

Addback prediction-test	Pressure prediction-test
-0.029603	2.479603
0.1029427	5.1170573
0.3209983	1.1490017
-0.078239	6.078239
-0.371096	9.281096
-0.461676	9.461676
-0.208986	9.208986
-0.186389	9.186389
-0.1289197	6.2289197
-0.184021	9.184021

After predicting the addback and pressure, the error of the predictions, rmse, was calculated, and amounted at around **rmse_{add} = 0.10** and **$\text{rmse}_{\text{press}}$ = 0.09**. Additionally, in order to create a better representation of the errors, both models were designed to produce graphic plots of the predictions compared to the actual test values, similar to those of a NN, described above.

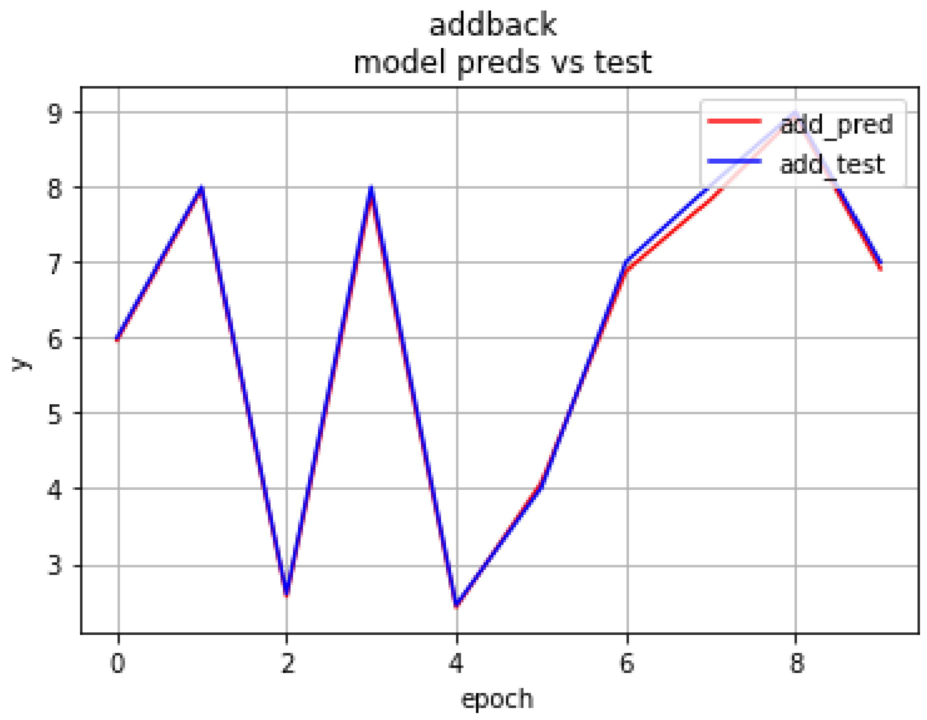


Figure 21. Addback prediction results of the xgb model comapred to the actual test value.

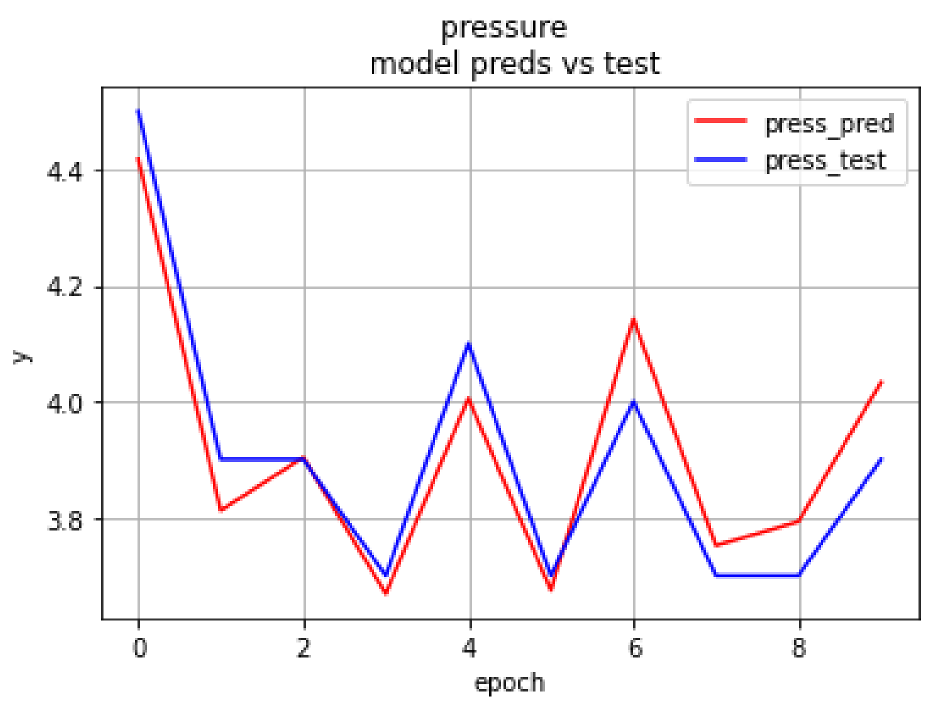


Figure 22. Pressure prediction results of the xgb model comapred to the actual test value.

Compared to the last paragraph and the results obtained from the NN model, the error of which was around **rmse$_{NN}$ = 0.31**, it is apparent that both XGBoost models perform with a much lower error than the NN, meaning that the predictions they generate are more accurate and reliable. Additionally, due to the limited number of hyperparameters in an xgb model, compared to a NN, the fluctuations are fewer, and the consistency of the results is better. Regarding the script, an XGBoost model was easier to comprehend and implement, smaller in size and less variables to be used, as well as faster in utilizing computing power to generate results. The script for a NN model requires knowledge of deep machine learning algorithm configuration, and it is more complex and longer. Additionally, both training and predictions were longer in performance, compared to the XGBoost. However, the big advantage of a NN is the fact that it produces multi-output results within the same model, making the evaluation and comprehension of them more intuitive, in contrast to the xgb model which fundamentally produces a single-output prediction, and thus, needing two models to perform the same task as the NN does.

5. Conclusion

During this project, data, from the Westinghouse Electric Sweden AB nuclear fuel fabrication factory, were collected, analysed, handled and manipulated in order to exctract vital information regarding the fabrication process. However, the data were collected manually from within the company's intranet, and many of them in printed format, rather than electronic. Hence, a great deal of time of this project was devoted to data treatment and cleaning, which coresponds well to a real problem data scientists are facing. Therefore, it is essential to point out once again, that automating the data collection and cleaning process would introduce a great deal of time saving and reduction of the human error factor, as well as render the machine learning process autonomous and much more efficient.

As a conclusion from the implementation of machine learning techniques to the problem this work is dealing with, many valuable lessons were learnt, and will be useful to adhere to in the future when performing similar tasks. Although neural networks are powerful deep learning algorithms, which are highly complex, and capable to adapting in a great variety of fields and data formats, they do have disadvantages stemming from their very own sensitivity and complexity attributes. Hence, for a NN model to be sensitive to changes, means that it can be employed for a plethora of datasets, but being prone to fluctuations and instabilities at the same time. This problem is treated by performing cross-validation, as well as having as large a database as possibly available, in the best format available to be utilized by the code.

However, when databases comprise a limited number of datapoints, with several missing values and other problems that require data cleaning, the final data is a subset of the initial one, cleaned and complete in every row and column, even smaller than the one it stemmed from. Thus, in such cases of simplified datapoints, one should consider using a machine learning model that is simple, which will represent the feature's behaviour in an optimum way, without introducing unnecessary complexity to the model.

This is the justification for why xgb models were used in this work as well, and after training and predicting outputs, it was verified that the non-excessive complexity of the dataset at hand, resulted in a more efficient predictive performance, with the minimum error recorded.

Besides the performance of the model in predicting labels based on unseen data, the work also concluded that employing a less complex model, resulted in shorter computing time, as well as to a more concise model with fewer lines of code.

To conclude, both neural networks and gradient boosting models are high-end machine learning algorithms with outstanding performances and impressively consistent results. Nevertheless, a big part of coding is a trial and error process in which many small modifications are made in order to adjust the model to its optimal settings. This process can be rendered easier by employing models best fit for the given dataset. However, the model choice is a trial and error process itself as well, as one lacks the necessary information to infer on which model would give the best results, prior to applying the models on the datapoint and evaluating and comparing the results.

The models designed in this work will be further adjusted by the Westinghouse factory, and will be employed during the manufacturing process, mainly throughout the pelletization stage, in order to predict the optimum amount of addback for any given fabrication dataset. Lastly, this work is leading the way towards a more concise digitalized data collection process, which not only be used for machine learning purposes, but also will make the fabrication data more robust, comprehensive and easily accessible for various statistical and computational tools in the future.

Appendix

A1. Model Flowchart

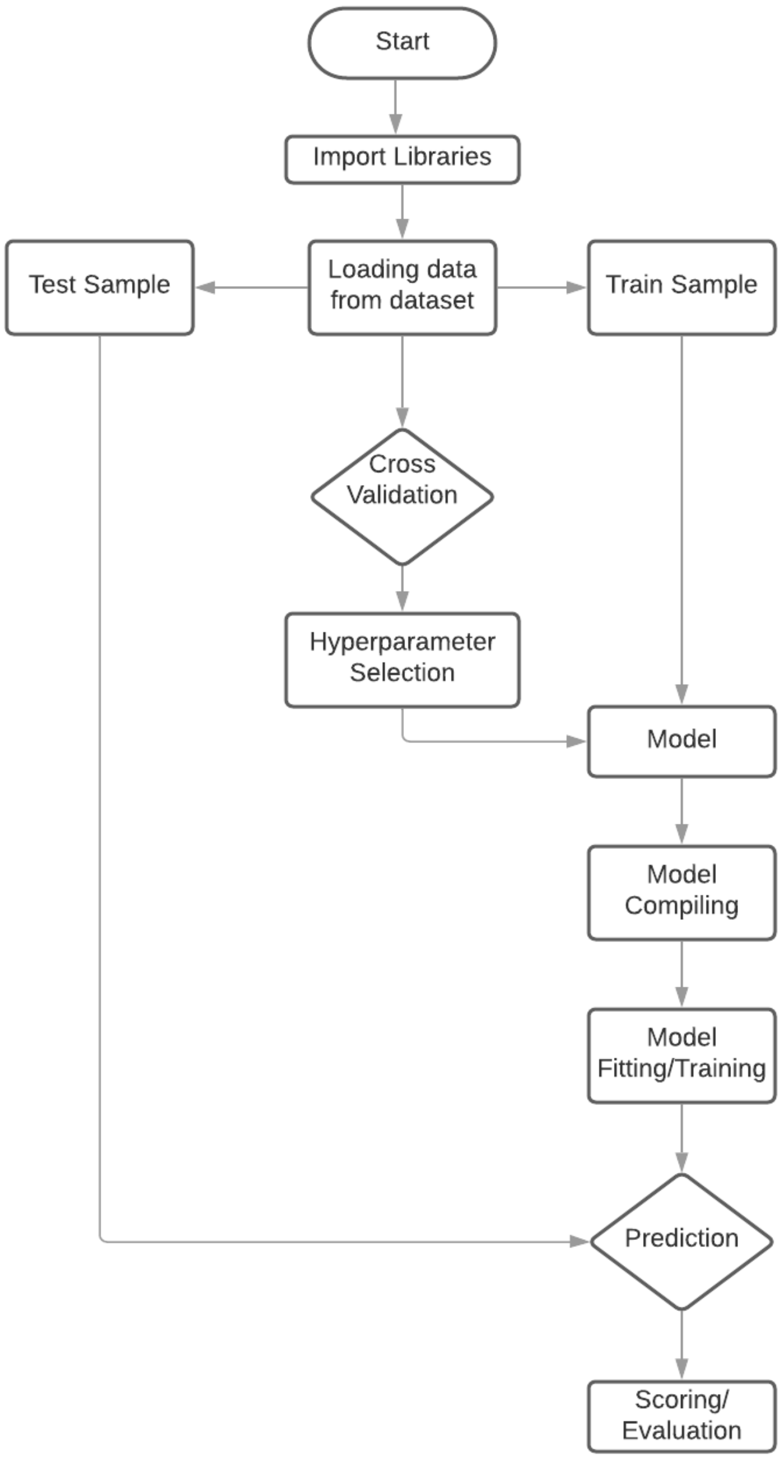


Figure A1: Flowchart of the model script.

References

Chapter 1

[1] World Nuclear Association. https://www.world-nuclear.org/ (Accessed June 7, 2020)
[2] Sven G. Brandberg (1973) The Conversion of Uranium Hexafluoride to Uranium Dioxide, Nuclear Technology, 18:2, 177-184, DOI: 10.13182/NT73-A31286
[3] H. Stehle, H. Assmann and F. Wunderlich (1974) Uranium Dioxide Properties for LWR Fuel Rods, Kraftwerk Union A G, Erlangen, Germany
[4] Y.W. Lee and M.S. Yang, Characterization of HWR Fuel Pellets Fabricated Using U02 Powders from Different Conversion Processes, Quality Control Department, HWR Fuel Division, Korea Atomic Energy Research Institute, Daejeon
[5] Guidebook on Quality Control of Water Reactor Fuel, Technical Reports Series No.221, International Atomic Energy Agency, Vienna, Austria, 1983
[6] KTH Royal Institute of Technology. Janne Wallenius. (2020). Nuclear Fuel Cycle. Lecture Slides. Fuel Fabrication: Powders To Assemblies [Online]. Available: canvas.kth.se
[7] Kun Woo Song, Keon Sik Kim, Young Min Kim, Youn Ho Jung (1999) Sintering of Mixed UO2 and U3O8 Powder Compacts, Advanced LWR Fuel Development, Korea Atomic Energy Research Institute, P.O. Box 105, Yusong, Taejon 305-600, South Korea
[8] Kun Woo Song, Keon Sik Kim, Ki Won Kang, Youn Ho Jung (2003) Grain Size Control of UO2 Pellets by Adding Heat-Treated U3O8 Particles to UO2 Powder, Advanced LWR Fuel Development, Korea Atomic Energy Research Institute, P.O. Box 105 Yusong, Daejon 305-600, South Korea

Chapter 2

[9] Andreas C. Müller and Sarah Guido, “Introduction”, Introduction to Machine Learning with Python, 2017, Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.
[10] Andreas C. Müller and Sarah Guido, “Introduction”, Introduction to Machine Learning with Python, p.49, 2017, Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.
[11] Pandas: https://pandas.pydata.org/ (Accessed June 7, 2020)
[12] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. https://scikit-learn.org/stable/index.html (Accessed June 7, 2020)
[13] DataCamp: https://learn.datacamp.com/ (Accessed June 7, 2020)
[14] Keras: https://keras.io/ (Accessed June 7, 2020)
[15] NumPy: https://numpy.org/ (Accessed June 7, 2020)
[16] Matplotlib: https://matplotlib.org/ (Accessed June 7, 2020)
[17] Seaborn: https://seaborn.pydata.org/ (Accessed June 7, 2020)
[18] Andrew W. Trask, “Grokking Deep Learning”, Manning Publications Co, 2019
[19] Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015 (Accessed June 7, 2020)

Elina Charatsidou, MSc
elinach@kth.se

Elina Charatsidou, MSc
elinach@kth.se