UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Informatics»

ΠΜΣ «Πληροφορική»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**Ανάπτυξη Μεθόδων Μηχανικής Μάθησης (SVMs και NNs) για την Πρόβλεψη Ιδιοτήτων Υλικών με Βάση την Χρήση Δεδομένων Ανάστροφου Χώρου** Development of Machine Learning Models (SVMs and NNs) for Identification of Structural Properties of Materials Based on k-Space Data
Student's name-surname: Ονοματεπώνυμο φοιτητή:	**KONSTANTINOS ZIOVAS** ΚΩΝΣΤΑΝΤΙΝΟΣ ΖΙΩΒΑΣ
Father's name: Πατρώνυμο:	**VASILEIOS** ΒΑΣΙΛΕΙΟΣ
Student's ID No: Αριθμός Μητρώου:	**ΜΠΠΛ/18072**
Supervisor: Επιβλέπων:	**Dionisios Sotiropoulos, Assistant Professor** Διονύσιος Σωτηρόπουλος, Επίκουρος Καθηγητής

Οκτώβριος 2021/ October 2021

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Dionisios Sotiropoulos **Assistant Professor**	**Georgios Tsihrintzis** **Professor**	**Efthimios Alepis** **Associate Professor**
Διονύσιος Σωτηρόπουλος Επίκουρος Καθηγητής	Γεώργιος Τσιχριντζής Καθηγητής	Ευθύμιος Αλέπης Αναπληρωτής Καθηγητής

Preface

This dissertation was carried out at the Department of Informatics, University of Piraeus. It is entirely the result of my own work, and it does not include the results of any work performed in collaboration. No part of this dissertation has been submitted for any other qualification or degree at any university.

Acknowledgements

I wish to express my gratitude to the people who helped me during this lengthy but deeply educational process of my MSc studies. Special mention must be given to my supervisor Dr. Dionisios Sotiropoulos for supporting me during my studies and particularly while working on this thesis.

Furthermore, I feel obliged to deeply thank my friends and family for standing by me every day of my life during this learning journey. They encouraged me to keep pursuing my goals and they went above and beyond to help me achieve them. For these reasons I would like to specially thank my girlfriend Maria and my parents Vasillios and Maria for their unwavering support. Completing this thesis is a result of you efforts as well.

Περίληψη

Η παρούσα εργασία διερευνά την πιθανή εφαρμογή δύο νέων μοντέλων μηχανικής μάθησης, ένα μοντέλο Νευρωνικών Δικτύων (NN) και ένα μοντέλο Support Vector Machine (SVM), για τη γρήγορη πρόβλεψη δομικών χαρακτηριστικών διαφόρων υλικών, τα οποία στηρίζονται σε δεδομένα ανάστροφου χώρου (k-space). Για τη λήψη του σήματος ανάστροφου χώρου από τα συγκεκριμένα υλικά θα μπορούσαν να χρησιμοποιηθούν ποικίλες τεχνικές, ωστόσο, οι πιο πολλά υποσχόμενες μέθοδοι είναι: i) Μαγνητική Τομογραφία (NMRI) και ii) Μικροτομογραφία Ακτίνων Χ (X-Ray mCT).

Για την εκπαίδευση των νέων μοντέλων, χρησιμοποιήθηκαν προσομοιωμένα δεδομένα ανάστροφου. Επιπλέον, μέσω της χρήσης εργαλείων της γλώσσας προγραμματισμού Python, αναπτύχθηκε μια αλυσίδα διεργασιών μηχανικής μάθησης που εμπεριέχει τα βήματα της προ-επεξεργασίας, εκπαίδευσης, επαλήθευσης, δοκιμής και ανάλυσης. Τα καινούργια αυτά μοντέλα, φαίνεται να προσφέρουν μια βελτιομένη απόδοση συγκριτικά με τις ήδη υπάρχουσες μεθόδους. Τέλος, με βάση τη γνώση από την παρούσα διατριβή, παρατίθενται προτάσεις για περαιτέρω βελτίωση και πιθανές νέες εφαρμογές των νέων τεχνικών που αναπτύχθηκαν.

Abstract

This thesis investigates the potential application of two new approaches, a Neural Network (NN) model and a Support Vector Machine (SVM) model, for fast estimation of structural properties of materials based on k-space data. The k-space signal for these materials could be acquired methods such as: i) Nuclear Magnetic Resonance Imaging (NMRI) and ii) X-Ray microtomography (X-Ray mCT).

In order to train and test our models we used simulated k-space data produced with a numerical method previously developed for a Bayesian prediction technique. Furthermore, using advanced tools available with the Python programming language, we developed a machine learning (ML) pre-processing, training, validation, testing and analysis pipeline. The new models investigated here seem to offer an improved performance compared to existing methods. Finally, suggestions for further work are presented based on the knowledge acquired from this thesis project.

Contents

1. Introduction

The analytical methods which are developed in this thesis aim at characterizing the structural properties of various materials based on their k-space signal. Specifically, the material property that we investigate here is the sphere size distribution of a dispersed phase within a continuous material phase. Examples of this include: i) the bubble size distribution in foams, ii) the droplet size distribution in emulsions and iii) the particle size distribution of solid-in-liquid dispersions. Knowledge of these structural properties is very important in the fields of process engineering and material science. They could be directly corelated to many physico-chemical properties and process parameters which need to be studied, monitored and controlled. For example, they could be linked with stability, safety, quality and performance of materials in fields such as food, pharmaceutical and petro-chemicals.

In order to test and prove the performance of these two new machine learning methods we used a series of simulated k-space data. Ideally real NMRI and X-ray data could have also been used in order to acquire the 1D k-space data. However, we opted for the simulated data for three main reasons:

1. Simulated data are easier, cheaper and faster to produce than real experimental data by orders of magnitude.
2. In order to acquire real experimental data we need access to equipment which is only available in very specialized laboratories. This would require a broader collaboration between different universities and this was not in the scope of this MSc thesis.
3. Numerically simulated data, assuming that they are sufficiently accurate representations of the real system, allowed us to precisely control, test and analyze our newly developed methods. They are therefore preferred during this initial proof-of-concept stage as they eliminate all effects and variability that could arise from experimental parameters.

A numerical method was developed, based on previous work of the author, which allowed us to produce a wide range of k-space data from various simulated samples. The parameter that we controlled and that we are mainly interested in is the sphere size

distribution. For each sample, the sphere size distribution can be described by the mean diameter D and the standard deviation σ.

To analyze these simulated k-space data we used Python along with various scientific and numerical computational libraries such as Scikit-learn, NumPy, pandas and seaborn. We developed two machine learning (ML) regression models i) a multi-layer perceptron neural network model and ii) a support vector machine model. For each model, a grid search was performed to identify the optimal hyper-parameters. Then the performance of each model was evaluated based on:

1. The mean absolute error and mean square error
2. The time required for training each model
3. The time required for a pre-trained model to produce a prediction

Finally, the effect of two k-space experimental parameters on the models' performance were tested and they are:

1. The signal-to-noise ratio in the k-space data (SNR)
2. The number of (simulated) set of experimental k-space data we used for training

2. Theoretical Background

2.1. Importance of Sphere Size Distribution in Material Science

Dispersions are multiphase systems, where at least one phase is dispersed in a continuous phase. These systems are commonly found in many different industries ranging from the food, pharmaceutical and petrochemical industries.

Two types of dispersions with particular interest and relevance to the most industries are foams and emulsions. In foams a gas phase is formed from spherical or near spherical (Sadoc & N.Rivier, 1999) bubbles within a continuous liquid phase whereas in emulsions both the dispersed and the continuous phases are liquids. The bubble or droplet size distribution, herein referred to as a generic sphere size distribution (SSD), is an important structural characteristics for foams and emulsions. The exact form of the SSD of dispersed systems may result from the production and treatment processes. In addition, the contribution of the dispersed phase to the viscoelastic properties of emulsions and foams can also depend on the SSD (Weaire & Hutzler, 1999; Wilson, 1989). Furthermore, the SSD can be related to the physicochemical properties and the stability of these systems (Campbell & Mougeot, 1999)(Sadoc & Rivier, 1997). Some examples from different industries include the use of foams in wet processing in the textile industry (Capponi et al., 1982), where the foam SSD effects the efficiency of the process, or the pharmaceutical industry where emulsions are used as drug delivery systems (Zuniga & Aguilera, 2008). In recent years foams find an increased application in the food industry with the emergence of functional foods (Green et al., 2013; Skurtys & Aguilera, 2007). Aerated food systems and emulsions can help to enhance sensory response whilst reducing the amount of flavouring used (Zuniga & Aguilera, 2008), reduce the calorific density of foods (Rodríguez-García et al., 2013) and deliver nutrients or bioactive molecules much more efficiently (Zuniga & Aguilera, 2008). Regardless of the specific application, it is highly desirable for scientists to be able to non-invasively characterise these systems andmonitor their evolution over time.

2.2.Basic Principles of Foams and Emulsions

The two systems share many common structural properties and a number of physical phenomena. Therefore most of the theories that have been developed for foams can be expanded to include emulsions. Indeed, in many experimental studies one system has been used as a model case for the other (Weaire & Hutzler, 1999). A similar approach was taken in this thesis where foam systems are studied but the results can also be applicable to emulsions. Therefore, in this section the theory will focus on foams but certain characteristics of emulsions will also be mentioned. The general foam and emulsion theory presented below can be found in various textbooks (Weaire & Hutzler, 1999)(Wilson, 1989) therefore only recent developments will be referenced when required.

2.2.1. Foams

A liquid foam is defined as a dispersion of gas in a liquid phase. Usually in order to assist the formation of bubbles and stabilize the foam, surfactants are added, to assist the foam formation by reducing the surface tension.

For most foams the volume fraction of gas, , ranges between 0.5 and 0.98. Based on the fraction of gas in the structure, the foams are divided into wet and dry foams. For gas fractions ϕ>0.9 , the bubbles in the foam deform each other leading to the familiar polyhedral structure that many food foams have. The liquid forms thin films around the air bubbles creating polyhedral cells. The surfaces of these cells are curved and the liquid films meet each other in lines which in turn meet in vertices. The structure described above conforms to dry foams (Weaire & Hutzler, 1999).

If a foam has a low gas fraction (ϕ<0.95), the idealized description mentioned previously is not accurate and the geometry is a little different. The idealized lines are replaced by channels of finite width which are called Plateau borders. Plateau borders contain the majority of the liquid. The cells of the foam lose their sharp edges and the corners are rounded off. Figure 1 shows a 2-D structure of dry and wet foam. The cells of the dry foam are replaced by bubbles in the wet foam.

Systems with very low gas fraction (ϕ<0.5) are better described as suspensions of bubbles or bubbly fluids rather than foams. In the work presented in this thesis, more emphasis is given on systems that are either well into the wet limit $\phi \leq 0.65$.

For ideal dry foams with very high gas fractions, Plateau in his work proposed the rules which are necessary for these systems to be at equilibrium. For those foams, in a 3-D structure, the liquid films can intersect only three at a time at angles of 120^{o}. Also no more than four lines can meet in a vertex, and these vertices have near perfect tetrahedral symmetry. Finally, at the points where a Plateau border comes in touch with a surface the joint is smooth and the surface normal is the same on both sides. If the gas fraction becomes too low ($\phi \leq 0.65$) the microstructure of the wet foam is no longer polyhedral and appears more like a liquid with individual bubbles dispersed in it (Weaire & Hutzler, 1999; Wilson, 1989).

When the structure loses its mechanical stability, Plateau's rules no longer apply and multiple junctions may be found. The point of that transition is called the wet limit (Sadoc & Rivier, 1997). At this point the bubbles tend to have an almost spherical shape. The transitions limits between dry and wet foams are not well defined.

2.2.2. Emulsions

Emulsions are also dispersions but unlike foams, both the dispersed and the continuous phase are liquids. Therefore the dispersed phase forms droplets in the continuous phase, instead of bubbles. In foam systems surfactants are usually added in order to decrease the surface tension. In the case of emulsions amphiphilic compounds, called emulsifiers, are used to produce stable systems (Schramm, 2014).

Another difference between foams and emulsions, is the size range of the droplets. Usually, the droplets sizes found in commercially useful emulsions are much smaller than the diameter of the bubbles in foams. Bubbles in commercial foam systems are in the range of 10 μm up to several mm. By contrast, the droplet radii of emulsions can range from 10 nm up to 1 mm. Specifically, emulsions in the range of 10 nm to 100 nm are called nanoemulsions. Systems with droplet radii in the range of 100 nm to 1000 nm are called

miniemulsions whereas systems in the range of 1000 nm to 1 mm are called macroemulsions (Hsiung et al., 2006; Kobayashi et al., 2007; Skurtys & Aguilera, 2007).

In most cases, emulsions are composed of two phases and both water-in-oil (W/O) and oil-in-water (O/W) emulsions can be produced. More complex systems can also be produced such as water-in-oil-in-water ($W_1/O/W_2$), oil-in-water-in-oil ($O_1/W/O_2$) or even multiple oil-in-water-in-water ($O/W_1/W_2$). Figure illustrates the basic structure of these systems(Chung and McClements, 2014).

2.3.Traditional Methods for Sphere Size Distribution Analysis

Currently, a wide range of methods are used for the estimation of the SSD. The most popular non-NMR techniques that are used to study dispersed systems are microscopy, light scattering methods, X-ray tomography, ultrasound spectroscopy and electrical conductivity measurements.

In addition, in recent years a number of NMR techniques have also been used to characterize the size distribution in food foams and emulsions and there is currently an expanding activity in this area (Johns & Gladden, 2002)(F. Mariette, 2009). The majority of the approaches have adopted pulsed field gradient (PFG) NMR but T_2 relaxometry and magnetic resonance imaging (MRI) have also been investigated.

Optical microscopy is the most widely used technique. Optimal microscopy can be used for a wide range of systems and applications due to its capability to produce images with very good resolution, relative simplicity and extensive development. Conversely, a disadvantage of optical microscopic is that the observations are limited to a few layers of foams and emulsions and thus they do not provide the global size distribution of the sample but only a local one. Furthermore, they can be destructive for the sample since dilution is, in many cases, required (Blonk & Aalst, 1993).

Light scattering is also commonly used and it can cover a wide range of sizes from sub-micron level up to centimetres (Durian et al., 1991). Additionally, multiple light scattering techniques can provide information about the dynamics and the evolution of foams and

emulsions (Höhler et al., 2014). A drawback of this approach is that only dilute systems can be studied and the dilution of more concentrated foams and emulsions can be destructive for the samples (Coupland & Mcclements, 2001).

Ultrasound spectrometry has been investigated as a potential low cost method for samples under a range of process conditions. However, this method can be seriously affected by particle impurities and the thermo-physical properties of the continuous and dispersed phase (Johns & Hollingsworth, 2007).

Electrical conductivity measurements often require the addition of an electrolyte in the aqueous phase which cannot be easily applied to non-diluted samples (Kostoglou et al., 2010). A similar technique called electrical capacitance tomography (ECT) can be used for image acquisition. The images can then be used in order to extract the SSD. This approach has two major drawbacks: (i) computational demanding and very time consuming algorithms have to be used, (ii) the final image resolution is also usually low (Hou et al., 2019).

3D X-ray tomography is a very powerful technique with the capability of providing high resolution images of the system's microstructure (Trater & Alavi, 2005)(A.J.Meagher et al., n.d.). However, its main disadvantage is that it is very time consuming, because of the image acquisition and reconstruction process. In many cases, it can also be described as an invasive technique (as the sample preparation techniques can be destructive) (Johns & Hollingsworth, 2007).

2.4.NNs and SVMs for K-Space Signal Analysis

Based on the review of the existing analytical techniques presented in section 2.3, we can see that there is a need for a fast and non-invasive method that can be used to study the sphere size distribution both in stable as well as in dynamic systems.

Our approach to develop such a method is to use the existing NMRI or X-Ray mCT techniques but in combination with a novel machine learning data analysis method. More specifically, as we discussed in section 2.3. a major drawback of both these methods

(NMRI and X-Ray mCT) is that they are commonly used to acquire a 3D image of a sample which is then analyzed further to extract the desired information about the sphere size distribution of the material. Both the 3D image acquisition and 3D image analysis are very time-consuming steps. It was however shown in previous work (Holland et al., 2012) that the information related to the sphere size distribution of a system exists in the 1-dimensional k-space signal of that system. Moreover, it was demonstrated that is possible to use that 1D k-space signal acquired from NMRI or X-Ray mCT, in combination with a Bayesian statistical method, to predict the sphere size distribution of various materials (K. Ziovas et al., 2016).

Despite the promising applications that this approach could have the Bayesian signal processing method, the authors reported that two major drawback of this method are (K. Ziovas et al., 2016):

i. It's relatively low accuracy-low confidence for the predicted results
ii. The need to build a numerical model of the desired system which is then used to build the Bayesian statistical model of the k-space signal.

The second point proved in particular proved to be extremely important in cases where the structure of the system under investigation is not well understood. In such cases the Bayesian sphere sizing method produces very low accuracy results which, as the author of the method explain, cannot be used to gain any useful information about the sphere size distribution (Ross et al., 2012a; K. Ziovas et al., 2016).

What we aim to do in this thesis is to build on this recent work and investigate other machine learning methods, namely SVMs and Neural Networks, in order to:

i. Improve the performance.
ii. Expand the range of potential applications for the 1D k-pace signal analysis method.

In this section we will discuss what the k-space signal is. Furthermore, we will present some examples of what the signal from a material with a spherical dispersed phase might look like. We will not discuss the details of how this signal can be acquired form NMR and X-Ray however since these are very complex techniques, and the focus of this thesis lies more on i) the signal properties itself and ii) the machine learning algorithms that we

implemented in order to analyze the signal. More details about the theory of NMRI and X-Ray mCT can be found in textbooks (Callaghan, 1993)(Abraham et al., 1988)(Baruchel et al., 2000; Ketcham, 2001) and other published work (Carlson et al., 2003; Cierniak, 2011; Laverse et al., 2012; Meagher et al., 2011)(F., 2009; Francois Mariette et al., 2012).

2.4.1. K-space signal and sphere size distribution

K-space is a name for the frequency domain of a spatial Fourier transform (reciprocal space) used predominantly by scientists in the MRI field (Callaghan, 1993, 2011)(Levitt, 2007).

More specifically the 1-dimensional k-space signal that we are investigating in this thesis can be described, generally, as the 1D Fourier transform of a real 1D image (or profile). To understand the properties of k-space therefore, one must gain a good understanding of what the Fourier transformation (FT) is and how it is used to transform time or space domain signal into temporal or spatial frequency domain.

2.4.1.1.Sound signal frequency analysis

An intuitive example of time domain to frequency domain transformation, is when the FT is used to express a musical chord (from the time domain) in terms of the volumes and frequencies associated with the individual notes of the cord.
The continuous FT is described by the equation (Bracewell, 1999):

$$F\{x(t)\} = X(\xi) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi\xi t} dt \qquad (1)$$

where $x(t)$ is a function of time t and ξ is the frequency. This equation allows us to map a signal from the time domain to the frequency domain. The inverse FT, as the name suggests, performs the inverse operation and it is described by the equation (Bracewell, 1999):

$$x(t) = \int_{-\infty}^{\infty} X(\xi)\, e^{j2\pi\xi t} dt \qquad (2)$$

Figure 1 shows an example of two simple sine waves with frequencies f = 2Hz and f = 6Hz in the time domain as well as their respective frequency domain plots. Since these are simple single frequency waveforms in the time domain, they only have one frequency component in the reciprocal frequency domain. More complex waveforms would have intensity peaks at various frequencies in the frequency domain.

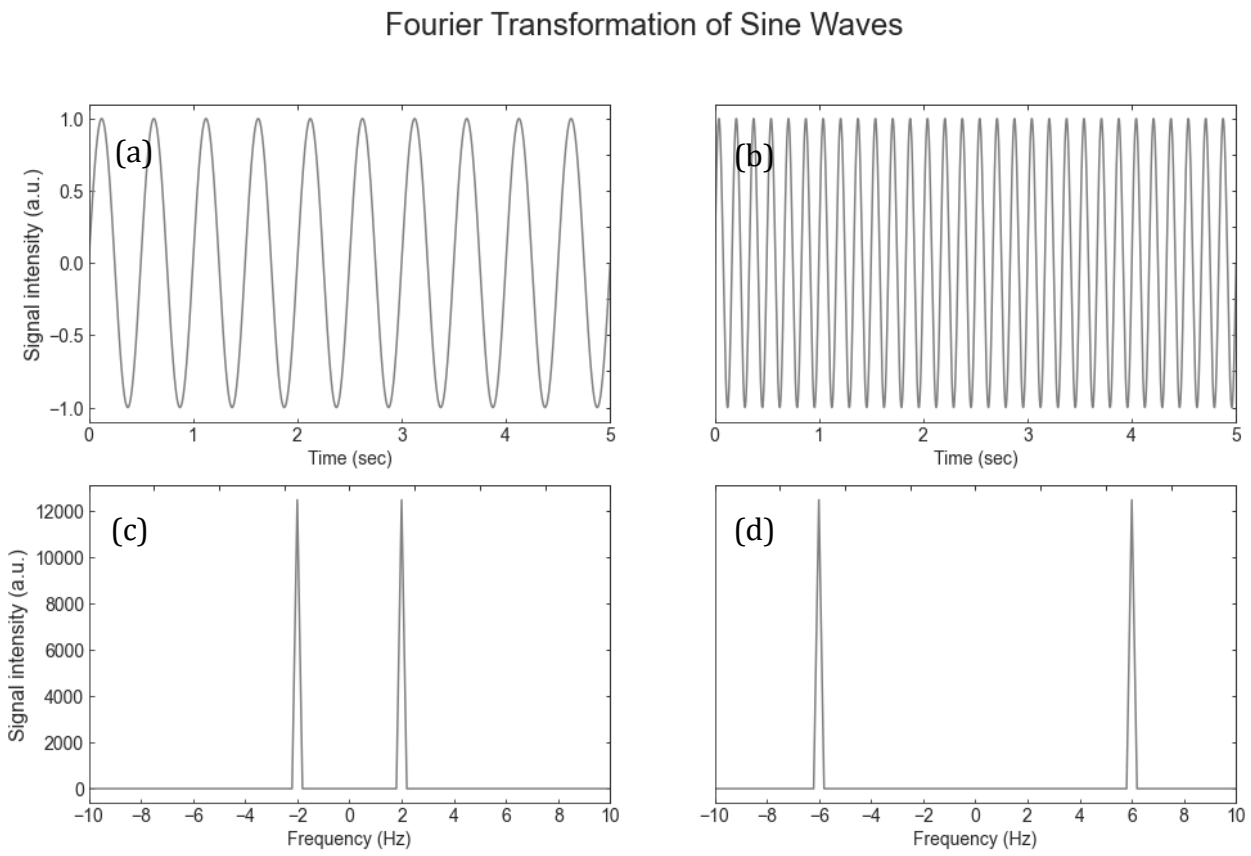

Figure 1. The time and frequency domain representations of sine waves with two different frequencies. a) The time domain signal for a sine wave with f = 2Hz, b) the time domain signal for a sine wave with f = 6Hz and c) and d) their frequency domain representations respectively.

Figure 2 shows the time and frequency domain of the combined waveforms of the two waves previously shown in figure 1. It easy to see that if more and more waveforms are added the frequency domain "signature" of the original wave would include peaks at multiple frequency points. The intensity of the peaks at these points is proportional to the relative contribution of each wave component to the total signal. In this example the wave with frequency f = 2Hz has an intensity twice that of the wave with f = 6Hz in the final combined wave. If now we invert the problem and we start with the frequency domain (k-space) data we could use this information to predict some properties for the original signal. For example, we might wish to calculate the average wavelength of the

waves in the original signal. This can be done analytically since we can calculate the wavelength of a wave given its frequency f and its speed C in the medium where the wave propagated with the equation:

$$\lambda = \frac{C}{f} \tag{3}$$

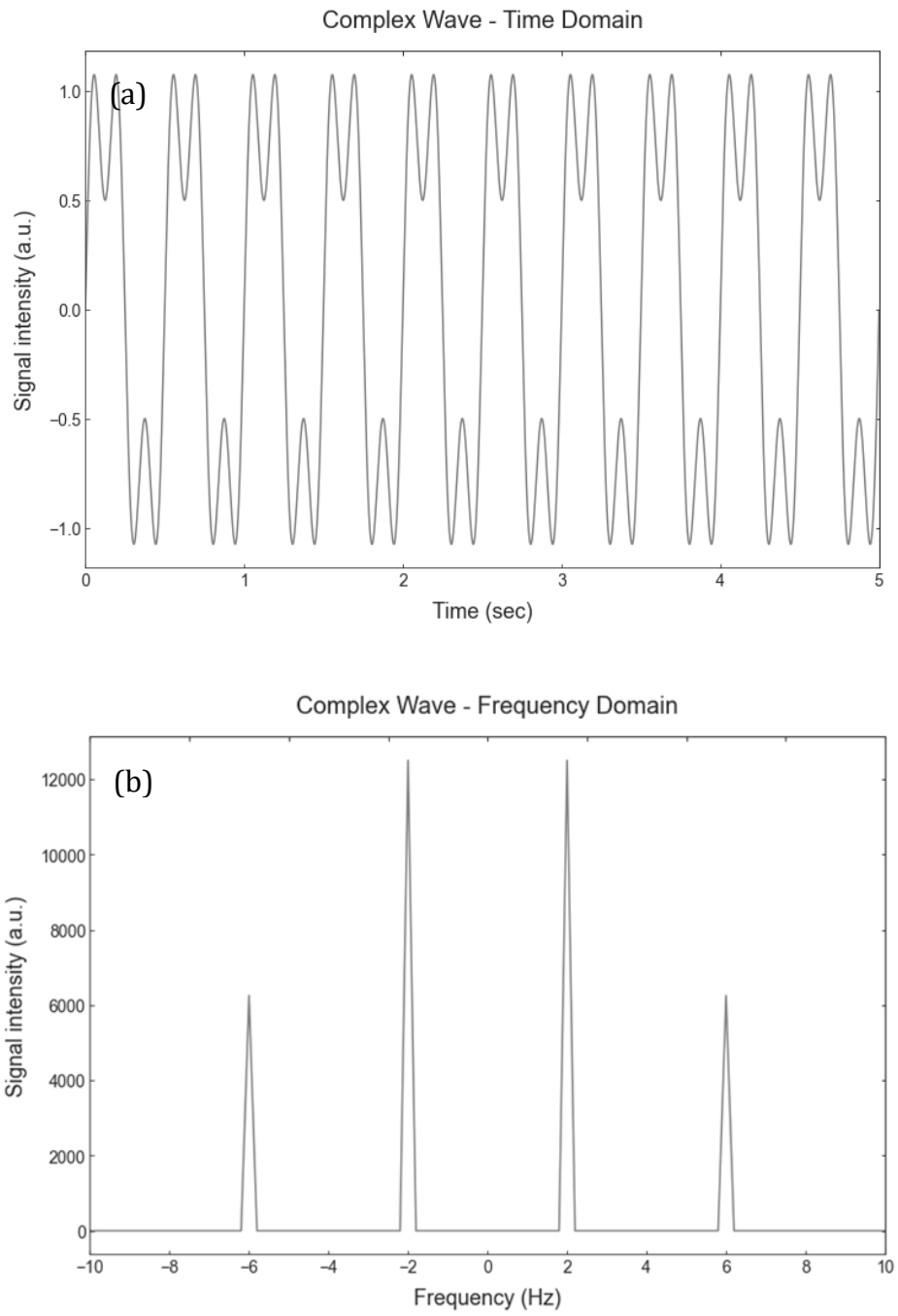

Figure 2. The (a) time and (b) frequency domain representations of two combined sine waves with initial frequencies f_1=2Hz and f_2=6Hz.

Usually, the propagation speed is either known or it can be measured. But let's assume that for a specific case the speed of the wave in the medium is unknown and furthermore the speed is not always constant, so it cannot be exactly determined by performing experiments. Then we cannot directly calculate the wavelength. It might be possible however to produce an estimation for the wavelength. One could use a machine learning algorithm, like those that we will discuss in later chapters, to train a model which can identify a correlation between the wavelength and the frequency. The model could be trained for various conditions where the propagation speed would take different values. Despite the changing propagation speed however the model would hopefully be able to create a correlation between the original wavelength and the frequency. It might for example be able to identify a pattern where the relative distance between the frequency peaks is always constant for a particular average wavelength compared to other waves, even though the absolute value of the frequency peaks might vary.

2.4.1.2. k-space signal patterns

Although the wavelength example might look as a not very interesting problem there are many cases where this approach could provide us with a reasonably accurate estimation for very challenging tasks. One such case we explore in this thesis. The goal is to estimate the average sphere size of a dispersion based on a k-space data set. The properties of the materials that we aim to study are often unknown and it is hard to determine and model these materials. Just like the wavelength example mentioned above, one could try to identify some patterns in the k-space signal with the help of ML models and estimate the sphere size rather than directly attempting to calculate it. One such approach we will explore in this thesis and the implementation details will be described in the "*Method Development and Implementation*" and "*Results and Analysis*" sections.

To gain an insight into what type of patterns the ML models will try to identify we can look at the k-space signal coming from two different samples. The only difference between the samples is that they have a dispersed phase with a different mean sphere size. What we are interested is to look at the k-space signal and identify a potential pattern that is corelated with the sphere size distribution. If such a pattern exists, then the ML models that we will train could learn to identify it. Figure 3 show the simulated

k-space signal from two samples where sample (*a*) has a mean sphere diameter of *D* = 3 au. and sample (*b*) has a mean sphere diameter of *D* = 6 au. The details of how we can simulate this signal will be presented in the "*Method Development and Implementation*" section. Moreover, this signal can be acquired from real samples using MRI and Xray techniques (Callaghan, 1991)(Baruchel et al., 2000)(K. Ziovas et al., 2016) as we already discussed, but the details of these methods are not in the scope of this thesis.

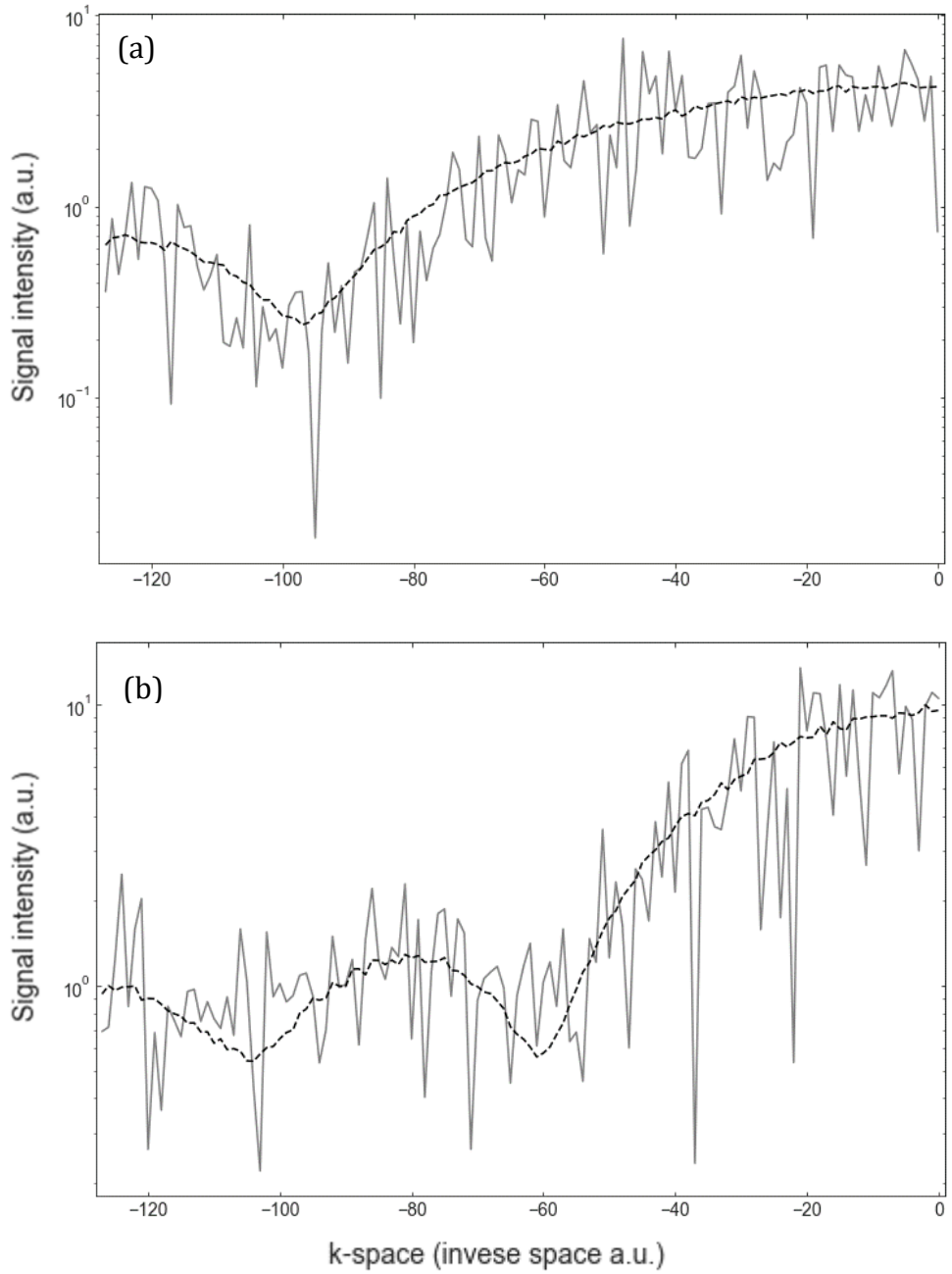

Figure 3. The k-space signal from two samples where (a) is a simulated sample with mean diameter D=3au and (b) is a simulated sample with mean diameter D=6au. The continuous fluctuating lines represent the data from a single sample while the dashed lines represent the underlining pattern found in the signal.

The signal is represented with the solid line, and it can be seen in both cases that it is noise and varies unpredictably from point to point. However, it is also possible to identify a pattern in this signal which we illustrate by the dashed lines. These patterns are similar to diffraction patterns (Håkansson et al., 1998) but they are buried in the background noise and may even be impossible to identify visually. A ML model should be able to learn to discover these patterns in the k-space signal, if they exist, and correlate them with a specific sphere size distribution (Le et al., 2020). Based on this intuition we aim to test the performance of two ML models ANNs and SVMs on identifying these patterns.

2.4.2. Overview of Artificial Neural Networks (ANN)

Artificial neural networks are currently one of the most popular machine learning methodologies. They have attracted a lot of interest the past few decades mainly because they seem to outperform most traditional machine learning methods in a variety of tasks from classification to regression. In this section we attempt to provide an overview of the theory and mathematical foundations of neural networks. We also present some of the most important aspects of the training process of a neural network.

2.4.2.1. Biological foundation of Neural Networks

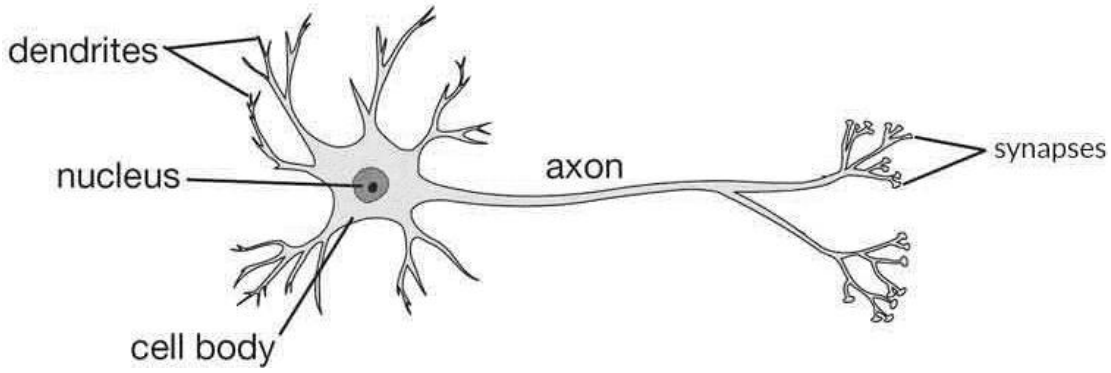

Figure 4. Depiction of a biological neuron and it most important components.

The brain is the component of our body that enables humans to learn and perform a variety of tasks. It is estimated to have about 10 billion neurons. Neurons are interconnected with each other, forming a large network where each neuron receives

input from other neurons through its synapses. The input of a neuron is summed and when this sum exceeds a particular threshold, the neuron sends an electrical spike to other neuron(s) through its axon (Bishop, 2006).

Perceptron is an algorithm in machine learning for supervised learning of binary classifiers, i.e., a function to determine the class in which the input vector belongs (Agarwal, 2019). The algorithm draws inspiration form the biological neurons. It receives a signal in its input, it performs some mathematical operations on it and then sums the results which is then passed as input to another neuron just like the biological neurons that we presented above. The basic structure and mathematical foundation of a perceptron is presented in figure 5.

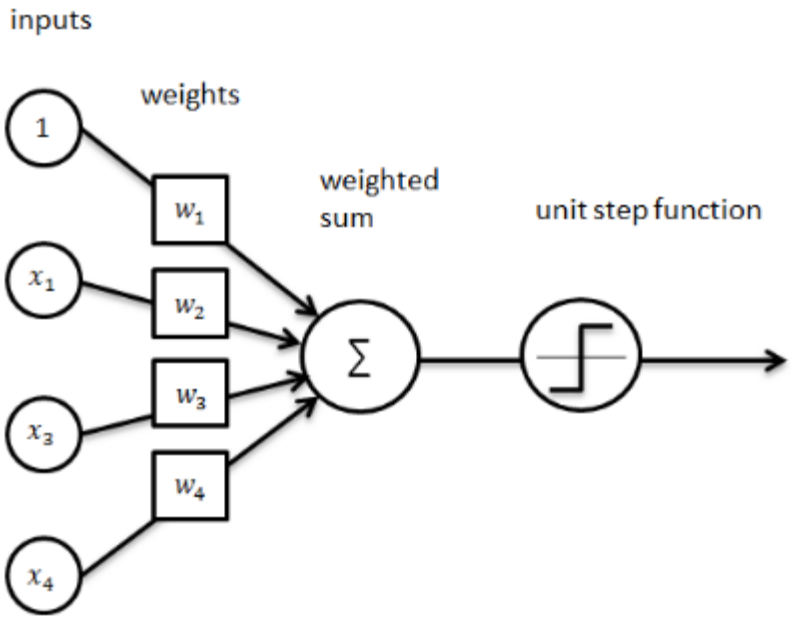

Figure 5. The structure of a simple perceptron model and its various components.

This structure can be mathematically written as:

$$f(x) = \begin{cases} 1 \; if \; w \cdot \vec{x} + b > 0 \\ 0 \; otherwise \end{cases} \quad (4)$$

where w is a vector of real-valued weights, b is bias and $\vec{x}$ is a vector of input values x (Géron, 2019).

2.4.2.2. From Perceptrons to Neural Networks

Using the perceptron model as a building block a basic artificial neural network can be constructed. The goal of a neural network is to produce a prediction for the value of a function f for a given set of inputs. A simple neural is in essence a multi-layer perceptron and it contains the following components: i) Neurons, ii) Bias units, iii) Synaptic weights, iv) Activation functions and v) Hidden layers (Ian et al., 2016).

Neurons

The neuron is the perceptron model along with its input and output as we presented in the previous section.

Bias units

Bias is an adjustable parameter of the neural network. It allows us to adjust the result of the activation function by adding a constant (i.e. the bias) to the neurons input. It can be thought of as analogous to the role of a constant(intercept) in a linear function (Chollet, 2018).

Synaptic weights

By continuing the analogy that we used to describe the bias, the weights can be thought of as the co-efficient of the equation that the NN is trying to solve. Just like the coefficient of a linear function, negative weights reduce the value of an output. Weights are probably the most important parameter as they form the core of the NN. During the training stage of a NN, the weights are initialized based on a given strategy (usually at random). Through the training process the values of the weights are optimized and the set of weights that produce the best prediction are chosen as the values for our model (Brauer, 2018).

Activation functions

An activation function dictates how the output of a node is transformed into the input for the next layer of the network. There is a wide range of activation functions to choose from and we will discuss some of them later. In general, they can be linear or nonlinear. If a nonlinear function is chosen it introduce a nonlinearity in the neural network design. The choice of activation function directly affects the performance of the neural network. It is not uncommon to use different activation functions in different parts of the model

depending on the task that each layer performs in the NN architecture. For example, the activation function used in the hidden layers are usually different that the function used in the output of the NN (Agarwal, 2019).

Hidden layers

A hidden layer is located between the input and output of the NN. It's job it consists of neurons and its job is to applies weights to its input and direct them through an activation function to the output. In practice, most neural networks have more than one hidden layer to be able to adjust and predict on the non-linearities that usually exist on the data we are trying to analyze. Having a single hidden layer would be equivalent to predict a simple linear function which, in most cases is not a good approximation for the results that we are trying to predict (Géron, 2019).

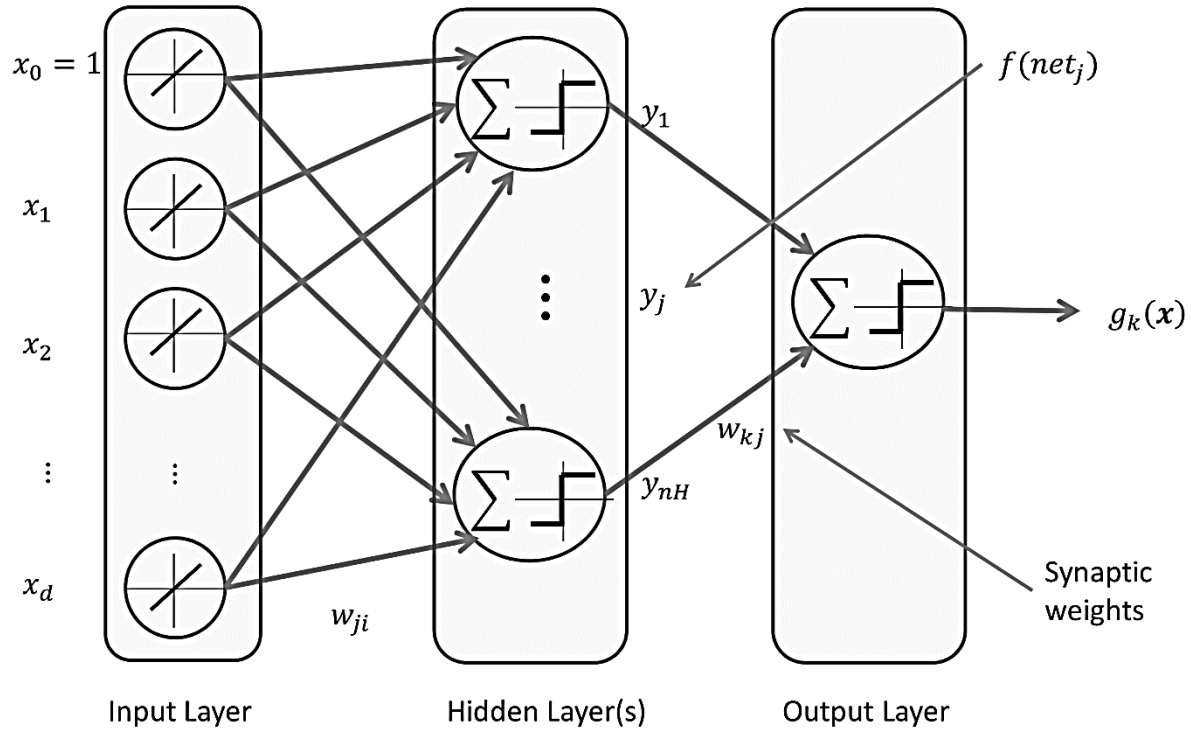

Figure 6. The structure of a NN with an input, and output and one hidden layer. The connections between the nodes and the different layers of the NN can be seen as well as the weight for each synaptic connection.

2.4.2.3. How does a NN produce a prediction

For the NN architecture presented in figure 6 we can say that the result of the propagation of the input signal x through a single hidden layer, before the result pass though the activation function is given by the equation (Deng & Yu, 2013):

$$net_j = \sum_{i=1}^{d} x_i w_{ij} + w_{j0} \quad (5)$$

And after that signal passed through the activation function *f(x):*

$$y_j = f(net_j) \quad (6)$$

In the output layer the signal is summed and passed through on more activation function *g(x)* in order to produce the final result of the NN:

$$z_k = g(\sum_{j=1}^{nP} w_{kj} y_j + w_{k0}) \quad (7)$$

where nP is the number of nodes in the hidden layer & w_{k0} are the bias values.

2.4.2.4.Training an ANN: Challenges and Considerations

We presented the mathematical formulas based on which a NN can produce a prediction based on an input vector $\vec{x}$. But for this calculation to work a set of parameters need to be calculated first such as the weight and the biases for each node and each layer of the NN. The estimation process of these parameters is what is commonly referred to as the "training" of a NN.

In order to evaluate how good, the predictions of the NN are, we use a loss function. This function is set as the objective function in an optimization problem where the control variables are the weights. During the optimization process the partial derivative of the cost function is calculated for each weight and the weights are then updated to achieve the optimum value for the cost function.

The most common strategy used for updating the values of the weights is back-propagation (Géron, 2019). Back-propagation allows us to find the partial derivative for each weight using the following equations. We can describe the loss function as:

$$J(w) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}\|t - z\|^2 \quad (8)$$

where *J(w)* is the loss function, t is the expected output, c is total number of outputs and z is the model prediction. We can then calculate the partial derivatives of the weights of each layer as follows:

$$\Delta w = -\eta\frac{\partial J}{\partial w} = -\eta\frac{\partial J}{\partial net} * \frac{\partial net}{\partial w} \quad (9)$$

So, for the last layer this calculation can be solved as:

$$\Delta w_{kj} = -\eta\frac{\partial J}{\partial w_{kj}} = -\eta\frac{\partial J}{\partial net_k} * \frac{\partial net_k}{\partial w_{kj}} \quad (10)$$

where net_k is:

$$\sum_{j=1}^{nP} y_i w_{kj} + w_{k0} \quad (11)$$

Equation 11 then becomes:

$$\Delta w_{kj} = -\eta\left(\frac{\partial J}{\partial z_k} * \frac{\partial z_k}{\partial net_k}\right) * y_j \quad (12)$$

$$\Delta w_{kj} == \eta(t_k - z_k) * f`(net_k) * y_j \quad (13)$$

where η is the learning rate. The learning rate is an important meta-parameter of the model which dictates how big the change of the weight values is between each iteration. Small values of learning rate will increase computational time. Very large values might

lead to model instability because the model will constantly overshoot the optimal value and it will never converge. A more detailed explanation as well as various strategies regarding the value of the learning rate can be found in the literature (Müller, A., & Guido, 2018).

The change of the weight values for the previous layers can then also be updated in a similar way by applying the same equations. After this backward pass is completed, the weights can be updated by simply applying the following equation:

$$w(m+1) = w(m) + \Delta w \tag{14}$$

Now that we have a mechanism for updating the weights based on the error(distance) between the desired and the predicted value we can choose an optimization strategy for the objective function. A commonly used optimization method is the stochastic gradient descent (Friedman et al., 2009; Géron, 2019). There is extensive literature covering the details of the stochastic gradient descent. For the purpose of this section, we can that the stochastic gradient descent is an iterative optimization method. Its goal is to start from a random point on a function and reach the lowest point (minimum) of that function by travelling down the slope of the function.

The entire training process that we described here can then be summarized as follows:

1. The weights of the NN are initialized with random values.
2. The input of the NN is passed through the layers of the NN, where the initial guesses for the weight values are used to calculate the output of the network.
3. The difference(error) between the expected and the estimated output is calculated. Based on this value we use the backpropagation method to update the weights of the NN
4. The new estimated values are then calculated. If they newly calculated error is higher than our target value, we repeat the calculations are repeated.
5. The algorithm, usually, terminates based on two conditions:
    a. Either because the error has decreased below an acceptable level that we have pre-defined or
    b. Because the maximum number of iterations has been exceeded.

It is important to terminate the process after a maximum number of iteration because the target value for the error may be never reached. This would cause our algorithm to oscillate indefinitely (Ian et al., 2016).

2.4.2.5. Practical aspects of Back-Propagation

Activation function

As mentioned in section 2.4.2.2 the activation function is an important component of the ANNs. The output signal of each node of a layer is passed through an activation function before it is fed to the next layer. The activation function transforms the signal in such a way that a non-linear behavior is introduced to the output signal. Without an activation function a neural network would work just like a linear regression model. The non-linearity introduced by the activation function allows the ANN model to "fit" more complex and non-linear curves on the data (Chollet, 2018; Deng & Yu, 2013; Géron, 2019).

Commonly used activation functions have a set of desirable properties that increase the performance of the ANN (Géron, 2019). An activation function should ideally be:

- ***Continuous and Smooth***. Smoothness is important for improving the performance of gradient based optimization methods. These methods converge easier to the optimal solution when smoother functions are used. Moreover, the activation function should produce a continuous output since its input signal, coming from a node, is also continuous.
- ***Non-linear***. As discussed earlier it is necessary for a non-linear output to be produced so that the ANN can better "learn" non-linear patterns that might exist in the training data.
- ***Differentiable***. This requirement arises from the need to calculate the partial derivatives for the back-propagation step presented in section 2.4.2.4.
- ***Monotonic***. The behavior of the function must be monotonic so that, as the weight of a neuron increases, the output of the activation function will also increase. A non-monotonic function would result in unpredictable behavior.

- ***Saturated***. This property limits the potential output range of a neuron between an upper and a lower limit. It is ensured therefore, that no single weight can dominate the final output signal.
- ***Linear or constrained for small values***. It might seem that this requirement contradicts the non-linearity that we mentioned earlier. However, if the behavior of the activation function is not linear (or constrained in another way) for small weight values the vanishing or exploding gradient problem arises (Deng & Yu, 2013). The vanishing gradients appear with certain activation functions whereas more layers are added to the ANN the gradient of the loss function tend to zero. This in turn makes the training process harder and maybe impossible. On the other hand, exploding gradient appear when error gradients accumulate over time and can result in extremely large gradients. In some cases, the weight value grows so fast that they overflow the memory and "NaN" values appear, which causes the optimization process to fail completely. These problems are discussed in more detail in the literature (Géron, 2019; Maleki et al., 2020; Rozenberg et al., 2012).

Loss functions

The purpose of the loss function in the ANN is to calculate the difference between the target (expected) output and the neural networks predicted value. Los functions and their properties are of great importance for ANNs and there is a lot of published work on this topic (Ian et al., 2016; Messaoud et al., 2020). Three broad types can however be identified:

1. ***Classification functions***. They are used when we wish to find the probability that our input signal belongs to a distinct category. The output in this case consists of distinct classes and a probability is assigned to each. Commonly used classification functions are margin classifier, negative log-likelihood and categorical cross-entropy.
2. ***Regressive functions***. When the desired prediction is a continuous quantity rather than a distinct class, continuous regressive functions are used. This category of functions includes mean square error and absolute error.

3. ***Embedding functions***. In some ML problems we are interested in determining how similar two (or more) inputs are. To determine this degree of similarity functions such as the L1 hinge error and the cosine error are used.

Optimization Algorithms

During the training process optimization algorithms are used to find the best combination of weight values that minimizes the loss function (Brauer, 2018; Deng & Yu, 2013; Ian et al., 2016). These algorithms can be divided in two categories, based on how the learning rate is chosen:

1. Algorithms with constant learning rate where the learning rate is the same (and fixed in value) for all weights. The stochastic gradient descent that we discussed earlier is such an algorithm.
2. Algorithms with adaptive learning rates. They provide various methods for adjusting the learning rate during the training process to offer a better balance between speed of convergence and accuracy (Friedman et al., 2009). "Adam" is a very popular adaptive algorithm.

2.4.3. Overview of Support Vector Machines (SVMs)

Another popular machine learning algorithm that has been relatively that can be used to tackle both regression and classification problems, just like the ANN, is the support vector machine (SVM) algorithm.

The theory behind SVMs has a long history and it was first developed during the 1960's however it was not until the early 1990's that practical, working classifiers that used the kernel trick were invented (James et al., 2013). SVMs in their original form could have only been used to analyze linearly separable data. The introduction of the kernel trick however (James et al., 2013), which will be discussed in this chapter, was a major advancement for SVMs since it made the analysis of non-linearly separable data, not only possible, but also computationally efficient.

2.4.3.1. Linearly separable data

A one-dimensional set of data, with two classes can be considered linearly separable if a point exists such that:

- It could separate the data into two distinct groups and
- Each group consists only of data points from a single class

Such a case is illustrated in Figure 7.

Figure 7. A 1-dimensional linearly separable set of data. The data can be easily separated in two classes by choosing an appropriate point on the x-axis.

As seen in figure 7 at least one point exists that can act as a boundary between the class on the left of the boundary and the class on the right. In this dimension therefore, the data set is linearly separable. If we now, try to separate a two-dimensional data set the boundary between the two classes is not anymore, a point but a straight line. This is illustrated in figure 8 where, again the two classes are being perfectly separated by a linear boundary. If we then move to a three-dimensional (3D) space the boundary now becomes a 2D plane.

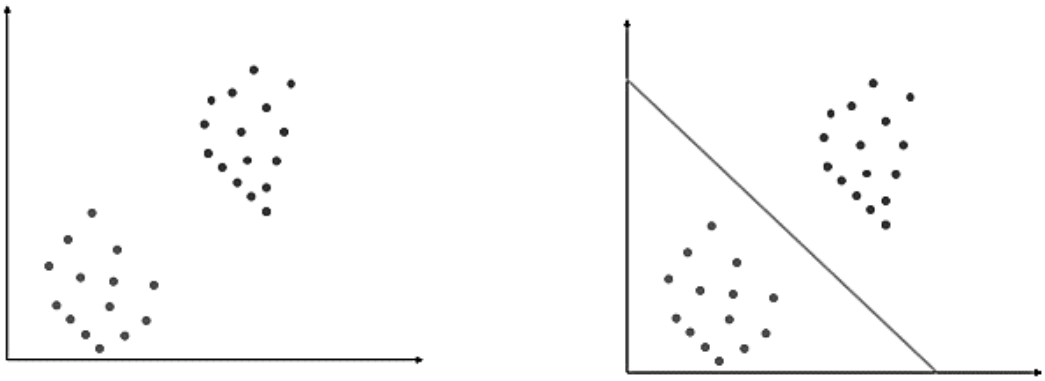

Figure 8. A 2-dimensional linearly separable set of data. The data can be easily separated in two classes by choosing an appropriate line such as the one illustrated on the right diagram.

One can imagine how this analysis can be further expanded to higher dimensional spaces. The decision boundaries for n-dimensional data sets are called hyperplanes. A hyperplane is therefore a n-1 dimensional subspace of the original n-dimensional data space.

2.4.3.2. Main concepts and mathematical foundation

Given the linearly separable data sets that we show in the previous section it is often desired to identify the best way that two (or more) data classes in that data set can be separated. As we discussed, there are potentially infinite numbers of decision boundaries between two linearly separable classes. The question that arises is how one can choose the decision boundary that provides the best separation between these classes. This is precisely where the SVM algorithm has proven to be useful. It allows us to identify the hyperplane that separates the two classes while providing the maximum margin on either side of the decision boundary. This is illustrated in figure 9.

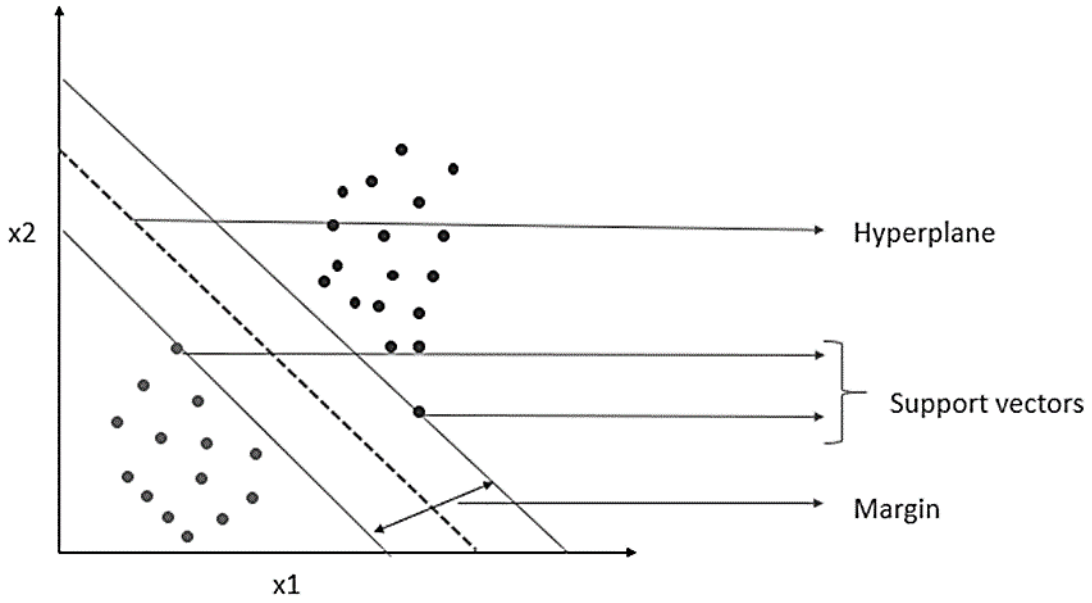

Figure 9. A hyperplane separating two classes in a 2D space. The margin as well as the support vectors of the hyperplane can be seen.

Before further investigating how an SVM can achieve this goal it is important to explain the basic terminology used in describing SVMs. The most important terms are:

- ***Support vectors***. As seen in figure 9, on either side of the decision boundary the are some points that are closer to the boundary that the rest of the points in each class. These are known as support vectors (Pisner & Schnyer, 2019). They are in essence the most important point of each class since they define where the hyperplane should be placed. If a support vector is removed from a class, then the decision boundary will change. On the other hand, removing any other point from either class will not affect the decision boundary.

- ***Margin***. This is defined as the distance between the support vectors and the decision boundary as can be seen again in figure 9. There can be two types of margins the soft and the hard margin. When an SVM model is created we could allow some of the data points from each class to fall within the margin on either side of the hyperplane. This is an instance of the bias-variance tradeoff (Müller, A., & Guido, 2018) where the model is allowed some degree of error on the training data in exchange for flexibility. If we restrict our model in such a way that no points are allowed to fall within the margin, then we are using a hard margin.

Given a data set where two linearly separable class A and B exist. We wish to identify the line that separate these two classes so that we can then use this decision boundary to predict the class of a new unknown data point *x* and place it in either class A or class B. For a two-dimensional data space, the decision boundary, as we saw before is a linear equation:

$$g(X) = w^T X + b = 0 \quad (15)$$

Where *w* is the weight vector, *X* is the input vector and *b* is the bias. It must be noted that the weight vector can be thought to represent the orientation of the hyperplane in the n-dimensional feature space.

Since the hyperplane is the decision boundary between the two classes in the entire feature space this means that for each feature vector there is a linear function $g(X_i)$. If the feature value is on the positive side of the hyperplane the equation is:

$$g(X_i) = w^T X + b > 0 \rightarrow X_i \in Class\ A \quad (16)$$

while if the feature value is on the negative side of the hyperplane it becomes:

$$g(X_i) = w^T X + b < 0 \rightarrow X_i \in Class\ B \quad (17)$$

In figure 10 we can see the value of this linear function for the hyperplane, the support vectors as well as some random points in each class.

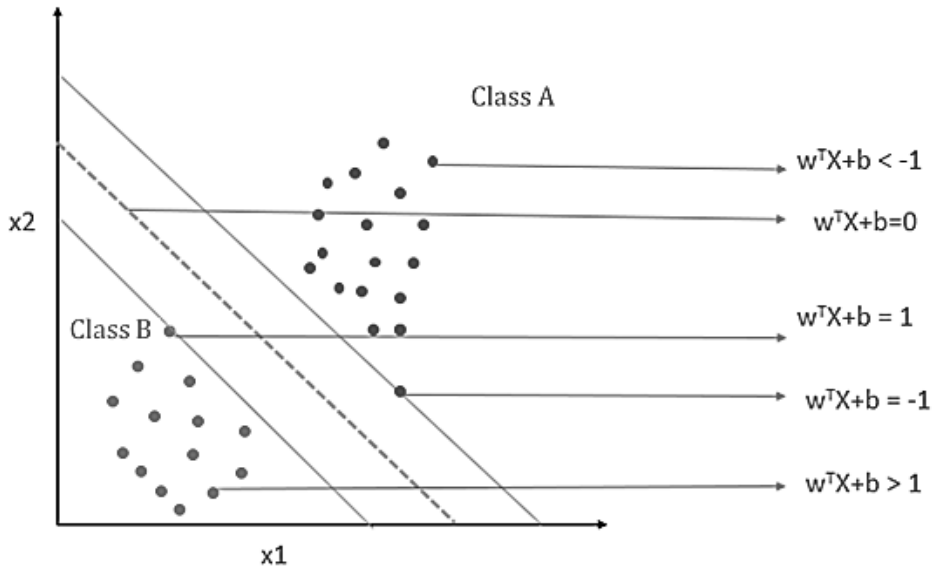

Figure 10. A hyperplane separating two classes in a 2D space. The margin as well as the support vectors of the hyperplane can be seen. Moreover, representative values of the boundary equation for various data points are illustrated.

2.4.3.3. Calculating the maximal margin classifier

As we discussed in the previous section the distance between the hyperplane and the support vectors is defined as the margin. In the case that we are studying, where our data are linearly separable, we can see that in fact one could draw an infinite number of hyperplanes between the data by simple rotating or traversing the decision boundary as can be seen in figure 11.

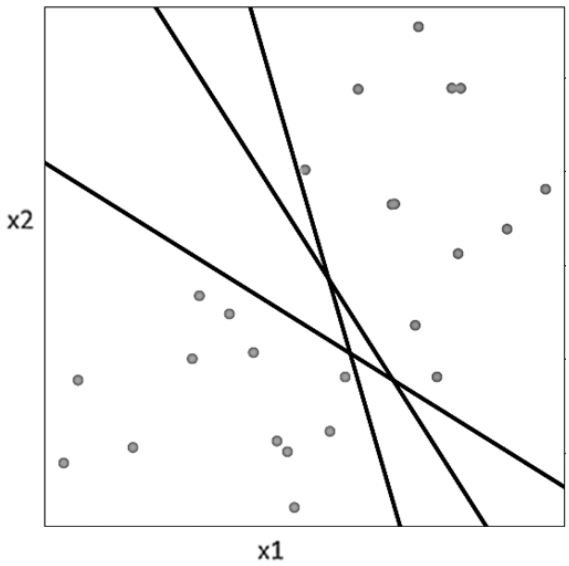

Figure 11. A representative number of candidate decision boundaries is illustrated for a D data set.

In order to choose the best classifier amongst these infinite options, we use the maximal margin criterion. Based on this criterion the hyperplane that provide the best separation between the two classes is the one that has the largest margin value (James et al., 2013). This can easily be understood based on intuition, since the hyperplane with the maximum margin is the one that is further away, on either side, from the classes that we wish to separate. It is possible to calculate the perpendicular distance from each data point in our observations and identify the support vectors and the margin distance. When calculating the maximum margin classifier, it is assumed that this classifier will still have a large margin on future test data as well. Even though the maximum margin classifier produces good results in most cases, it is also prone to overfitting when trained on large feature spaces (Müller, A., & Guido, 2018).

The calculations for the maximal margin classifier consist in fact of an optimization problem where the objective function to be maximized is the margin *M*:

$$maximize\ M \tag{18}$$

Subject to the following constraints:

$$\sum_{j=1}^{p} w_j^2 = 1 \tag{19}$$

$$y * (w^T X + b) \geq M$$

These constraints ensure that all observations lie on the correct side of the hyperplane and that no observation is closer to the hyperplane than the distance *M*. Details for solving strategies for this optimization have been published and are extensively available in literature (Rozenberg et al., 2012).

2.4.3.4. Non-perfectly separable classes - Support Vector Classifiers

It is easily understood form the analysis of the maximal margin classifier presented so far, that that approach could only work when observation can be perfectly separated by a hyperplane. In many cases however this is not possible, and we are instead interested in

a classifier that offers the best (but not a perfect) separation of the data points. Furthermore, even if two classes could be perfectly separated, the maximal margin classifier might lead to overfitting on the training data. To avoid this some degree of uncertainty is usually desired. This uncertainty allows our model to generalize better to new data and minimizes its sensitivity to the original training set. We can therefore think of the support vector classifiers (SVC) as a more efficient, less sensitive and more generalizable extension of the maximal margin classifier.

The optimization problem for a support vector classifier is like the one presented for the maximal margin classifier, but we can now relax the constraints and add a fuzzy factor that will allow our model to misclassify a given number of observations. Specifically, the optimization problem is described by the following equations:

$$maximize\ M \tag{20}$$

Subject to the following constraints:

$$\sum_{j=1}^{p} w_j^2 = 1$$

$$y * (w^T X + b) \geq M(1 - \epsilon_i) \tag{21}$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$$

Where ϵ are slack variables that allow some of the observation points to be misclassified and C is the tuning parameter that controls the number of misclassifications that we wish to allow for our SVC. Large values for the parameter C would lead to many observations to fall within the margin and therefore the model would rely on many support vectors. Such a model would, in theory, display low variance but higher bias. On the other hand, if a small value for C is chosen the SVC, a smaller number of points would fall within the margin and the model would rely on fewer support vectors as well. With this tuning the model would

display higher variance but potentially lower bias. An example with different values of C is illustrated in figure 12.

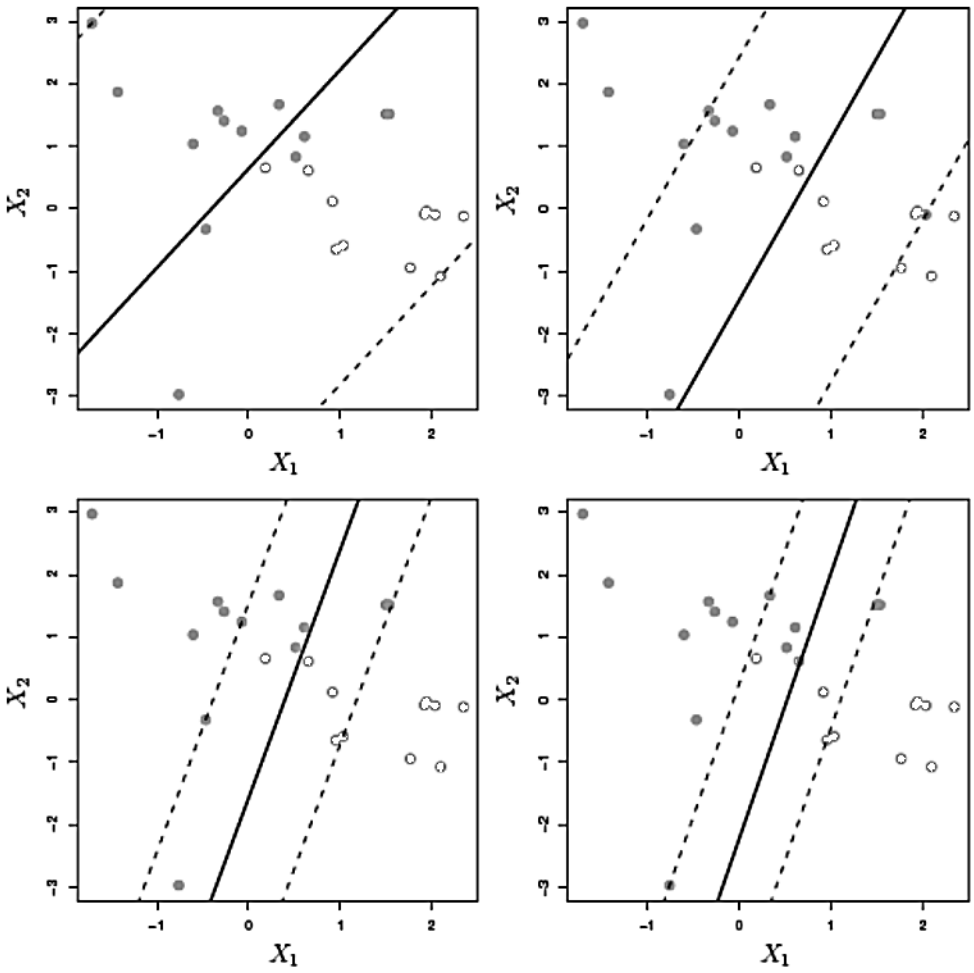

Figure 12. An SVC fitted on the same data set with varying values for the tunning parameter C. The value for the tuning parameter is higher on the top left and then decreases gradually for the top right, bottom left and bottom right panel. As the value of C decreases the extent of misclassified points is reduced.

2.4.3.5. Non-linearly separable data – Support Vector Machines

We have investigated so far, the basic theory and mechanisms used for data that can be linearly separated in two distinct classes, however this assumption is not always true for real data. In fact, in most real applications the linear classifiers would perform very poorly on predicted the desired class because they cannot account for the non-linearities in the data. Therefore, the SVCs can be further extended in order to accommodate for this non-linearities. This can be achieved by expanding the original feature space with the use of higher degree polynomials (quadratic, cubic etc.). For example, if we start with the features:

$$X_1, X_2, .., X_{p-1}, X_p$$

We can then expand this feature space by calculating higher order relationships between these features such as:

$$X_1, X_1^2, X_2, X_2^2, .., X_{p-1}, X_{P-1}^2, X_p, X_P^2$$

If we then apply a linear classifier in this enlarged space, it might be able to successfully separate the observations in distinct classes. It should be noted that the classifier used would still be a linear classifier but fitted on higher order feature space which means that in the original feature space the classifier would appear as non-linear. In the example above this would mean that in the original feature space the classifier would be a quadratic polynomial. Of course, higher orders of polynomial can be used until a satisfactory separation is achieved.

It is easy to see however that this simplistic approach faces a major challenge. Specifically, the need for an increased feature space with higher order of relationships (polynomial or other) can easily lead to computationally unmanageable problems. Addressing this challenge is exactly where the success of the Support Vector Machines lies.

SVMs attempt to use the higher order relationships that might exist in our data, but they do that with a computationally efficient way. This is what SVM practitioners and scientists call the kernel trick. The mathematics behind the kernel trick are relatively complex and have been analyzed extensively in the literature (Rozenberg et al., 2012). The main idea however is that one only needs to calculate the inner product only between the support vectors to capture the non-linear relationships in the data set. Because the support vectors are usually a very small subset of the original data set, this calculation more efficient than the original approach where the higher order feature space had to be first produced and then a linear function had to be fitted on this enlarge space. The final calculation for estimated a non-linear classifier is:

$$f(x) = \beta_0 + \sum_{i \in S} a_i \langle x, x_i \rangle \qquad (22)$$

where α are the correlation parameters between each pair of support vectors. Another major computational improvement is achieved by not calculating the inner product of the support vector directly. This is the heart of the kernel trick. Instead, the inner product can be calculated by a numerical approximation, with the use of various expansion series

(kernels). These kernels (numerical approximations) make the computation of the inner product very efficient. Furthermore, depending on the type of kernel used, one could approximate the relationship between the support vectors in a different way. Some kernels could produce better results in specific cases than others. Therefore, a data scientist can experiment with and choose the kernel that performs the best in each specific scenario. This can be seen in Figure 13 where a polynomial and a radial kernel are used to separate two classes.

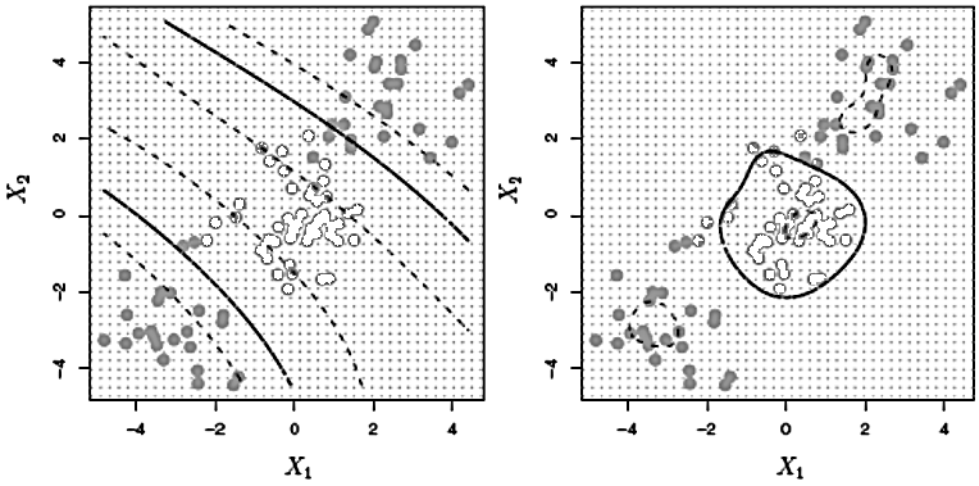

Figure 13. An SVM fitted on the same data set with two different kernels. On the left plot the kernel used is a 3rd degree polynomial and, on the right, it is a radial kernel.

3. Method Development and Implementation

In this chapter we will present the methodology, software tools and programming aspects that we used for developing the ANNs and SVMs models and for producing our simulated k-space signal of sphere size distributions

3.1.Development of NN and SVM Code

In this section we present the data preparation, data analysis, machine learning training, validation and testing methodologies that we followed to study the performance of NN and SVM on analyzing our k-space signal. We also discussed the tools selection process and the analytical approach that we chose for this work.

3.1.1. Machine learning pipelines

When investigating a ML problem, the aim is to develop a model that can be trained on an existing training data set and be latter used to perform prediction on new data. Moreover, this model is often part of a larger application with various components. This is a very complex process with various distinct steps that require different methodologies, skills and tools. This process, starting from the raw data and ending with a software application that uses a trained ML model is known as ML pipeline. The most important steps of a ML pipeline are illustrated in figure 14. These broad categories can be further broken down into more detailed steps.

Data preparation

This step comes before even considering a machine learning approach. It involves all the required steps that a data scientist/software developer needs to follow to acquire good quality data but also gain a very important insight on this data as well. Specifically, these steps include:

- Data collection and storage
- Data cleaning and feature engineering
- Exploratory data analysis and visualization

Typically, a data scientist would start this process by collecting raw data either from existing data repositories or by producing/collecting the data from a source. These raw data might include bad quality or irrelevant data which had to be cleaned and prepared (Friedman et al., 2009). Furthermore, not all the feature of the data are relevant for the analysis to follow so one might choose only a subset of the original feature space to work with. On the other hand, feature augmentation techniques also exist that allow scientists to enlarge their feature space by producing new features from the original ones (James et al., 2013).

Finally, it is usually extremely helpful to perform a manual exploratory data analysis and visualization to better understand the information hidden in the given data. This insight can prove to be essential for the following model selection step. In cases where the data are not well understood wrong ML approaches are chosen that lead to poor performance (Weber, 2020).

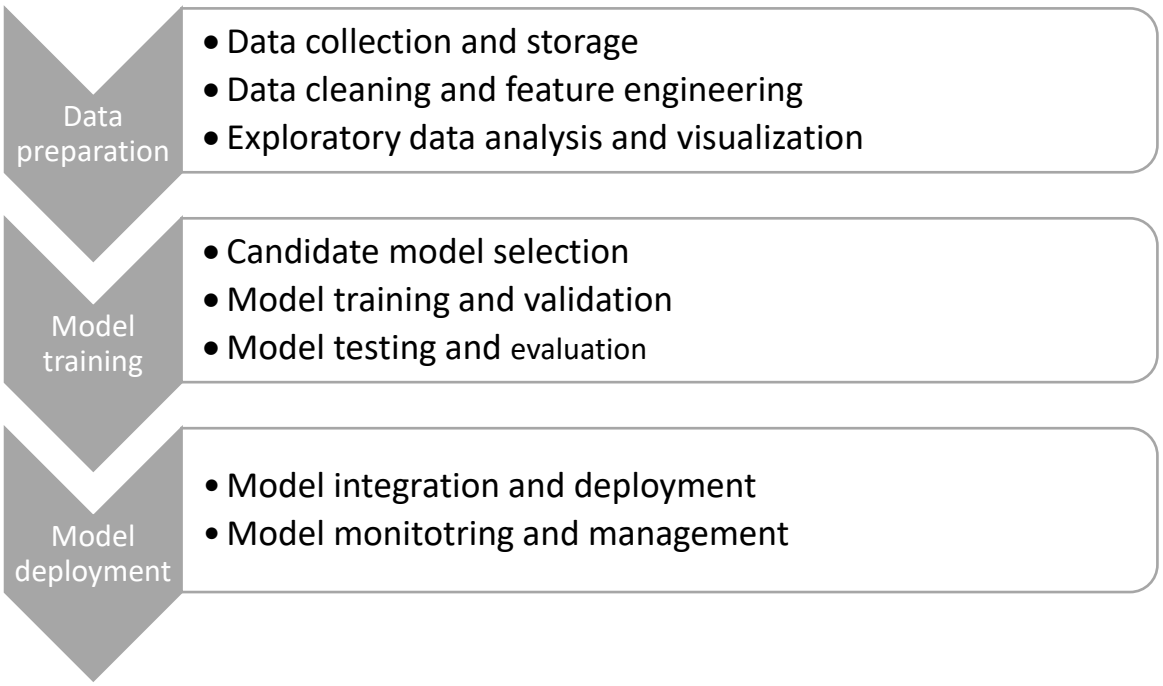

Figure 14. The main steps and processes involved in a ML pipeline starting from raw data and ending with model deployment and monitoring.

Model training

When the data pre-processing step is completed and after we have gained an insight into the data by performing a preliminary analysis, the machine learning training step follows.

This can also be said to include the following tasks:

- Candidate model selection
- Model training and validation
- Model testing and evaluation

The process begins by exploiting the insight that we gained from the data preparation step to identify potential ML algorithms that could tackle the problem at hand. After the candidate models have been identified the main training ad validation steps begin which are usually the core of a ML development task. In this step the models are trained based on a predefined data set. During their training we evaluate the effectiveness of our training strategy by validating the results with a subset of our training data. Again, numerous strategies exist for this step like train/test split, k-fold validation and others (Friedman et al., 2009). The validation process ensures that our training is leading to an improved version of our model while at the same time avoids overfitting and excessive computations with minimal performance benefits (James et al., 2013).

When the training step is completed, we then present our candidate models with various data sets that they have not seen before. It is important to avoid data leakage between the training and the testing steps because this could lead to false performance results. The models must always be tested against previously unseen data (Hannes Hapke, 2020). The performance of each method is evaluated using various performance metrics such as mean square error and mean absolute error for regression problems and accuracy, precision and recall for classification problems (Bishop, 2006).

The best performing model is selected to be used in the final application.

Model deployment

A ML model is usually the core of a functionality or even of an entire software application. As such it need to be integrated, deployed, and managed with the rest of the application. This is usually the area where software developers take over from data scientists as they are responsible for integrating the ML model with a bigger application and it needs to work and communicate with other components. More importantly the model performance must be constantly monitored and new iteration of the model have to be

constantly tested and deployed without disturbing the normal operation of the application.

In this project we do not aim to develop such an application but rather investigate the potential application and performance of NN and SVM on analyzing the k-space signal. Therefore, we will not discuss this part of the pipeline more in this thesis. The implementation details of such an application could however be investigated in future work.

3.1.2. Development tools and methodology

The work presented in this thesis focused on the first two steps of a ML project pipeline, namely i) data preparation and analysis ii) ML model training and validation. Specifically, our aim was to:

- Produce, prepare and analyze the simulated k-space signal from materials with a specific structure, namely dispersions
- Investigate the performance, advantages and challenges that ANN and SVMs present when trying to make predictions about material properties based on this k-space signal

Tools

Currently there is a variety of software tools that are available and help scientists address the challenges involved in each step of the ML pipeline. For the first two steps of the pipeline, where our work is focused, we decided to use the following main tools:

- Python 3. Multi-paradigm high-level interpreted language. Currently it is the most popular choice amongst Data and ML scientists (Géron, 2019; Lee, 2019).
- Pandas and NumPy. The two most popular Python modules for performing scientific computations (NumPy) and performing complex manipulation of large data-frames (Pandas).
- Seaborn and Matplotlib. Seaborn is a high-level data visualization library in Python which works very well with NumPy and Pandas data structures. It allows scientists to produce numerous graphs with minimal code. It is used extensively

to illustrate and analyze data. Seaborn is based on the more low-level Matplotlib library which we also used to customize our graphs.

- Scikit-learn. One of the most popular libraries for training, validating and testing ML models. Although it is not a performance-oriented library like some other alternatives (it does not support GPU acceleration for) it offers a very user friendly and uniform interface while at the same time it supports most of the current ML methods. This makes it the most popular choice for RnD and educational purposes in the Data and ML fields. In our case it allowed us to use the same high-level approach and develop the same pipeline for data preparation, training, validation and testing. Then we could simply swap in and out the ANN or VM models without having to rebuild the entire pipeline for each model. If someone wishes to improve the computational time of our models, after the investigation phase, it is possible to transfer the implementation to a more performance-oriented library like PyTorch, Keras or TensorFlow (Brauer, 2018)(Gulli & Pal, 2017)(Stevens et al., 2020).
- Jupyter Notebooks. This is a browser-based interactive computational environment which has recently become extremely popular with data scientists. It offers a platform where the user can create and run in real-tile code blocks, store data, display markdown text, images and graphs. This allows data scientists to have access to all the components of a Data/ML project in the same place with an easy interface. Combining code, data, graphs and text in a single location helps scientific communication greatly.
- Git. We used git and GitHub for version control. This also allows us to share the project in the future with other scientists. The repository of this project will be made public after the completion of the examination process (Kostas Ziovas, 2021).

Methodology

With the use of the tools described above we developed the scripts and modules necessary for simulating, visualizing and storing the k-space data as described in section 3.2.2. The simulation parameters that we had to specify are: i) sphere size distributions,

ii) noise levels, iii) sample sizes and iv) number of simulations to perform. The details and values for these parameters will be given in the "Results and Analysis" section.

After producing the desired data sets, we developed the data processing and ML model training pipeline. This includes:

1. Loading, formatting and scaling of data
2. Specifying the grid search parameters for each ML model. These parameters vary for each model, but they are the hyperparameters that a scientist has to tune in order to achieve the optimal performance for each ML model. Scikit-learn offer the same interface for conveniently creating grid search structures for any type of ML model.
3. Training and validating the ANN and SVM models using the k-fold validation method
4. Testing the performance of the optimally tuned model on a previously unseen data set
5. Storing the ML model for further analysis

The final step in our project was to perform a sensitivity analysis on our models ad identify their performance characteristics. We tested and visualized how the ANNs and the SVMs perform on analyzing unseen k-space data of various samples. The parameters that we tested are specific for each model and again will be presented in the "Results and Analysis" section.

Throughout this process we tried to use good coding practices and build reusable tools whenever possible. Our aim to have an automated, uniform and easy to use understand workflow. It is composed of Python code libraries, comments, text-block explanations, graphs and images for each step. This makes the process easy to follow for anyone who might be interested in studying or expanding the current work. As mentioned earlier we used git for version control and the entirety of this project will be publicly available as an open-source project in GitHub after the completion of the examination process. The GitHub repository address can be found in the '*References*' chapter (Kostas Ziovas, 2021).

3.1.3. Problem formulation

While developing our methodology we could choose to formulate the problem of estimating the sphere size distribution of a sample from its k-space signal as (i) a regression problem or (ii) a classification problem.
In the past a Bayesian method was used to analyze the k-space signal and in that work there was a predefined set of possible states that the model could predict based on k-space data. The problem was therefore a classification problem. However, it is more intuitive and of higher practical use to approach this task as a regression problem. By nature, the question that we are facing covers a continuous space of possible answers. There is not, in reality, a finite set of sphere size distribution that our system can have. It might be known that the values always fall within a specific range, but they could take any value in this range. So conceptually classification is not the most appropriate choice for this task.

Furthermore, as we will explain in the "Results and Analysis" section, by discretizing the problem in order to turn it into a classification task the performance degrades because the variance increases. The reason is that by using classification a sample that is misclassified will be assigned a value far different that its true value. This is because the classes are assigned at large intervals and there is not continuous value assignment as in a regression problem. Even small errors in the classification problem create a much worse overall model performance.

A classification formulation would be preferred if the property of the material that we are trying to estimate is discrete by nature, or if it is easier for humans to understand in a discrete form. For example, if we wanted to estimate the presence of a dispersed phase or not in our sample then it might be more natural to say that we have two classes one if there is a dispersed phase in the sample and one where there is not. Then for each sample analyzed we could get a clear 'YES' or 'NO' answer.

3.2.Development of K-Space Signal Simulation Code

In the previous chapter, section 2.4.1, we discussed the theory related to the 1-dimensional k-space signal. We also showed how this 1D k-space signal varies based on the sphere size distribution in our samples. In this section we discuss the how such a signal can be created through simulations, instead of performing real NMRI or X-Ray experiments.

3.2.1. Simulation vs Real Experiments

In order to test and prove the performance of the two new machine learning methods, we produced a series of simulated k-space data. Ideally real NMRI and X-ray data could have also been used. The simulated data however offer some compelling advantages:

1. In order to acquire real experimental data we needed access to specialized equipment which is only available in very few laboratories. This would mean that we had to collaborate with a laboratory or a different university which has such an equipment available. Moreover, a significant amount of time would be required to configure that equipment to our specific experimental needs while a source of funding would be necessary to cover the costs of using and maintaining that equipment. These requirements were not in the scope of this MSc thesis.
2. Simulating data on a computer is easier, cheaper and faster than performing complex and time-consuming experiments.
3. Numerically simulated data, assuming that they are sufficiently accurate representations of the real system, allowed us to precisely control, test and analyze our newly developed methods. They are therefore preferred during this initial proof-of-concept stage as they eliminate all effects and variability that could arise from experimental parameters.

3.2.2. Simulation Process Overview

In this section an overview of the method is presented with emphasis on the numerical implementation rather than the mathematical foundation which has been presented previously in a series of papers (Holland et al., 2011a, 2012; K. Ziovas et al., 2016). All numerical code has been written and implemented using Python 3 and run on an AMD Ryzen 5 2600 with 6 Cores (12 Threads). The most important Python libraries used, other

than the standard library, are: NumPy, Pandas, Seaborn, Matplotlib, Multiprocessing and Functools.

Initially, a model for the size distribution must be chosen. The choice of the model is dictated by the sphere size distribution of the physical system that is to be analyzed. Since we do not have a specific sample to work with, we chose the most common sphere size distribution type: a Gaussian size distribution (Campbell & Mougeot, 1999). By implementing a Gaussian model, it is possible to train, test and validate the performance of the ML methods with a very commonly found size distribution. These finding can then be easily adapted to a different size distribution if a specific use-case requires it.

Two assumptions have been made for the implementation of the numerical simulations:

i. The spheres are randomly distributed in space.
ii. A Gaussian sphere size model is assumed.

The assumption of random spatial distribution is incorporated in the theoretical analysis presented in (Holland et al., 2012), on which this method is based. However, it is possible to expand the model for non-random spatial distributions if this is desired. To fully describe a Gaussian distribution two parameters, have to be identified: i) the mean sphere diameter D_m and ii) the standard deviation σ. Since we are producing our samples through simulations, we chose to use an arbitrary size range for the spheres' D_m parameter ranging from 0.1 arbitrary length units (a.u.) up to 1 arbitrary length units (a.u.). The same was done for the standard deviation σ. These assumptions are expressed in equations 23 and 24.

$$0.1 \leq D_m \leq 1 \tag{23}$$

$$0.1 \leq \sigma \leq 1 \tag{24}$$

In order to produce the simulated sample, the following steps are followed:

1) A random number N of spheres is generated. The number of spheres in each sample is random in order to ensure that the ML models can predict the correct SSD independently of the sample size. Each of these N spheres is randomly sampled from a Gaussian distribution with a specified D_m and σ.

2) After a diameter has been assigned to a sphere the projection of this 3D sphere down to a 1D is calculated using the equation:

$$h(r,x) = \pi(r^2 - x^2) \tag{25}$$

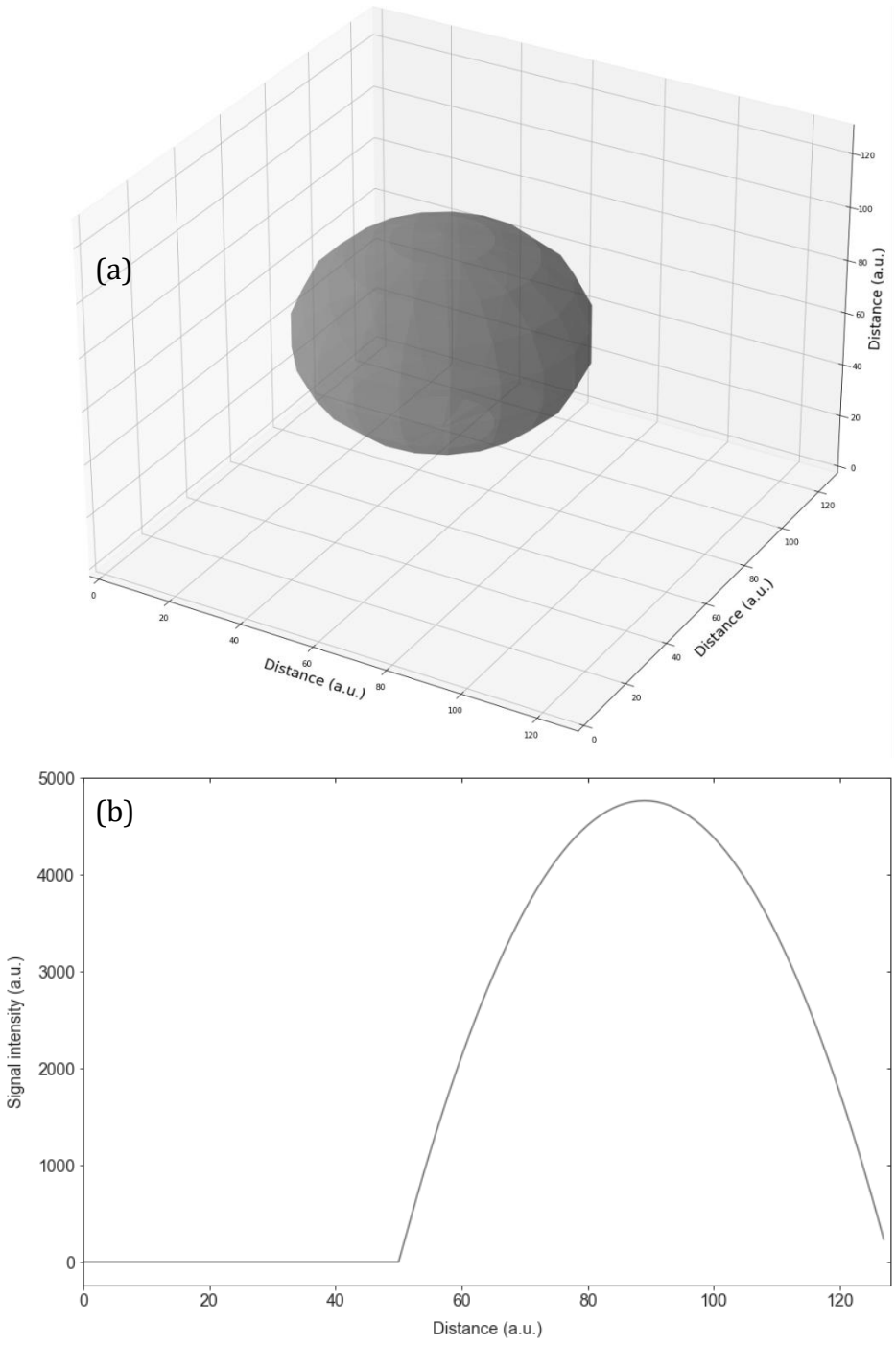

Figure 15. (a) A 3D representation of a single sphere with arbitrary units of length, (b) the 1D volume projection of the same sphere on the x-axis

where r is the radius of the sphere and x is the position in 1D space. A full mathematical analysis of this method is presented in the literature (Holland et al., 2012). This process is illustrated in figure 15. The 1D signal profile acquired in this manner is proportional to the volume of the original sphere at each position along the x-axis. This is a key point for this simulation because this produces a signal like what a water droplet would produce in an NMRI experiment. The reason is that the 1D signal acquired from a water sphere in a magnetic field during an NMRI experiment is directly proportional to amount of water at each position on the x-axis. In this case the NMRI signal in the x-axis would be directly proportional to the volume of the sphere. Our simulations are therefore good approximations of a real experiment (Holland et al., 2012; Ross et al., 2012a; K. Ziovas et al., 2016).

3) Each sphere is then shifted randomly within this 1D space and added on to the other sphere projections. When the process for all the spheres is completed the real space1D projection looks like that of figure 16.

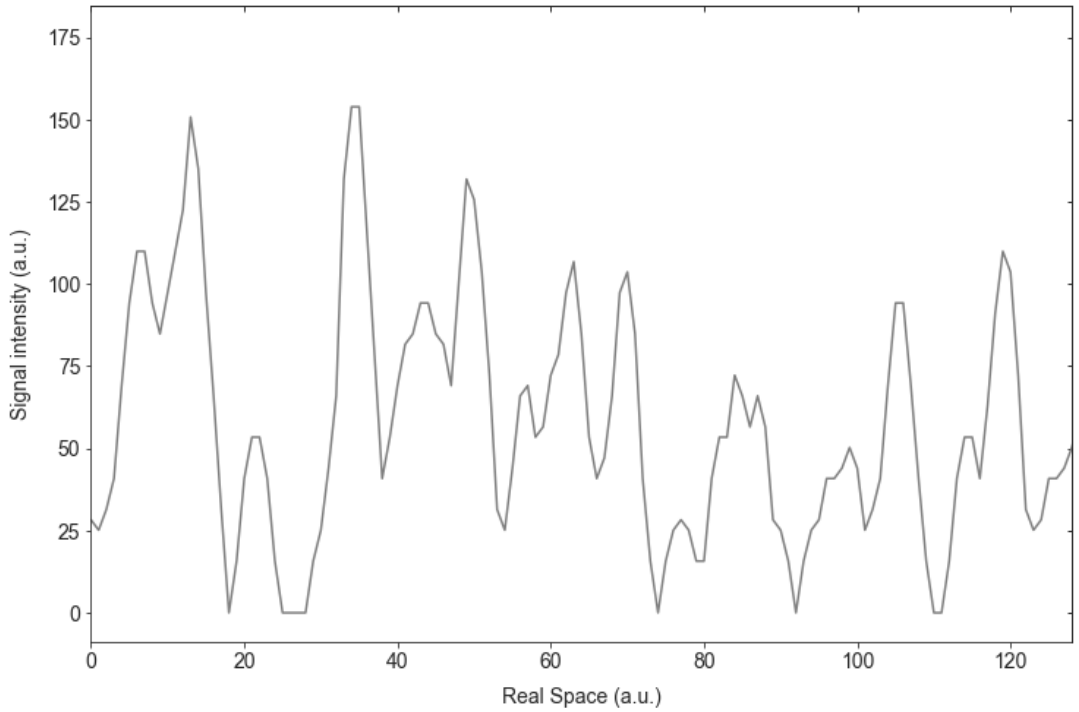

Figure 16. A 1D volume projection of N spheres on the x-axis using arbitrary units of length.

4) The 1D real space profile is Fourier transformed and the corresponding k-space signal is acquired. Figure 17 includes the k-space signal that is produced from the 1D profile of figure 16.

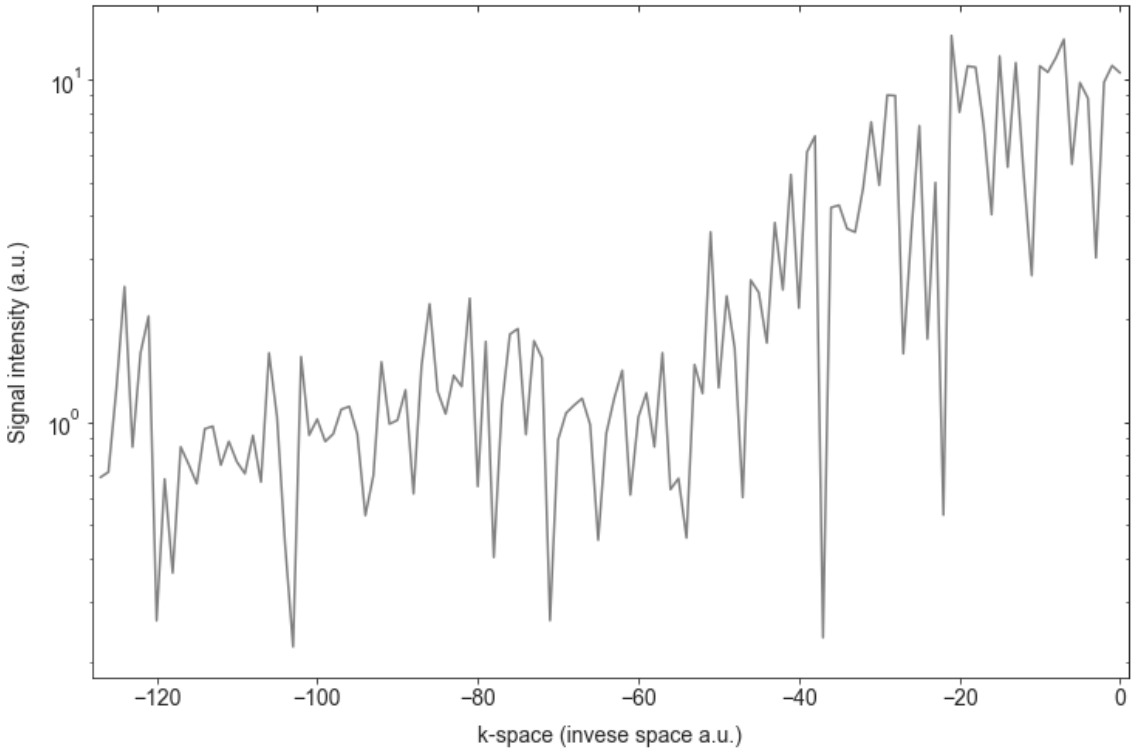

Figure 17. The k-space signal corresponding to a simulated sample with N spheres. The k-space data are produced by applying a FFT on the real-space data.

The simulation process that we presented in order to generate the k-space signal would be practically equivalent to the way that we would acquire the k-space signal from an 1D X-Ray profile experiment (K. Ziovas et al., 2016). In an X-Ray experiment we would first acquire the real space 1D intensity profile and then produce the k-space data. In contrast to this, if we performed a NMRI experiment our original data would be the k-space signal itself (Holland et al., 2011b, 2012) and from that we could produce the real space profile if desired. The theory behind each method was discussed in chapter 2, sections 2.4.1 and 2.4.2 but more information can be found in the literature (Holland et al., 2011b, 2012; Ross et al., 2012a; K. Ziovas et al., 2016).

4. Results and Analysis

In this section we present the results of the k-space signal simulations as well as the training, validation, and evaluation of the ANN and SVM models. The code used for every part of this thesis can be found from a link to public GitHub repository in the 'References' section (Kostas Ziovas, 2021).

4.1.K-Space Signal Simulations

For the k-space signal simulations we used the methodology described in the previous section. We performed a range of simulations for samples with various sphere size distributions. Table 1 includes all the parameters and their values used to produce the data that we used later to train and test our models.

Table 1. Parameters used for producing k-space simulations

	Value Range
Sphere Size Distribution Type	Gaussian
Mean Diameter	1-10 a.u.
Standard Deviation	0-25% of Mean Diameter
Noise Level	1-10% of Total Signal Intensity
Number of Spheres per Sample	40-4000
K-space Size	256 points
Total Number of Simulations	100000

As seen in table 1 the type of sphere size distribution was always gaussian and the size of the k-space sample simulated was always 256 points as this corresponds to a typical MRI experiment for this type of analysis (K. Ziovas et al., 2016).

The number of spheres in the simulation is the equivalent of the sample size if a real experiment was to be performed. We chose this range which covers two orders of magnitude of sample sizes to ensure that the analysis we perform is independent of the

samples sample. This is also in line with previously published work (Holland et al., 2012; Ross et al., 2012a).

The mean diameter and standard deviation were varied randomly and independently so that a representative data set is produced. The noise level was also varied within a range of 1-10% of the total signal intensity. The reason we calculate the noise as a percentage of the total intensity is that because the intensity of the signal varies on different k-space points we need a common point of reference. It may also seem that a noise level of 10% is small but the signal in k-space decays in a logarithmic fashion (Callaghan, 1993) as we move away from the center of k-space therefore even a noise level of 10% would completely dominate a very big part of the signal away from the center of k-space (Levitt, 2007)(Callaghan, 2011).

The simulations with all the varying parameters where combined in a single csv file based on which the training and validation of both ANN and SVM took place.

4.2.Neural Networks Training and Analysis of K-Space Signal

In this section we present the training, validation, testing and analysis of the ANN model.

4.2.1. Training and Validation

As we discussed in section 3.1, we formulated the task as a regression problem. We chose the Multi-Layer Perceptron (MLP) Regressor model and used Scikit-learn's interface to create a hyper-parameter grid search. Furthermore, we also used Scikit-learn's Multioutput Regressor (Géron, 2019) since in our analysis, we wish to predict both the mean diameter D of the sphere size distribution as well as the standard deviation σ. The Multioutput Regressor tool allows ML models that produce a single prediction output to be used for multiple prediction cases.

It should be noted, and it can be seen in the projects code, that before training our NN model we performed a pre-processing scaling step on the original data. Input data scaling is very important when training ML models because the models are sensitive to the absolute signal intensity that they are trained on. If the values of the input data varies

widely it has a detrimental effect on the ability of the model to identify any patterns (Hastie et al., 2016; James et al., 2013). The scaling that we performed is a simple MinMaxScaler which is available within the Scikit-learn package. This scaler simply re-maps all the signal within a (0-1) range.

4.2.1.1. Hyperparameter tuning

Table 2 includes all the hyperparameters that we tested. These hyperparameters are very important for the performance of our model and they are not estimated from the data themselves but instead they are specified by the data scientists who is developing the model. To be able to identify a close-to-optimal set of hyperparameters a grid search operation is performed like the one we present here. In this grid search all possible combination of the hyperparameters specified in table 2 are used.

Table 2. NN hyperparameter values tested

	Value Range
Activation function	Logistic, tanh, relu
Solver	lbfgs, sgd, adam
Learning Rate	constant, invscaling, adaptive
Momentum	0.1, 0.2, 0.4, 0.6, 0,8
Beta_1	0.1, 0.25, 0.5, 0.7, 0.9
Beta_2	0.0125, 0.025, 0.05, 0.1, 0.25, 0.5, 0.75
Alpha	0.0000125, 0.000025, 0.00005, 0.000075, 0.0001, 0.00025, 0.0005
Cross validation folds	5 folds

The values that we chose to investigate in the grid search cover the most used activation functions, solvers and learning rate strategies. Moreover, a representative range of values was chosen for the *beta* and *alpha* parameters in the range of values usually recommended (Hastie et al., 2016; James et al., 2013). It must be emphasized that the hyperparameter tuning is a very empirical process and even though some general

recommendations exist, it is up to the data scientist to decide how the tuning process will be performed (James et al., 2013). This is an area of ML where a lot of research is being done aiming at reducing the trial-and-error process and establishing a more rigorous scientific approach for the hyper-parameter tuning. However, at the time when this thesis is written no such generally accepted approach exists. Details about the characteristics of each of the hyperparameters and their performance characteristics can be found in the literature.

Apart from the training meta-parameters defined in the table above for the ANN model the most important decision is probably the structure of the NN. Specifically, we must decide how many layers and how many nodes in each layer our model will have. As is the case with the hyperparameters described above there is not generally accepted method that would yield the optimal NN structure. This is also a try and error process where a data scientist must experiment with various complexities for the NN (Bishop, 2006). In this project we tried to cover a sufficiently wide range of structures. Specifically, the combinations that we considered can be found in table 3:

Table 3. NN architectures tested

Layers and Number of Nodes			
L1:64, **L2**:128	**L1**:128, **L2**:256	**L1**:128, **L2**:64	**L1**:64, **L2**:32
L1:64, **L2**:128, **L3**:256	**L1**:128, **L2**:256, **L3**:512	**L1**:128, **L2**:64, **L3**:32	**L1**:256, **L2**:128, **L3**:64
L1:512, **L2**:256, **L3**:64	**L1**:512, **L2**:128, **L3**:64	**L1**:512, **L2**:64, **L3**:32	**L1**:512, **L2**:256, **L3**:128, **L4**:64
L1:512, **L2**:128, **L3**:64, **L4**:32	**L1**:512, **L2**:1024, **L3**:512, **L4**:128	**L1**:512, **L2**:1024, **L3**:512, **L4**:128, **L5**:64	**L1**:512, **L2**:256, **L3**:128, **L4**:64, **L5**:32
L1:1024, **L2**:512, **L3**:256, **L4**:128, **L5**:64	**L1**:1024, **L2**:512, **L3**:256, **L4**:128, **L5**:64, **L6**:32	**L1**:2048, **L2**:1024, **L3**:512, **L4**:256, **L5**:128, **L6**:64	**L1**:2048, **L2**:512, **L3**:256, **L4**:128, **L5**:64, **L6**:32
L1:4096, **L2**:2048, **L3**:512, **L4**:256, **L5**:128, **L6**:64	**L1**:2048, **L2**:1024, **L3**:512, **L4**:256, **L5**:128, **L6**:64, **L7**:32	**L1**:4096, **L2**:2048, **L3**:1024, **L4**:512, **L5**:256, **L6**:128, **L7**:64, **L8**:32	**L2**:8192, **L2**:4096, **L3**:2048, **L4**:1024, **L5**:512, **L6**:256, **L7**:128, **L8**:64, **L9**:32

It can be easily seen that there is a very large number of possible combinations. Especially if one combined the hyperparameter grid search with the Neural Network structure grid

search a total of C = 793,800 combinations must be evaluated. The computational time for each iteration depends on the specific value of the hyperparameters but even with a very optimistic time of 100 sec per parameter combination this entire space search would require ***2.5 years*** to evaluate on a single core. It is apparent that not all these combinations can be investigated.

In order to reduce computational time, we initially fixed the NN structure to a four-layer network with the following nodes:

L1:256 -> **L2**:128 ->**L3**:64 -> **L4**:32

where the letter L represents each layer by number and the value following are the number of nodes in each layer.

By fixing the NN structure to this initial configuration, we performed a hyperparameter grid search to identify the values that yield the best performance. This assumes that the hyperparameter grid search is not heavily correlated with the NN structure itself given that a structure relatively close to the optimal structure is used (Bishop, 2006)(Hastie et al., 2016). After identifying the desired values, we then performed a second grid search on all the NN configurations that are in table 3.

4.2.1.2. Results

The NN that achieved the highest score during training had the hyperparameter values shown in table 4. It's mean-square-error (MSE) on the test data was MSE = 0.0028 for the mean diameter estimation and MSE = 0.028 for the standard deviation. The mean-absolute error (MAE) was MAE = 0.037 for the mean diameter and MAE = 0.136 for the standard deviation.

It should be noted that, in order to have an easy direct comparison between the performance of our model on estimating the mean diameter D and the standard deviation σ, we scaled the labels of our data set. Specifically, when we created our data set the range of values for the diameter D were 0-10 au and for the standard deviation 0-2.5 au. For the MSE and MAE to be easily comparable we rescaled the labels assigned to our data so that

both the diameter *D* and the standard deviation *σ* are in a range 0-1au of distance. The same process we repeated latter during the testing step.

The number of layers and nodes is not the only characteristic of the NN structure. The arrangement of the layers and nodes is also important. For example, observe the following NN structures:

- **L1**:256 -> **L2**:128 ->**L3**:64 -> **L4**:32
- **L1**:32 -> **L2**:64 ->**L3**:128 -> **L4**:256
- **L1**:64 -> **L2**:128 ->**L3**:256 -> **L4**:32

Table 4. Parameters of best performing NN

Activation function	Relu
Solver	Adam
Learning Rate	Constant
Beta_1	0.5
Beta_2	0.05
Alpha	0.000075
NN Structure	**L1**:2048, **L2**:1024, **L3**:512, **L4**:256, **L5**:128, **L6**:64, **L7**:32
Training time	1.5 hours
Performance - Mean Diameter	MSE=0.0028, MAE=0.037
Performance - Standard Deviation	MSE=0.028, MAE=0.136

These three NN have the same number of layers and same number of nodes therefore are of comparable complexity. However, the structure of the NN is different in each case since the placement of the layers is different:

- In the first approach the nodes start from a high value and 'fan-in' towards the output
- In the second approach the nodes 'fan-out'
- In the third approach the node initially 'fan-out' and after the third layer the 'fan-in' again before the output

Each of these structures allow for a different amount of complexity at different parts of the NN. It has been shown that in some problems specific type of structures perform better than others (Deng & Yu, 2013). In our case we have observed that for comparable level of complexity, like the example above, the structure that produced the highest score was always a 'fan-in' structure.

4.2.2. Testing

The best performing ANN model identified by the analysis in the previous section was tested against a previously unseen part of our simulated data to evaluate its performance. The reason that this testing step is necessary is because it is common for ML models to score very well against their training data but then perform poorly on new data. Moreover, this is a sign of overfitting during the training process and therefore model testing is important before using a newly trained ML model. The test data that we are using is a reserved part of data from the original simulated dataset.

4.2.2.1. Results

Table 5 shows the performance summary of the NN model.

Table 5. NN performance results on test data

Performance - Mean Diameter	MSE=0.008, MAE=0.069
Performance - Standard Deviation	MSE=0.067, MAE=0.21
Estimation Time -1k Estimations	1.75 sec

The performance results on the new test data are showing an expected behavior. Both the estimation for the mean diameter D and the standard deviation σ have approximately double the error of the estimations obtained during training on the training data set. This is an expected behavior since all the models perform slightly better on the data that they have been trained on (Chollet, 2018; Géron, 2019). If the performance difference was larger, closer to an order of magnitude, then that would mean that our training process forced our model to overfit on the training set. We would need to revisit the training

process and improve the validation and implement a strategy to reduce overfitting such as:

- Model simplification
- Early stopping
- Different validation method
- Regularization
- Dropouts

Details and use cases for all these techniques can be found in the literature (Hastie et al., 2016). We can also visualize the output of our model over the test set so we can gain a better insight about the performance of the model on our data. Figure 18 shows the predicted mean diameter *D* of our NN model based on the k-space signal.

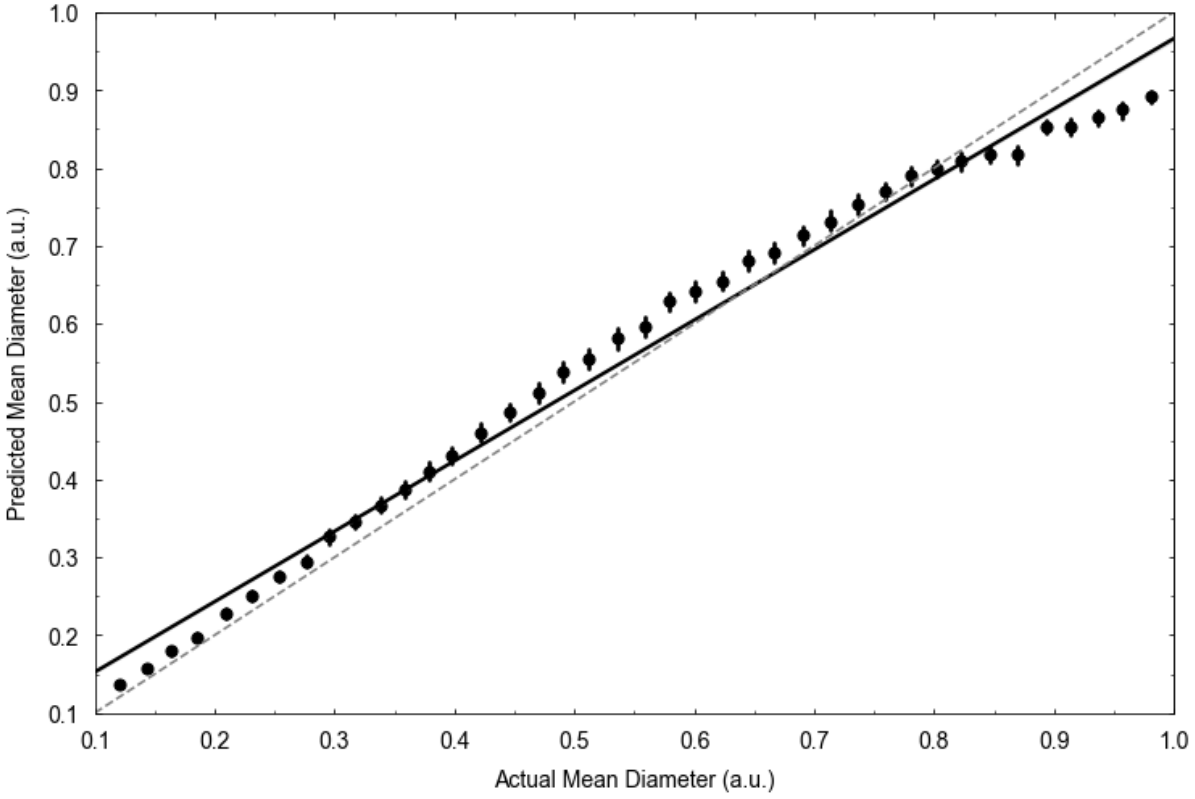

Figure 18. The NN-predicted mean diameter D plotted against the actual mean diameter. Each dot is a bin constituted from 40 individual measurements. The solid black line is the best fit linear prediction of the MLP regressor. The dashed line represents the ideal output if the model estimation for all the diameters was equal to the actual diameter of each sample.

Because we have tested our model over 1k samples it would be hard to get a clear picture by plotting all of them. So instead, we binned our prediction in 40 bins and each dot in

figure 18 represents the center of this bin. The error bars are the 95% confidence interval area around the center of each bin.

Figure 19 respectively shows the predicted standard deviation of our NN model based on the k-space signal. The same binning process was applied. As mentioned earlier in this section in order to be able to easily compare the performance and output of our model for both the mean diameter D and the standard deviation σ, we rescaled the region of values for both parameters so that they both cover a range of 0-1au of distance. This also allow direct comparison of their MAE and MSE values.

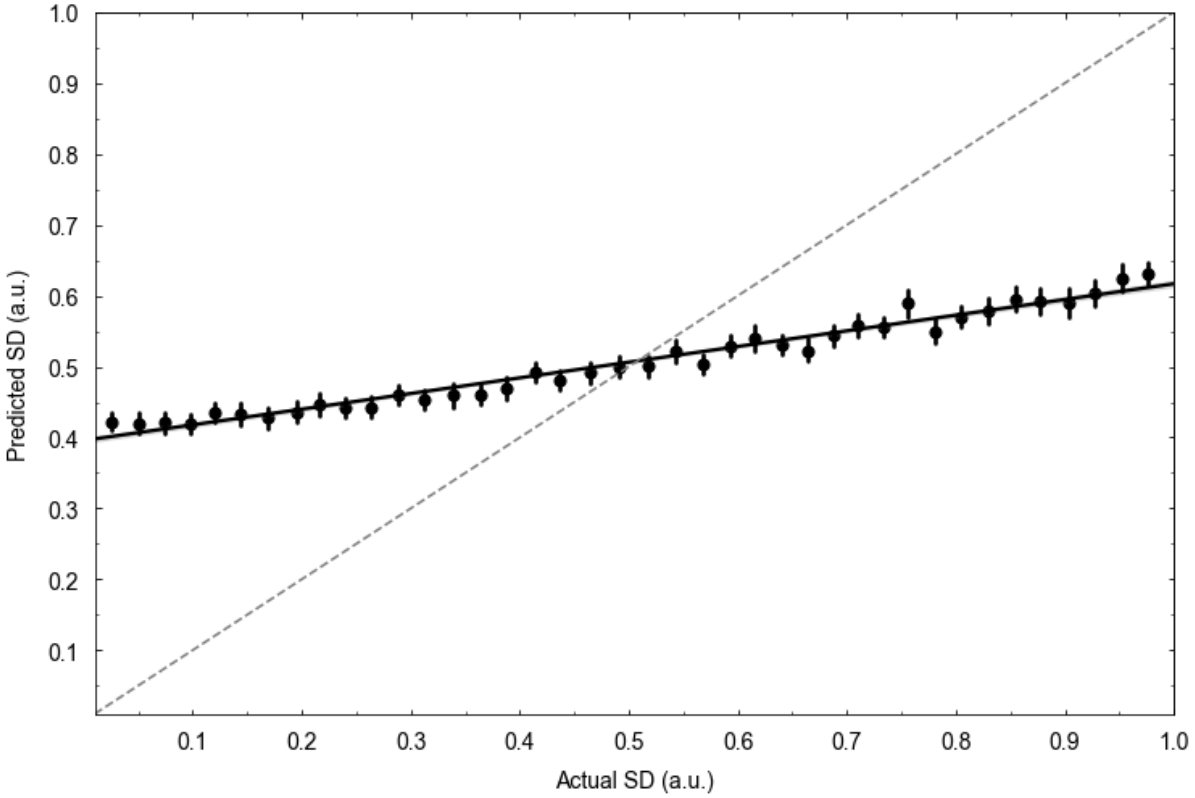

Figure 19. The NN-predicted standard deviation σ plotted against the actual standard deviation σ. Each dot is a bin constituted from 40 individual measurements. The solid black line is the best fit linear prediction of the MLP regressor. The dashed line represents the ideal output if the model estimation for all the σ was equal to the actual σ value for each sample.

4.2.2.2. Discussion

From the results presented in section 4.2.2.1 we can see that the ANN performs well on the task of predicted the mean sphere size of the distribution present in the sample under analysis. Despite some small fluctuations throughout the range of values, the diagonal line of expected diameters follows the model predictions very closely. Moreover, the

variance of the predicted values for each bin of sizes as they are represented by each dot in figure 18 is very low. The 95% interval error bars are almost not visible on most points, which mean that the model offers high repeatability. This good performance is also captured in the values of the MAE and MSE for the mean diameter as shown in table 5.

A more interesting and not so easy to understand behavior is observed when focusing on the predictions of the standard deviation for each sample. Our model displays again great repeatability with very high confidence and very 'tight' 95% confidence intervals. However, it is consistently predicting standard deviation values very far away from the real value. With a closer inspection one can see that the model is attempting to flatten out the prediction line to a constant value of about 0.5 au, exactly in the middle of the value range. This is a very different behavior than any other model (for example the Bayesian models) has ever reported before (Holland et al., 2011a, 2012; Ross et al., 2012a, 2012b). The MSE and MAE values are also much higher for the standard deviation than what they were for the mean diameter predictions. One explanation for this behavior might could be given if we consider the role that the parameters mean value μ and the standard deviation σ play in a Gaussian distribution. The mean value represents the value around which the distribution is centered. As a Gaussian distribution converges towards this mean value as the standard deviation grows smaller and smaller until the mean value ends up describing all of the spheres in a sample. In that case the Gaussian distribution would have converge to a uniform distribution. The standard deviation on the other hand is itself in a sense a measurement of uncertainty. It provides information for the degree of divergence of a Gaussian distribution from this ideal mono-disperse distribution where all spheres would have the same diameter as the mean value.

Given this analysis it might be clearer why the ANN is attempting to adopt an agnostic approach towards predicting the standard deviation. The standard deviation is treated as a value of uncertainty over the prediction of the mean diameter. The ANN model is trying to ignore this uncertainty and predict a value for the mean diameter that would more accurately describe the sample if all the samples had the same standard deviation! Therefore, the values that the ANN predicts for the standard deviation converge towards the value in the center of the range for the standard deviations. To phrase this differently the ANN seems to assume that all samples have approximately the same standard

deviation (~ 0.5 au) and based on that it is estimating the mean diameter that would most accurately describe that sample.

Of course, this is only an intuitive analysis based on the fundamental properties and the statistical meaning that the mean value and the standard deviation have for a Gaussian distribution. A more rigorous and in-depth analysis could be performed in the future to explain why this approach is favored by the ANN.

4.2.3. Sensitivity Analysis

After identifying and testing the best performing ANN model we performed a sensitivity analysis to gain a deeper insight into the model performance and how this could be affected by various parameters.

4.2.3.1. Parameters investigated

For this sensitivity test we monitored how the MAE, the MSE, the fitting (training) time and the prediction time of our model changed based on:

- The number of simulations used for training the model
- The ANN structure complexity (number of layers and number of nodes)
- The noise level present in the signal

We also investigated the effect of the standard deviation on the accuracy of the predicted mean diameter. This sensitivity tests were done following the insights gained by the analysis presented in section 4.2.2, where we show that the ANN adopted an agnostic approach towards the standard deviation but achieved very good results on the mean diameter. It was therefore important to investigate the relationship between these two parameters.

4.2.3.2. Results

<u>Number of simulations</u>

As seen in figure 20 the number of experiments used to train the NN is following the law of diminishing returns where initially the MSE and MAE are decreasing rapidly as the number of experiments is growing. However, after ~5,000 experiments the rate of

improvement slows down and reaches a plateau at ~10,000. After this point there is close to zero reduction in the error. At the same time the time required to train the NN increases rapidly, in a near-logarithmic rate, with very little gain in performance after ~10,000 experiments. Therefore, for type of data that we investigated in this thesis a number between 5,000 and 10,000 experiments offers the best performance-to-computational requirements ratio.

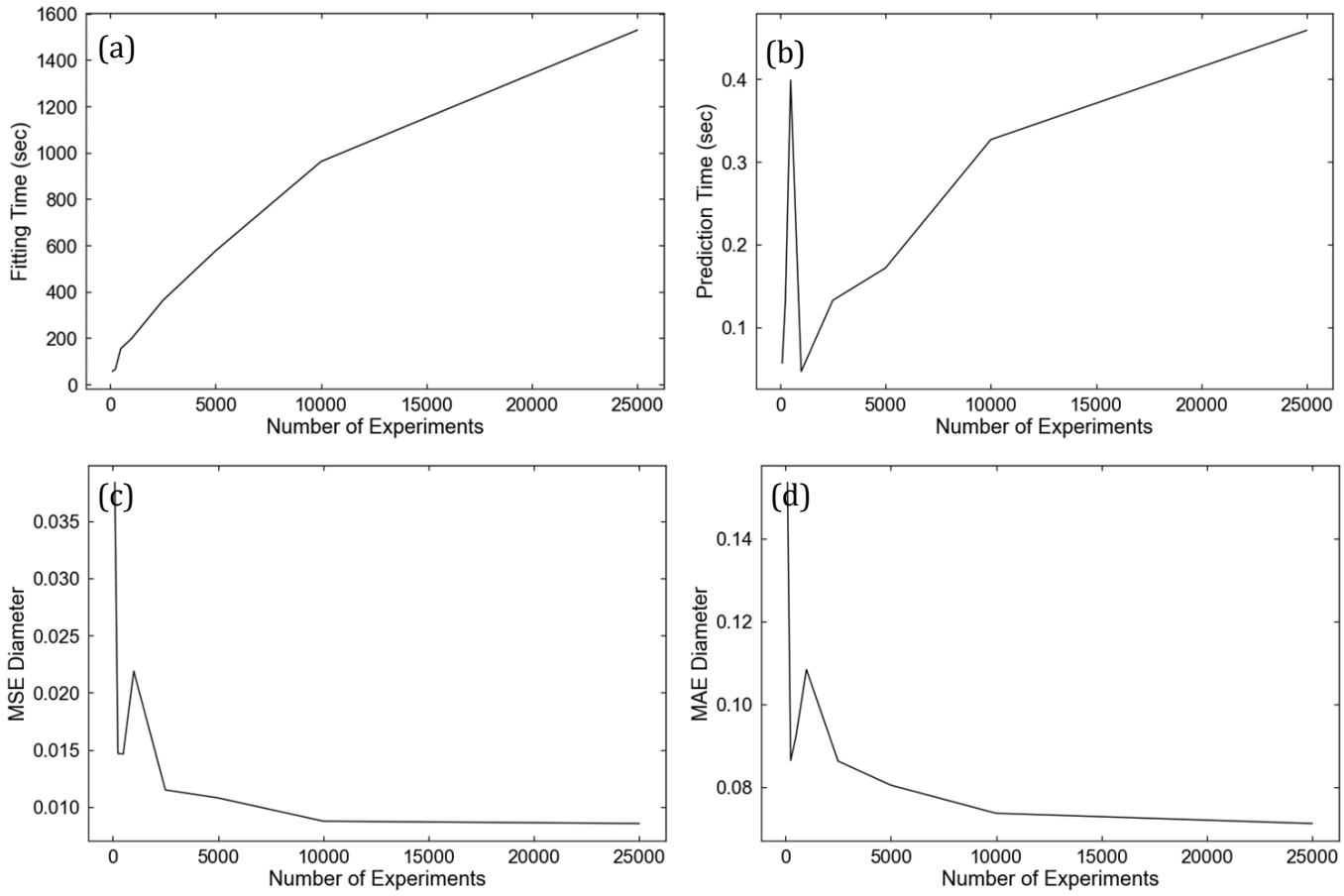

Figure 20. Effect of the number of experiments used during training on the (a) Fitting time, (b)Prediction time, (c) MSE and (d) MAE of the NN model.

Model complexity

Another very important parameter, as we already discussed, is the complexity of the NN. The number of layers and nodes used in the NN architecture have a different effect both on its performance as well as on the computational requirements. We can see in figure 21 that the performance improvement follows the same inverse logarithmic rate as we observed previously for the number of experiments. The difference in this case however is that the increase in computational requirements does not follow a quasi-log relationship as before but instead it seems to follow an exponential rate. This means that

for a given increase in the model complexity the computational requirements increase with an exponential rate therefore rapidly penalizing models with increased complexity.

By inspecting the graphs in figure 21 we see that the inflection point where the computational time increases very fast is for NN with 6 to 7 layers.

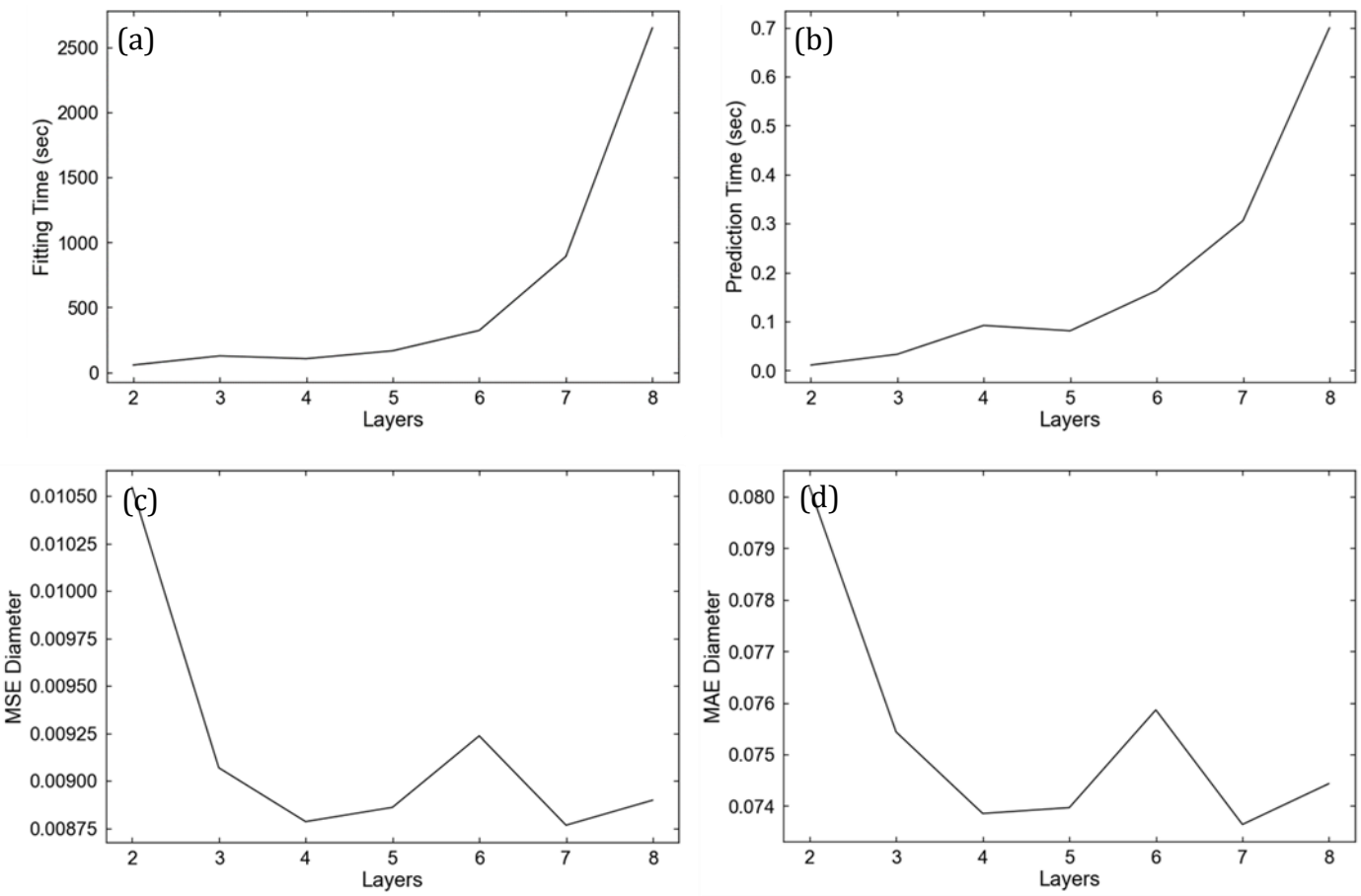

Figure 21. Effect of the number of layers of the NN used during training on the (a) Fitting time, (b)Prediction time, (c) MSE and (d) MAE of the NN model.

In the same time very small improvements in performance are observed for NN with more than 4 layers. The seven-layer architecture however produces the best performance both for MSE and MAE without requiring extremely large computational times. We chose this architecture in our thesis to achieve the maximum performance while maintaining relatively low computational requirements. Any architecture with more than 7 layers offers no advantages. If someone wanted to achieve the best performance-to-computational requirements ratio an architecture with 4 layers should be chosen as it offers only slightly worse MSE and MAE but with half the computational requirements.

Noise level

The set of graphs in figure 22 shows the effect of noise on the performance of the NN. The noise level is not a parameter that a scientist or engineer can control. There are ways to improve the signal-to-noise ratio (SNR), or noise-to-signal (NSR) in our case, by increasing the number of samples acquired, increasing the sample size and by using more sensitive equipment amongst other (Callaghan, 1993). All these solutions have a limit upon further improvement cannot be achieved. Therefore, it is important to know how well our NN can perform in situations with varying noise levels.

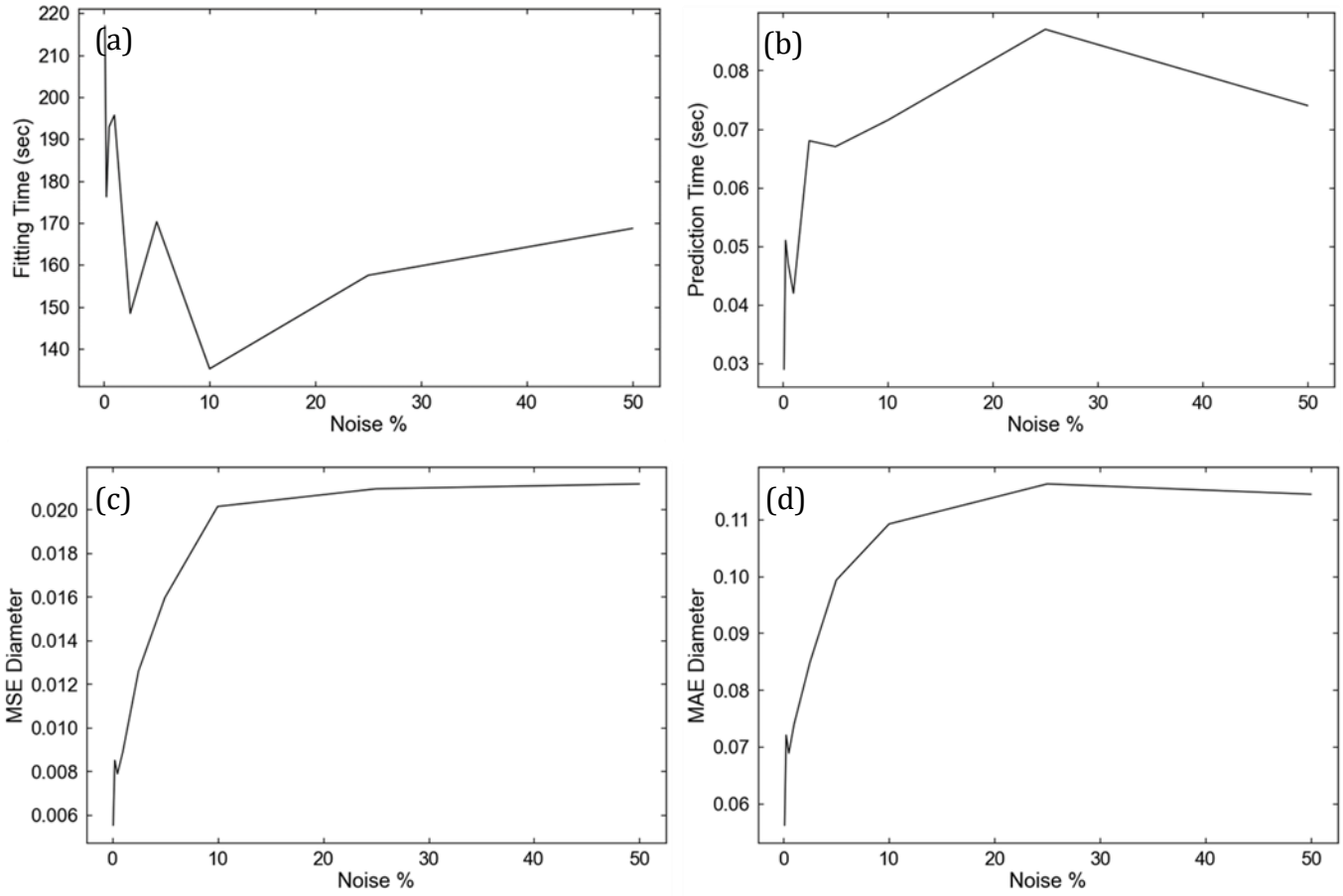

Figure 22. Effect of the noise level in the data used during training on the (a) Fitting time, (b)Prediction time, (c) MSE and (d) MAE of the NN model.

The MSE and MAE increase with a logarithmic rate and plateau at about 20% NSR. This value may seem low but as we already discussed the k-pace signal decays with a logarithmic rate away from the center of k-space. The NSR is measured between the center of k-space and the edges of k-space where the signal level could be many orders of magnitude different. Therefore, a NSR of 20% between k-space center and edges is

considered a very high value (Callaghan, 1993, 2011). This means that the NN can perform a regression on the k-space data even based on extremely noise dominated data.

It is interesting to observe the effect of noise on training and prediction times. The prediction time is as expected negatively affected; however, the training time displays a more interesting behavior where for smaller values of noise, the training time decreases as the noise level increases until the noise reaches a level of 10%, where the training time starts increasing again. An explanation of this could be related to the difference in the signal level as we move away from the center of k-space.

As we already discussed the signal has values orders of magnitudes different far away from the center of k-space. As we start from a pure signal and slowly increase the noise level the points far away from the center are affected first. These points are quickly dominated by the noise therefore the NN ignores them during training. In practice therefore, as the noise increases the number of points that our NN considers is reduced creating an apparent reduction in training time. This hypothesis is supported by the fact that at the same time the MSE and MAE error increase rapidly since the NN has fewer points to base its predictions on. It can be seen in figure 23 how noise-dominated the signal becomes when comparing a sample with 0.1% of noise to signal level compared to a sample with 10% noise to signal ratio.

It is obvious from this graph that as the noise level keeps rising and reaches 10% the magnitude of the noise becomes comparable to the signal level of the points near the center. At this noise level the entire signal is affected therefore the NN cannot simply ignore points anymore but instead it tries to perform the best prediction based on noise affected data points. This then begins to affect the training time since the model must now learn to identify the k-space pattern from noisy data. The training time increases, and the MSE and MAE begin to approach a plateau since all points are already affected by noise

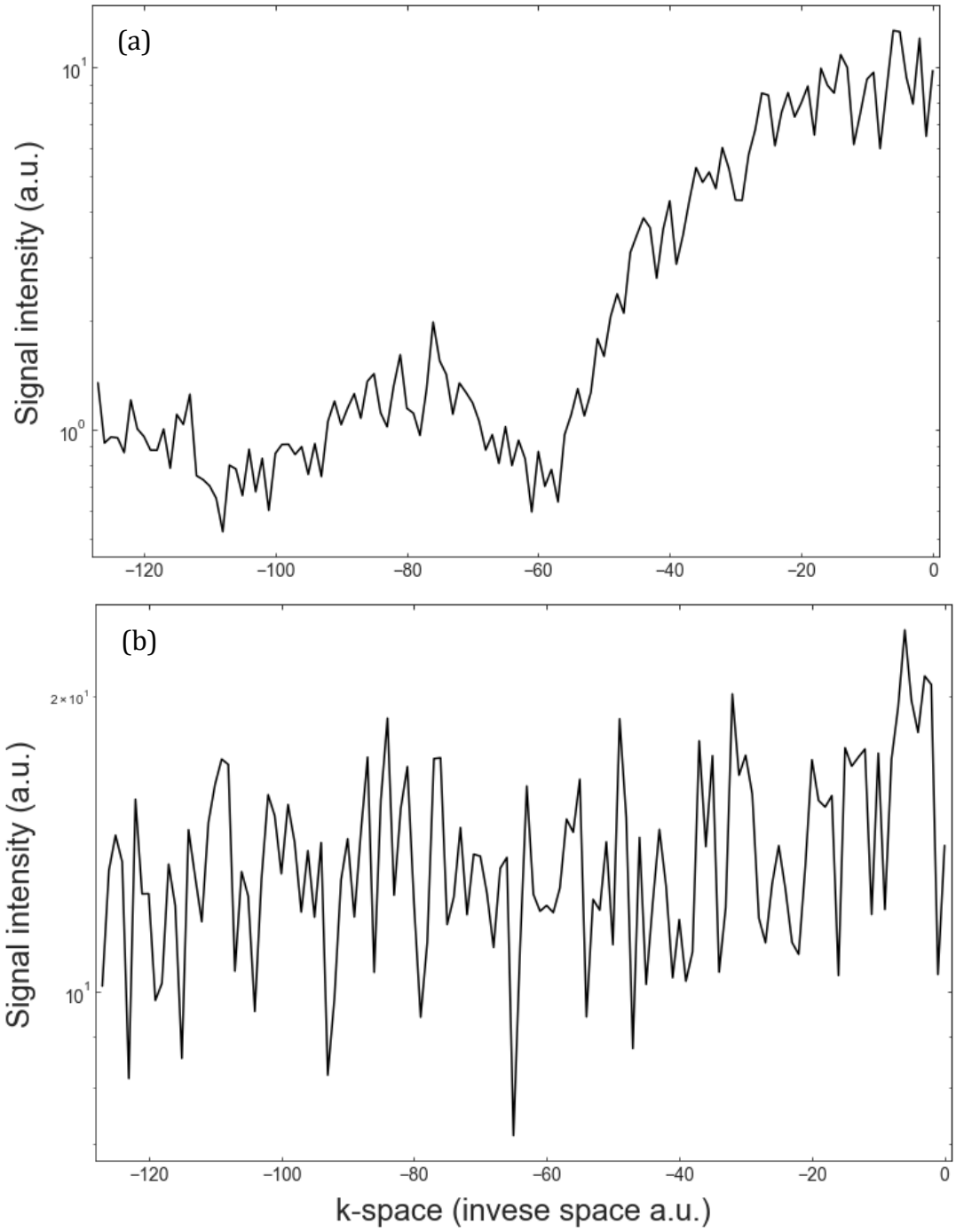

Figure 23. The k-space signal of a sample with mean sphere diameter D=6 au with (a) 0.1% noise-to-signal ratio and (b) 10% noise-to-signal ratio

Mean diameter-standard deviation correlation

In section 4.2.2 we presented the NN estimation against the real values both for the mean diameter D as well as the standard deviation σ of the sphere size distributions in our samples.

We explained how the model offers an improved accuracy for the mean diameter but converges towards an agnostic approach for the estimation of the standard deviation, since the parameter σ can be viewed itself as a representation of uncertainty over the mean diameter. The two parameters are corelated so in order to investigate the effect of standard deviation the prediction we split the data based on the standard deviation in 4 parts with standard deviation i) $0 < \sigma \leq 0.4$ au, ii) $0.4 < \sigma \leq 0.6$ au, iii) $0.6 < \sigma \leq 0.8$ au and iv) $0.8 < \sigma \leq 1$ au. We can see the results in figure 24.

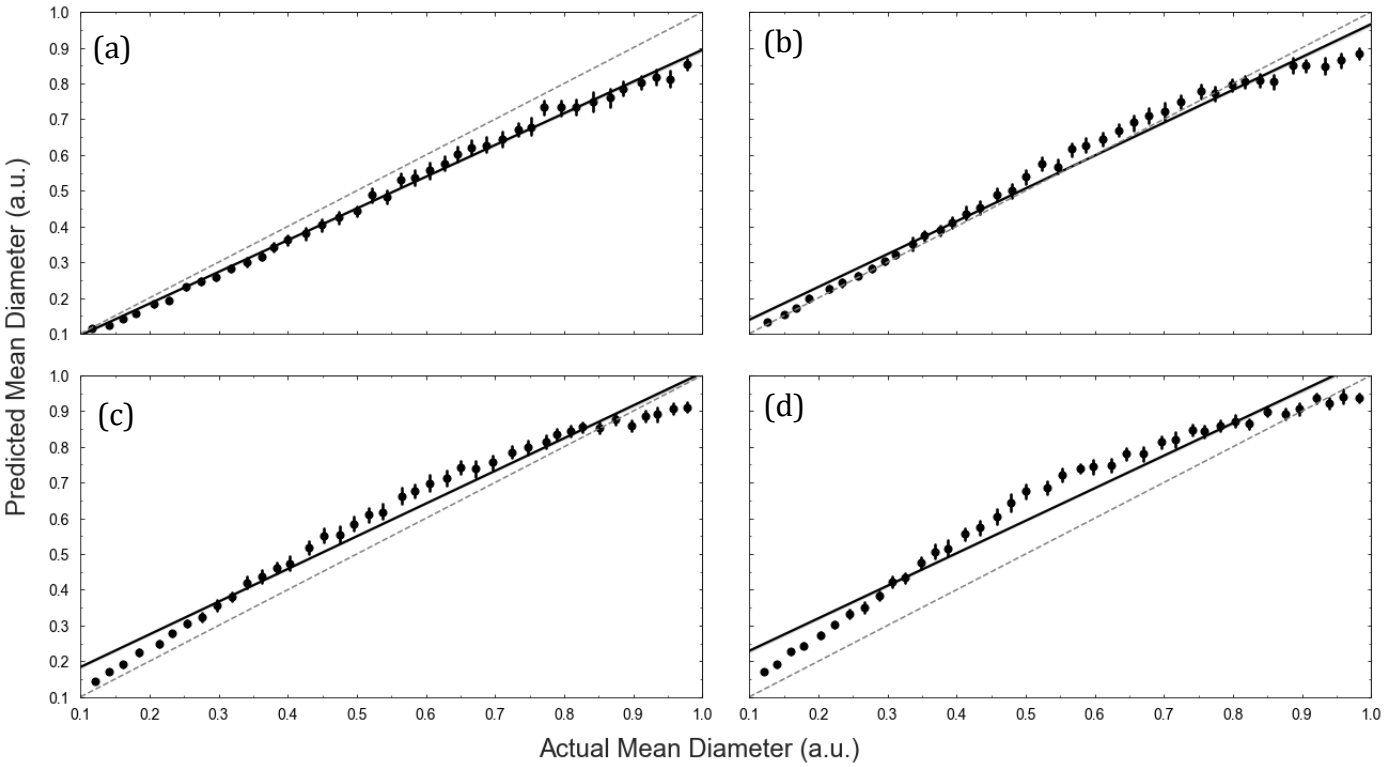

Figure 24. . The NN-predicted mean diameter D plotted against the actual mean diameter. Each dot is a bin constituted from 40 individual measurements. The solid black line is the best fit linear prediction of the MLP regressor. The dashed line represents the ideal output if the model estimation for all the diameters was equal to the actual diameter of each sample. Each plot corresponds to samples with different range of σ values (a) $0 < \sigma \leq 0.4$, (b) $0.4 < \sigma \leq 0.6$ au , (c) $0.6 < \sigma \leq 0.8$ au and (d) $0.8 < \sigma \leq 1$ au

The NN is trying to find a middle ground between the samples with different σ ranges since the range $0 < \sigma \leq 0.4$ au seems to produce a small underestimation of the mean diameter, the $0.8 < \sigma \leq 1$ au is overestimated and the two graphs in the middle ranges are more accurately estimated.

This analysis provides some ground for future work where a different approach could be taken. Instead of having a single NN model for the entire range of standard deviations a model ensemble could be trained on different ranges of standard deviations. During the

analysis step when we would like to extract a prediction for a sample, the candidate sample would be processed by all models. The model that is trained on the standard deviation range closer to the real standard deviation of the sample will produce the smallest error. This is a modern approach which produces more accurate results in most cases with the tradeoff of requiring to train and test against multiple models instead of a single model. Given the current state of computing this is not a particularly hard task in most cases (Bishop, 2006).

4.3.Support Vector Machines Training Analysis of K-Space Signal

In this section we followed the same overall process as the one presented in section 4.2 for the ANN model. This is a result of the streamlined ML pipeline-based approach that we implemented. The pre-processing, training, validation, testing and analysis steps were implemented in such a way that any model can be easily swapped in and replace another ML model without requiring major changes in the pipeline. The only major change between the models is around the specific hyper-parameters that we tuned and tested since they differ between ML models.

We will go through the same steps presented in section 4.2 and we try to avoid a duplication in the analysis. We will reference the discussion already presented in the previous section where necessary and expand it with information specific about the SVM model.

4.3.1. Training and Validation

For the SVM analysis we chose the SVM Regressor model and used Scikit-learn's interface to create a hyper-parameter grid search. Like the ANN model, we also used Scikit-learn's Multioutput Regressor (Géron, 2019) since we wish to predict both the mean diameter D as well as the standard deviation σ.

Before training our SVM model we performed the same pre-processing scaling step on the original data that we discussed in section 4.2.1. The scaling that we performed is a simple MinMaxScaler which is available within the Scikit-learn package. This scaler simply re-maps all the signal within a (0-1) range.

4.3.1.1. Hyperparameter tuning

Table 6 includes all the hyperparameters that we tested. To be able to identify a close-to-optimal set of hyperparameters a grid search operation where all possible combination of the hyperparameters specified in table 6 are used.

Table 6. SVM hyperparameter values tested

	Value Range
Kernel function	Linear, Polynomial, Radial
C	0.05, 0.075, 0.1, 0.125, 0.175, 0.25
Gamma	scale, auto
Degree of polynomial (for polynomial kernel)	2, 3, 4, 5
Epsilon	0.005, 0.075, 0.01, 0.0125, 0.025

The values that we chose to investigate in the grid search cover the most used options for the kernel function (James et al., 2013). Moreover, a representative range of values was chosen for the other parameters.

It can be seen by simply comparing table 6 with table 2 of the NN model that a much smaller number of hyper-parameters needs to be tested for the SVM compared to the NN which reduces computational time required during the training phase. This could be seen as a potential advantage of the SVMs over the NN (Battineni et al., 2019; Deng & Yu, 2013; Otchere et al., 2021).

4.3.1.2. Results

The SVM that achieved the highest score during training had the hyperparameter values shown in table 7. It's mean-square-error (MSE) on the test data was MSE = 0.008 for the mean diameter estimation and MSE = 0.068 for the standard deviation. The mean-absolute error (MAE) was MAE = 0.064 for the mean diameter and MAE = 0.21 for the standard deviation. The training time required for the SVM is 15 sec for the optimal set of meta-parameters whereas the ANN required 1,5 hours! This is a very significant

difference in favor of the SVM. The kernel trick reduces the number of require computations resulting in a very efficient ML model (James et al., 2013).

Table 7. Parameters of best performing SVM

Kernel function	Radial
C	0.125
Gamma	scale
Epsilon	0.0125
Training time	15 sec
Performance - Mean Diameter	MSE=0.008, MAE=0.064
Performance - Standard Deviation	MSE=0.068, MAE=0.21

4.3.2. Testing

The best performing ANN model identified by the analysis in the previous section was tested against a previously unseen part of our simulated data to evaluate its performance. The test data that we are using is a reserved part of data from the original simulated dataset.

4.3.2.1. Results

Table 8 shows the performance summary of the SVM model.

Table 8. SVM performance results on test data

Performance - Mean Diameter	MSE=0.008, MAE=0.07
Performance - Standard Deviation	MSE=0.069, MAE=0.22
Estimation Time -1k Estimations	1.75 sec

The performance results on the new test data are showing very similar characteristics to the ANN. In fact, the error values are almost identical between table 8 and table 5. Despite

being much more computationally efficient therefore the SVM produces identical estimations compared to the ANN when tested on new data.

In contrast to this the performance of the ANN was much better (the error was almost half) compared to the SVM on the training data set. This could be a very good indicator that we might have overfitted our ANN on our training data. Following the discussion of section 4.2 about the model complexity that we chose for our ANN we could in fact choose a simpler model with 4 layers which could perform equally well but with smaller required training time. In the future if someone wishes to perform a practical implementation of the work presented in this thesis a 4-layer model could be chosen.

We can also visualize the output of our model over the test set so we can gain a better insight about the performance of the model on our data. Figure 25 shows the predicted mean diameter *D* of our SVM model based on the k-space signal.

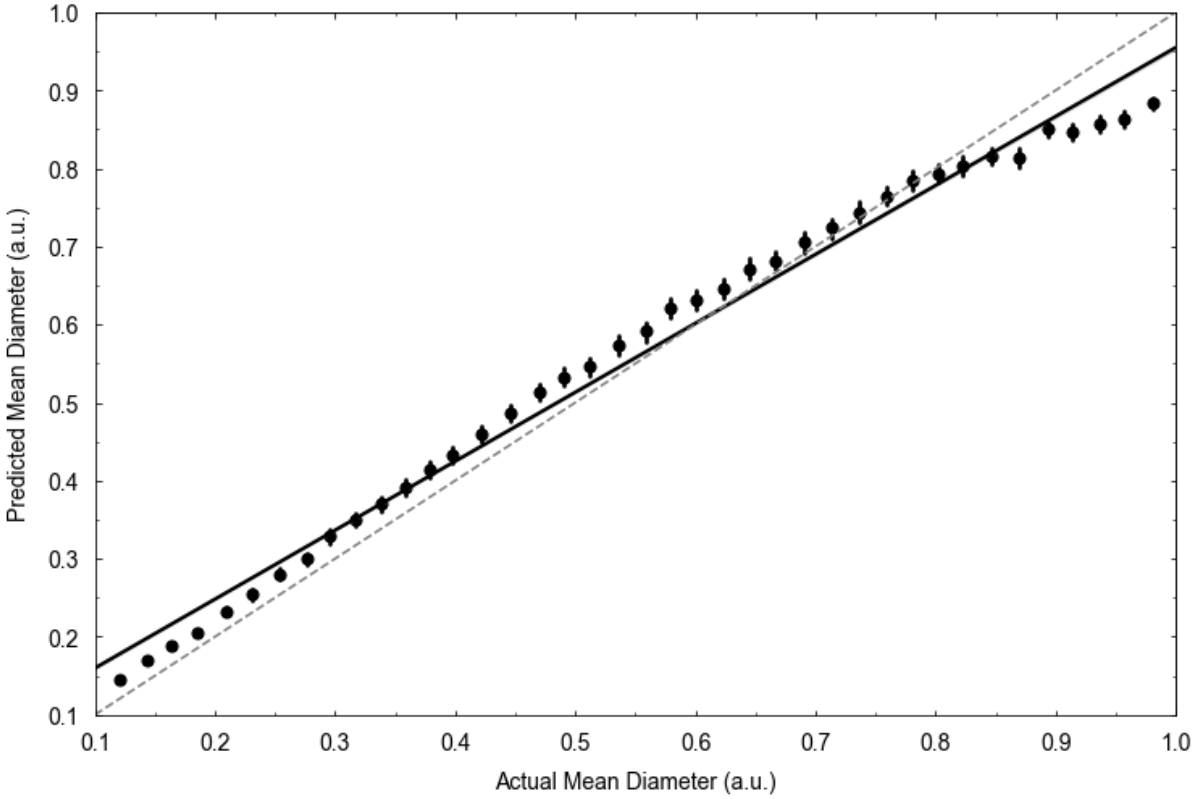

Figure 25. The SVM-predicted mean diameter D plotted against the actual mean diameter. Each dot is a bin constituted from 40 individual measurements. The solid black line is the best fit linear prediction of the SVM regressor. The dashed line represents the ideal output if the model estimation for all the diameters was equal to the actual diameter of each sample.

We binned our prediction in 40 bins and each dot in figure 25 represents the center of this bin. The error bars are the 95% confidence interval area around the center of each bin. This is like the process followed for the NN.

Figure 26 respectively shows the predicted standard deviation of our SVM model based on the k-space signal. The same binning process was applied. As mentioned in section 4.2.1 in order to be able to easily compare the performance and output of our model for both the mean diameter D and the standard deviation σ, we rescaled the region of values for both parameters so that they both cover a range of 0-1au of distance. This also allow direct comparison of their MAE and MSE values.

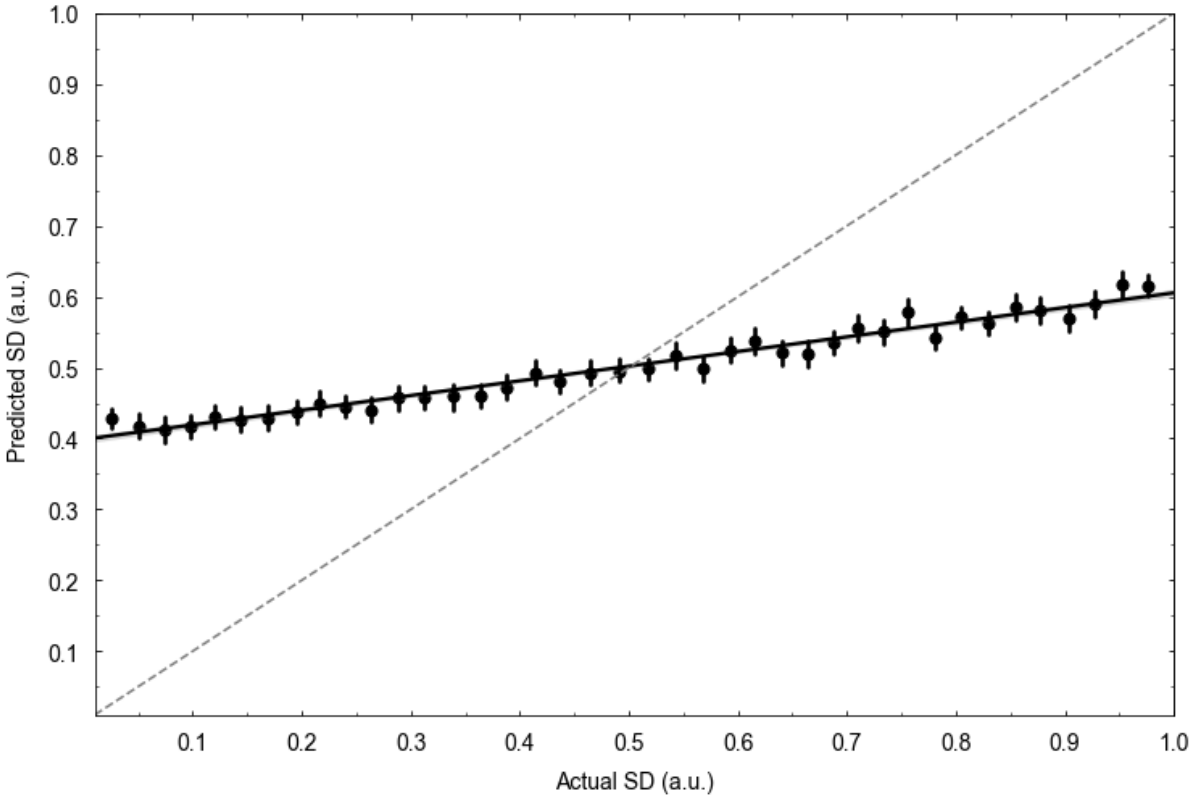

Figure 26. The SVM-predicted standard deviation σ plotted against the actual standard deviation σ. Each dot is a bin constituted from 40 individual measurements. The solid black line is the best fit linear prediction of the SVM regressor. The dashed line represents the ideal output if the model estimation for all the σ was equal to the actual σ value for each sample.

4.3.2.2. Discussion

From the results presented in section 4.3.2.1 we can see that the SVM, like the ANN, performs well on the task of predicted the mean sphere size of the distribution present

in the sample under analysis. Despite some small fluctuations throughout the range of values, the diagonal line of expected diameters follows the model predictions very closely. Moreover, the variance of the predicted values for each bin of sizes as they are represented by each dot in figure 25 is very low. This good performance is also captured in the values of the MAE and MSE for the mean diameter as shown in table 8.

Regarding the predictions of the standard deviation, we observe the same characteristics as for the ANN. Specifically the model produces an output that tends to create a constant output around 0.5 au. This is the same agnostic approach that the ANN also displayed and it was discussed in detail in section 4.2.2.

4.3.3. Sensitivity Analysis

After identifying and testing the best performing SVM model we performed a sensitivity analysis to gain a deeper insight into the model performance and how this could be affected by various parameters.

4.3.3.1. Parameters investigated

For this sensitivity test we monitored how the MAE, the MSE, the fitting (training) time and the prediction time of our model changed based on:

- The number of simulations used for training the model
- The ANN structure complexity (number of layers and number of nodes)
- The noise level present in the signal

We also investigated the effect of the standard deviation on the accuracy of the predicted mean diameter. This is the same analysis that we performed in section 4.2.3.

4.3.3.2. Results

<u>Number of simulations</u>

As seen in figure 27 the number of experiments used to train the SVM is following the law of diminishing returns where initially the MSE and MAE are decreasing rapidly as the number of experiments is growing. However, after ~10,000 experiments the rate of improvement slows down and reaches a plateau. After this point there is close to zero

reduction in the error. At the same time the time required to train the SVM increases rapidly, in a near-exponential rate, with no gain in performance after ~10,000 experiments.

There is a difference here between the SVM and the NN Model. As we see in figure 20 presented in section 4.2.3.2 the required fitting time grows at much faster rate with the number of experiments for the SVM compared to the NN. The SVM follows an exponential rate whereas the NN a logarithmic rate.

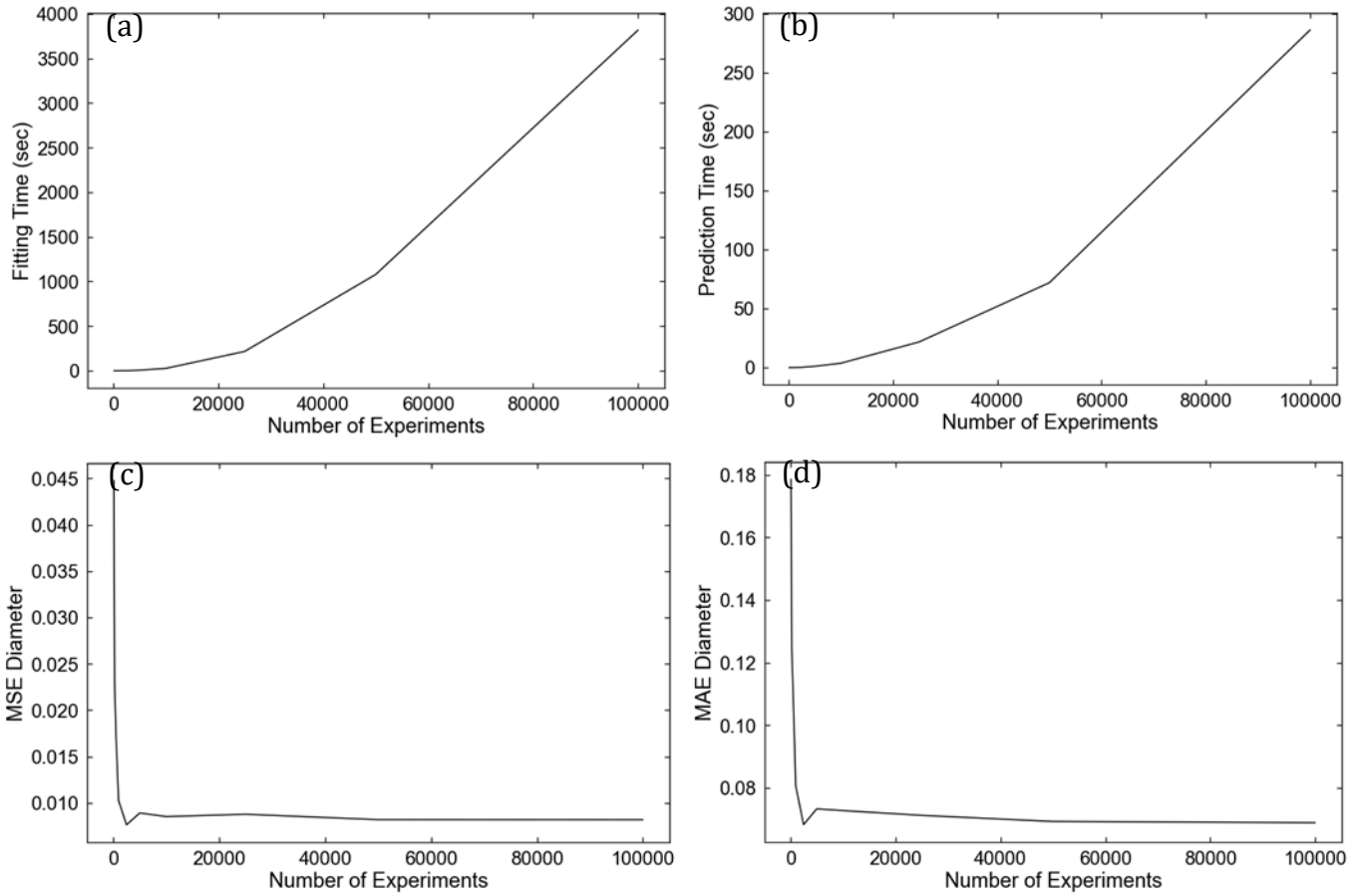

Figure 27. Effect of the number of experiments used during training on the (a) Fitting time, (b) Prediction time, (c) MSE and (d) MAE of the SVM model.

Noise level

The set of graphs in figure 28 shows the effect of noise on the performance of the SVM. We discussed the effects of noise in the analysis of k-space signal in detail in section 4.2.3.2.

The MSE and MAE increase with a logarithmic rate and plateau at about 10% NSR. This A lower value compared to the ANN. This might imply that the ANN performs better with noisy data.

Furthermore, unlike the ANN the fitting time and prediction time also increase as the noise level increases. This is an expected behavior, and it implies that the SVM does not ignore noisy data like the ANN did for lower values of noise. The performance of the SVM decreases continuously as the noise level increases following a logarithmic pattern.

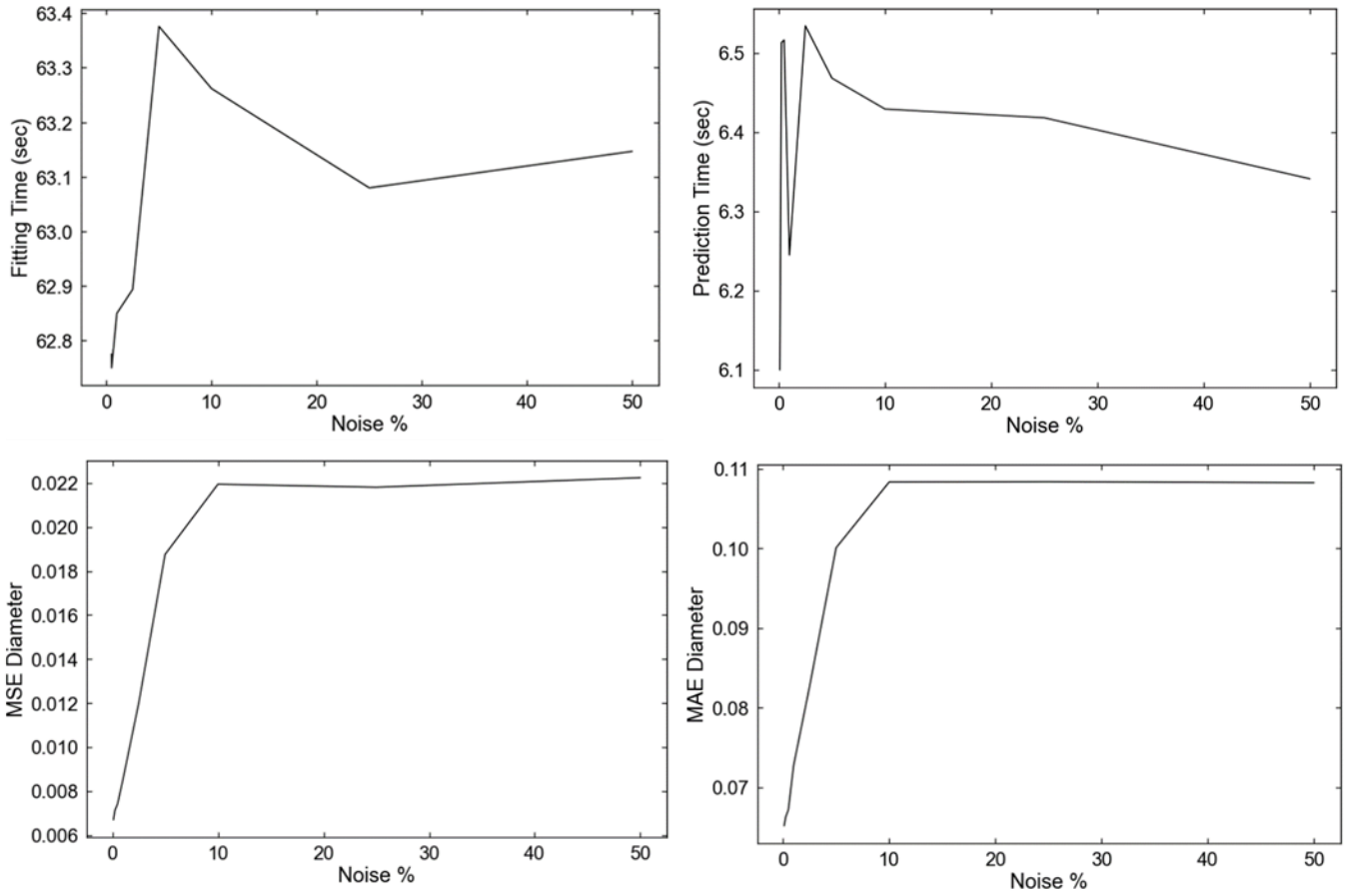

Figure 28. Effect of the noise level in the data used during training on the (a) Fitting time, (b)Prediction time, (c) MSE and (d) MAE of the SVM model.

Mean diameter-standard deviation correlation

In figure 24 we see the effect of the standard deviation on the estimation of the mean diameter for the SVM. We explained in section 4.2.3 how the standard deviation σ can affects the estimation of the mean diameter. Like the NN, we split the data based on the standard deviation in 4 parts with standard deviation i) $0 < \sigma \leq 0.4$ au, ii) $0.4 < \sigma \leq 0.6$ au, iii) $0.6 < \sigma \leq 0.8$ au and iv) $0.8 < \sigma \leq 1$ au. The SVM is also trying to find a middle ground

between the samples with different standard deviation ranges. Like in the case of the NN model the range $0 < \sigma \leq 0.4$ au seems to produce a small underestimation of the mean diameter, the $0.8 < \sigma \leq 1$ au is overestimated and the two graphs in the middle ranges are more accurately estimated.

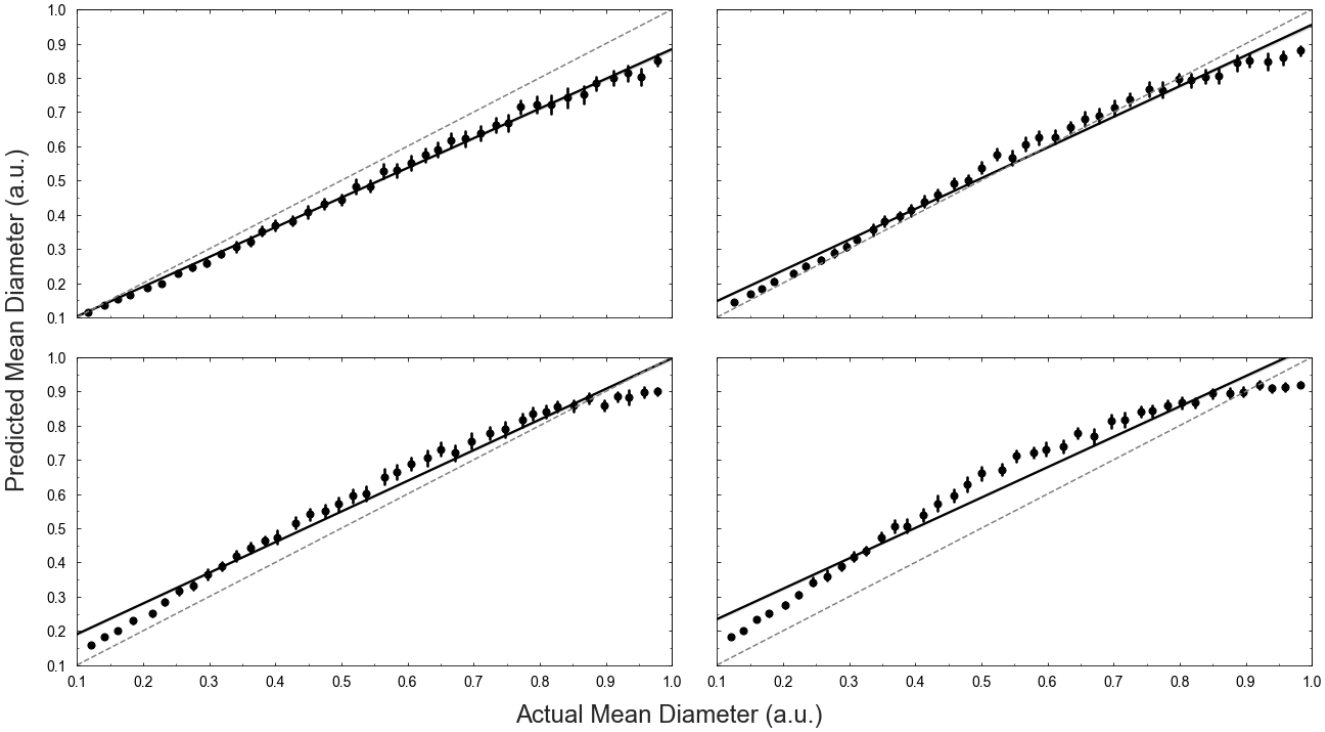

Figure 29. . The SVM-predicted mean diameter D plotted against the actual mean diameter. Each dot is a bin constituted from 40 individual measurements. The solid black line is the best fit linear prediction of the SVM regressor. The dashed line represents the ideal output if the model estimation for all the diameters was equal to the actual diameter of each sample. Each plot corresponds to samples with different range of σ values (a) $0 < \sigma \leq 0.4$, (b) $0.4 < \sigma \leq 0.6$ au , (c) $0.6 < \sigma \leq 0.8$ au and (d) $0.8 < \sigma \leq 1$ au

A model ensemble approach could also further improve the performance of the SVM. Instead of having a single model trained on all the possible standard deviations we could instead train many SVM models specialized in different ranges of standard deviation. This was again discussed in detail in section 4.2.3.

4.4.Comparing the Bayesian, NNs and SVMs models

4.4.1. NN vs SVMs

We have already discussed the most important characteristics of the ANN and the SVM model in section 4.3. Both models perform equally well and the results they produce on the test data set are almost identical. Moreover, their behavior during the testing and sensitivity analysis phases was similar (see sections 4.2.3 and 4.3.3). Nevertheless, some differences have been observed and can be summarized as follows:

- The ANN seem to perform better on noisy data than the SVM model
- The SVM requires orders of magnitude less computational time during training to achieve results equivalent to these of the NN (15 secs for SVM compared to over 1 hour for the NN)
- Despite requiring much less time during training the SVM requires 270 sec to produce a result on a test set. In contrast the ANN requires only 1.7 sec. This is a significant advantage for the NN model for time sensitive applications

Each approach has therefore some advantages that make it the best fit for specific applications.

4.4.2. NN and SVMs vs the Bayesian method

It is not easy to compare the performance of the ANN on predicted the mean diameter with previous attempts, like the Bayesian approach found in the literature (Holland et al., 2012; Ross et al., 2012a, 2012b). The reason is that in the Bayesian approach a classification approach was used rather than a regression which we present here. Nevertheless, for comparison reasons, figure 30 includes the predicted mean diameters and the standard deviations of sphere size distributions when using the Bayesian approach (Holland et al., 2012; Ross et al., 2012a), as they are reported in the literature.

We can see that the ANN and SVM developed in this thesis, both seem to produce a significantly better prediction for the mean sphere size (diameter or radius). The predictions of our models are not only more accurate but also more precise since the 95%

confidence interval are significantly smaller than those reported in the literature for the Bayesian method and presented in figure 30.

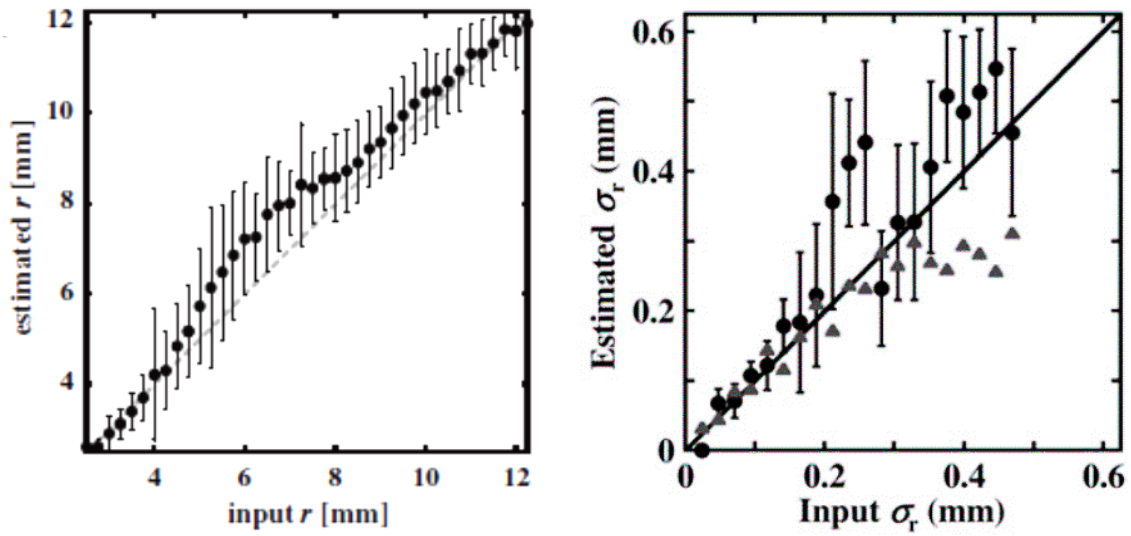

Figure 30. (a) Bayesian-predicted mean radius against real radius (b) Bayesian-predicted standard deviation against real standard deviation of samples with Gaussian sphere size distribution as reported in the literature.

It has to be noted however that the range of values for which the prediction of the mean diameter (or radius) are reported for the Bayesian method are the regions where the method *'exhibits it best performance'* [sic] (Holland et al., 2011b; Ross et al., 2012a) This is noted by the authors in all relevant papers and it is highlighted that a good understanding of the sample under analysis is needed before using the method so that the scientist can pre-configure the Bayesian methods parameters such that the predicted values can fall within this range of optimal performance. We have no such step or requirement in our method. If we were however to choose the region with the highest accuracy from our predictions, this would be around the center of the value range in figures 18 and 25. We would in that case also see that an even better performance is achieved there for the ANN and SVM compared to the Bayesian method.

On the other hand, as we already discussed in section 4.2.2 and 4.3.2 that the ANN and the SVM seems to adopt an agnostic approach regarding the value of the standard deviation. We do not observe this behavior on the results from the Bayesian method.

These could be because of two reasons:

- The region for which the results for the standard deviation are presented is also selected so that they achieve the higher accuracy. Again, we could achieve such a behavior by selected the region around the center of the figure 18 and 25
- The estimation of the sphere size and the standard deviation was formulated as a classification problem in the Bayesian approach. In that case a specific set of predefined values for the standard deviation was given to the model. This is very different than the regression approach that we take here. Specifically in the Bayesian classification approach a high penalty would occur if all values where misclassified and moved to the class around the center of the value range.

However, with a more careful analysis of the results from the Bayesian approach a similar behavior to our models might in fact be observed. This behavior has not been analyzed or discussed before in the literature (Holland et al., 2011b; Mitchell et al., 2013; Ross et al., 2012a; Wu et al., 2014). In figure 30 the Bayesian model exhibits a very low confidence and high 95% confidence intervals for the predictions of the standard deviation. This could be an equivalent behavior to the ANN-SVM models, simply expressed in a different way. The Bayesian model might in fact also adopt an agnostic approach not assigning all sample to the class near the center of the value range for the standard deviation but by near-randomly assigning classes to each sample. In other words, it would try to predict the correct class for the standard deviation for each sample but with very low confidence. It is very likely that the wrong class is assigned to each sample therefore *on average* the differentiation of standard deviations is very low. This is the same result that the ANN and SVM provide but with a different mechanism.

The previous analysis is presented here for the first time, and it is important to note that the development of the ANN and SVM models provided us with new insights for the behavior of a different model (the Bayesian model).

Another parameter that is of importance when comparing the three models with each other is the time required to develop and train each method as well a the time required to produce a prediction. In this comparison the SVM seems to be the method that offers the fastest training time as it only requires a few second to achieve a performance that is

better than the Bayesian method and comparable to the NN. The slowest method is the Bayesian approach since it requires the estimation of the Rayleigh parameters for each point in k-space for each possible sphere size distribution used in the classification task. The NN are also time consuming and depending on the model complexity (number of layers and nodes) training could require a time of several minutes up to over an hour.

The comparison above covers the time required for training each model. After the training step however, the model is deployed and is used to predict the sphere size distribution of new samples. This is the most time-critical step for many analytical processes (Bishop, 2006)(Hannes Hapke, 2020; Weber, 2020). In this comparison the NN is the best performing method since it requires just over a second to produce an output and this time can be reduced easily by improving the available computational resources. The Bayesian method offers a slightly higher prediction time (a few seconds) and it is also bound to the computational resources available. Finally, the SVM model requires the largest amount of time to produce a result, and the time was in our case a few minutes, significantly lower than both the other methods.

5. Conclusions

In this thesis we developed and analyzed two new methods that can be used for studying the properties of various materials.

More specifically there is a need for new, faster methods for studying and characterizing the properties of materials both for scientific as well as for industrial purposes. We focused on a specific type of materials called dispersions. These materials are comprised of at least two distinct phases where one secondary phase is dispersed in near spherical particles within a continuous phase. Dispersions are found in a very wide range of applications, and it is often necessary to determine the sphere size distribution of the dispersed phase within the continuous phase. The information about the sphere size distribution can be related to many structural and functional properties of the materials (A.J.Meagher et al., n.d.; P.R.Garret, 1993)(Chung & McClements, 2014; Wilson, 1989).

There are currently various traditional methods for analyzing these materials but there is still a need for a very fast, reliable, and non-invasive method that can be used both in a laboratory as well as a production environment. Recently a new method was proposed which utilizes k-space signal acquired from either a NMRI or an X-Ray machine and combined with a Bayesian probabilistic model it can provide a very fast estimation for the sphere size distribution (Holland et al., 2012)(K. Ziovas et al., 2016).

Based on this Bayesian method and taking advantage of the progress made in recent years in the fields of NN and SVM we proposed a new expansion and improvement of this k-space signal analysis approach. We used simulated k-space data of materials which have a Gaussian sphere size distribution for their dispersed phase. Our goal was to be able to predict with a good accuracy the mean sphere diameter and the standard deviation for each sample based only on a 1D set of k-space data. Only 128 k-space data points were used for each sample. We developed, trained, and tested a NN and a SVM regression model that can take a set of k-space data as input and produce a very good estimation for the sphere size distribution in a sample of a material. The two methods that we tested formulated the problem as a regression task. This contrasts with the previously proposed Bayesian method which approached it as a classification problem.

Moreover, an extensive meta-parameter optimization was performed, and we managed to produce two models that perform well at predicting the mean sphere diameter from a material sample. The estimation for the standard deviation of the sphere size distribution within our samples was not estimated with equally high accuracy, similarly to the results presented in the past from the Bayesian approach. The combined results of our methods with the findings from the Bayesian approach show that this result stems from the intrinsic nature of the standard deviation as a variable that measure the uncertainty over the mean sphere size.

In summary both the new approaches appear to offer significant improvements over the existing methods for a wider variety of parameters such as i) wider range of sphere size distribution ii) faster development, testing and deployment times iii) better performance for noisy data. Comparing the two new methods with each other we can briefly say that the SVM offers good performance with lower computational requirements during the model training phase. On the other hand, the NN model seems to perform better with noisy data and more importantly it has a two orders of magnitude faster prediction time. This fast prediction time can be extremely important for real-time analysis applications where a method is required to produce reliable estimation for the structure of a sample within a sub-second time window (Stevenson et al., 2006)(P.R.Garret, 1993).

The investigation presented here can be seen as only a first step in the potential application of new ML methods such as NN and SVM on the analysis of k-space signal. More work can be done in many directions which could either improve the performance of the proposed methods or find new applications where ML models can be combined with k-space data to provide information about various other properties of materials. Some potential extensions of our work are presented in the "*Future Work*" chapter.

6. Future Work

In the work presented in this thesis we extended some ideas that have been recently developed. These ideas the usage of 1D k-space data acquired from a sample of a material in combination with a ML method to produce an estimation about the properties of the given material. We extended the previous work by:

- Investigating two ML methods that have grown rapidly in popularity the past decade. The ANN and SVM have become the center of great amount of groundbreaking scientific work which has provided great advances in many fields. Through our work we have demonstrated that they offer very promising results in the field analytical techniques for material science.
- Gaining better understanding about the previously proposed methods, their advantages, and limitations and how they can be improved with the use of regression models.

The development and investigation work performed for this thesis however demonstrated that there are many more potential avenues to be explored.

A path of potential future work could include an attempt to better understand the behavior and characteristics of the NN and SVM models trained on 1D k-space signal of materials with a dispersed phase. A better understanding could be gained on the interaction of the features of the k-space signal and how these features affect the training and performance of each ML model. For example, it is often the case that various artifacts exist in real k-space signal acquired with NMRI or X-Ray mCT (Callaghan, 1993)(Baruchel et al., 2000; Ketcham, 2001). It is important to understand how these artifacts and various other signal characteristics affect the proposed methods.

Regarding possible improvements of the proposed NN and SVM models, topics to study could include:

- Investigation of the performance of the models on materials with sphere size distributions different than Gaussian. For example, log-normal and binomial distributions could be studied.

- Creation of an ensemble of models where each model could be specialized in different set of material properties. like the distribution type, standard deviation and any other structural parameter of the sample. This would provide a much more accurate and powerful tool with the main downside being the longer computational time required for training all the individual models.
- Training of the models with real experimental data acquired with NMRI or Xray mCT. This would allow the models to learn from the features that exist in the real material samples and to be fine tuned specifically for the material properties of interest.

Finally, the ML models used here to predict the sphere size distribution of materials based on their k-space signal could also be used to predict many other properties of various materials. In particular any material property that is proven to have an effect on the k-space or spectral signal, acquired with an Xray, MRI or NMR machine, could be predicted with the use of a ML model. For example, the chemical composition of a material can be monitored with NMR spectroscopy. In many cases multiple molecules produce very complicated spectral signatures which could also be affected by various artifacts. A properly trained ML model should be able to produce good estimations for the composition of the materials much faster than traditional methods.

7. References

A.J.Meagher, Mukherjee, M., Weaire, D., Hutzler, S., Banhert, J., & F. Garcia-Moreno. (n.d.). *Analysis of the internal structure of a monodisperse liquid foam by X-ray tomography*.

Abraham, R. J., Fisher, J., & Loftus, P. (1988). *Introduction to NMR Spectroscopy*. Wiley.

Agarwal, A. (2019). *Introduction to Artificial Neural Networks - Explanation, Formulation & Derivation*.

Baruchel, J., Buffiere, J.-I., Maire, E., Merle, P., & Peix, G. (2000). *X-ray Tomography in Material Science*. Hermes Science Publications.

Battineni, G., Chintalapudi, N., & Amenta, F. (2019). Machine learning in medicine: Performance calculation of dementia prediction by support vector machines (SVM). *Informatics in Medicine Unlocked*, *16*. https://doi.org/10.1016/j.imu.2019.100200

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blonk, J. C. G., & Aalst, H. Van. (1993). Confocal scanning light microscopy in food research. *Food Research International*, *26*, 297–311.

Bracewell, R. (1999). *The Fourier Transform & Its Applications*. McGraw-Hill Science.

Brauer, J. (2018). *Introduction to Deep Learning_ with Complexe Python and TensorFlow Examples*.

Callaghan, P. T. (1991). *Principles of Nuclear Magnetic Resonance Microscopy*. Oxford University Press.

Callaghan, P. T. (1993). *Principles of Nuclear Magnetic Resonance Microscopy* (Illustrate). Oxford science publications.

Callaghan, P. T. (2011). *Translational Dynamics and Magnetic Resonance*. Oxford University Press.

Campbell, G. M., & Mougeot, E. (1999). Creation and characterisation of aerated food products. *Trends in Food Science & Technology*, *10*, 283–296.

Capponi, M., Flister, A., Hasler, R., Oschatz, C., Robert, G., Robinson, T., Stakelbeck, H. P., Tschudin, P., & Vierling, J. P. (1982). Foam Technology in Textile Processing. *Review of Progress in Coloration and Related Topics*, *12*, 48–57.

Carlson, W. D., Rowe, T., Ketcham, R. A., & Colbert, M. W. (2003). Applications of high-resolution X-ray computed tomography in petrology, meteoritics and palaeontology. *Geological Society, London, Special Publications* , *215*(1), 7–22. https://doi.org/10.1144/GSL.SP.2003.215.01.02

Chollet, F. (2018). Deep Learning with Python, Manning. In *Manning*.

Chung, C., & McClements, D. J. (2014). Structure–function relationships in food emulsions: Improving food quality and sensory perception. *Food Structure*, *1*(2), 106–126. https://doi.org/10.1016/j.foostr.2013.11.002

Cierniak, R. (2011). *X-Ray Computed Tomography in Biomedical Engineering*. Springer-Verlag.

Coupland, J. N., & Mcclements, D. J. (2001). Droplet size determination in food emulsions : comparison of ultrasonic and light scattering methods. *Journal of Food Engineering*, *50*, 117–120.

Deng, L., & Yu, D. (2013). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, *7*(3–4), 197–387. https://doi.org/10.1561/2000000039

Durian, D. J., Weitz, D., & Pine, D. J. (1991). Multiple light-scattering probes of foam structure and dynamics. *Science (New York, N.Y.)*, *252*(5006), 686–688. https://doi.org/10.1126/science.252.5006.686

F., M. (2009). Investigations of food colloids by NMR and MRI. *Current Opinion in Colloid & Interface Science*, *14*(3), 203–211. http://www.sciencedirect.com/science/article/pii/S1359029408000952

Friedman, J., Hastie, T., & Tibshirani, R. (2009). *The Elements of Statistical Learning Data Mining, Interface and Prediction Preface to the Second Edition*. 809.

Géron, A. (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media.

Green, A. J., Littlejohn, K., Hooley, P., & Cox, P. W. (2013). Formation and stability of food foams and aerated emulsions: Hydrophobins as novel functional ingredients. *Current Opinion in Colloid & Interface Science*, *18*(4), 292–301. https://doi.org/10.1016/j.cocis.2013.04.008

Gulli, A., & Pal, S. (2017). *Deep Learning with Keras*. Packt Publishing.

Håkansson, B., Pons, R., & Söderman, O. (1998). *Diffraction-like effects in a highly concentrated w/o emulsion: a PFG NMR study*. *16*(5–6), 643–646.

Hannes Hapke, C. N. (2020). *Building Machine Learning Pipelines*. O'Reilly Media.

Hastie, T., Tibshirani, R., & Friedman, J. (2016). *The Elements of Statistical Learning*. Springer.

Höhler, R., Cohen-Addad, S., & Durian, D. J. (2014). Multiple light scattering as a probe of foams and emulsions. *Current Opinion in Colloid & Interface Science*, *19*(3), 242–252. https://doi.org/10.1016/j.cocis.2014.04.005

Holland, D. J., Blake, a., Tayler, a. B., Sederman, a. J., & Gladden, L. F. (2012). Bubble size measurement using Bayesian magnetic resonance. *Chemical Engineering Science*, *84*, 735–745. https://doi.org/10.1016/j.ces.2012.08.024

Holland, D. J., Blake, A., Tayler, A. B., Sederman, A. J., & Gladden, L. F. (2011a). A Bayesian approach to characterising multi-phase flows using magnetic resonance: application to bubble flows. *Journal of Magnetic Resonance (San Diego, Calif. : 1997)*, *209*(1), 83–87. https://doi.org/10.1016/j.jmr.2010.12.003

Holland, D. J., Blake, A., Tayler, A. B., Sederman, A. J., & Gladden, L. F. (2011b). A Bayesian approach to characterising multi-phase flows using magnetic resonance: application to bubble flows. *Journal of Magnetic Resonance (San Diego, Calif. : 1997)*, *209*(1), 83–87. https://doi.org/10.1016/j.jmr.2010.12.003

Hou, X., Wang, G., Su, G., Wang, X., & Nie, S. (2019). Rapid identification of edible oil species using supervised support vector machine based on low-field nuclear magnetic resonance relaxation features. *Food Chemistry*, *280*(December 2018), 139–145. https://doi.org/10.1016/j.foodchem.2018.12.031

Hsiung, S.-K., Chen, C.-T., & Lee, G.-B. (2006). Micro-droplet formation utilizing microfluidic flow focusing and controllable moving-wall chopping techniques. *Journal of Micromechanics and Microengineering*, *16*(11), 2403–2410. https://doi.org/10.1088/0960-1317/16/11/022

Ian, G., Yoshua, B., & Courville, A. (2016). Deep learning adaptive computation and machine learning. *Prmu*, 1–801. www.deeplearningbook.org

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introdution to Statistical*

Learning. Springer-Verlag.

Johns, M. L., & Gladden, L. F. (2002). Sizing of emulsion droplets under flow using flow-compensating NMR-PFG techniques. *Journal of Magnetic Resonance (San Diego, Calif. : 1997)*, *154*(1), 142–145. https://doi.org/10.1006/jmre.2001.2469

Johns, M. L., & Hollingsworth, K. G. (2007). Characterisation of emulsion systems using NMR and MRI. *Progress in Nuclear Magnetic Resonance Spectroscopy*, *50*(2–3), 51–70. https://doi.org/10.1016/j.pnmrs.2006.11.001

Ketcham, R. (2001). *X-ray Computed Tomography (CT)*. University of Texas at Austin.

Kobayashi, I., Uemura, K., & Nakajima, M. (2007). Formulation of monodisperse emulsions using submicron-channel arrays. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, *296*(1–3), 285–289. https://doi.org/10.1016/j.colsurfa.2006.09.015

Kostoglou, M., Varka, E.-M., Kalogianni, E. P., & Karapantsios, T. D. (2010). Evolution of volume fractions and droplet sizes by analysis of electrical conductance curves during destabilization of oil-in-water emulsions. *Journal of Colloid and Interface Science*, *349*(1), 408–416. https://doi.org/10.1016/j.jcis.2010.05.032

Laverse, J., Frisullo, P., Conte, A., & Nobile, M. A. Del. (2012). X-Ray Microtomography for Food Quality Analysis. In *Food Industrial Processes - Methods and Equipment* (pp. 339–362).

Le, W. T., Maleki, F., Romero, F. P., Forghani, R., & Kadoury, S. (2020). Overview of Machine Learning: Part 2: Deep Learning for Medical Image Analysis. *Neuroimaging Clinics of North America*, *30*(4), 417–431. https://doi.org/10.1016/j.nic.2020.06.003

Lee, W. (2019). Python® Machine Learning. In *Python® Machine Learning*. https://doi.org/10.1002/9781119557500

Levitt, M. H. (2007). *Spin Dynamics: Basics of Nuclear Magnetic Resonance* (2nd ed.). Wiley

Maleki, F., Ovens, K., Najafian, K., Forghani, B., Reinhold, C., & Forghani, R. (2020). Overview of Machine Learning Part 1: Fundamentals and Classic Approaches. *Neuroimaging Clinics of North America*, *30*(4), e17–e32. https://doi.org/10.1016/j.nic.2020.08.007

Mariette, F. (2009). Investigations of food colloids by NMR and MRI. *Current Opinion in*

Colloid & Interface Science, *14*(3), 203–211. https://doi.org/10.1016/j.cocis.2008.10.006

Mariette, Francois, Collewet, G., Davenel, A., Lucas, T., & Musse, M. (2012). Quantitative MRI in Food Science & Food Engineering. *EMagRes*, *Figure 1*, 1–8. https://doi.org/10.1002/9780470034590.emrstm1272

Meagher, A. J., Mukherjee, M., Weaire, D., Hutzler, S., Banhart, J., & Garcia-Moreno, F. (2011). Analysis of the internal structure of monodisperse liquid foams by X-ray tomography. *Soft Matter*, *7*(21), 9881. https://doi.org/10.1039/c1sm05495c

Messaoud, S., Bradai, A., Bukhari, S. H. R., Quang, P. T. A., Ahmed, O. Ben, & Atri, M. (2020). A survey on machine learning in Internet of Things: Algorithms, strategies, and applications. *Internet of Things (Netherlands)*, *12*, 100314. https://doi.org/10.1016/j.iot.2020.100314

Mitchell, J., Chandrasekera, T. C., Holland, D. J., Gladden, L. F., & Fordham, E. J. (2013). Magnetic resonance imaging in laboratory petrophysical core analysis. *Physics Reports*, *526*(3), 165–225. https://doi.org/10.1016/j.physrep.2013.01.003

Müller, A., & Guido, S. (2018). Introduction to Machine Learning with Python: A Guide for Data Scientists 1st Edition. O'Reilly Media. In *Journal of Chemical Information and Modeling* (Vol. 53, Issue 9).

Otchere, D. A., Arbi Ganat, T. O., Gholami, R., & Ridha, S. (2021). Application of supervised machine learning paradigms in the prediction of petroleum reservoir properties: Comparative analysis of ANN and SVM models. *Journal of Petroleum Science and Engineering*, *200*, 108182. https://doi.org/10.1016/j.petrol.2020.108182

P.R.Garret. (1993). Recent developments in the understanding of foam generation and stability. *Chemical Engineering Science*, *48*(2), 367–392.

Pisner, D. A., & Schnyer, D. M. (2019). Support vector machine. In *Machine Learning: Methods and Applications to Brain Disorders*. Elsevier Inc. https://doi.org/10.1016/B978-0-12-815739-8.00006-7

Rodríguez-García, J., Salvador, A., & Hernando, I. (2013). Replacing Fat and Sugar with Inulin in Cakes: Bubble Size Distribution, Physical and Sensory Properties. *Food and Bioprocess Technology*, *7*(4), 964–974. https://doi.org/10.1007/s11947-013-1066-

z

Ross, J. G., Holland, D. J., Blake, A., Sederman, A. J., & Gladden, L. F. (2012a). Extending the use of Earth's Field NMR using Bayesian methodology: Application to particle sizing. *Journal of Magnetic Resonance*, *222*, 44–52. https://doi.org/10.1016/j.jmr.2012.05.023

Ross, J. G., Holland, D. J., Blake, A., Sederman, A. J., & Gladden, L. F. (2012b). Extending the use of Earth's Field NMR using Bayesian methodology: application to particle sizing. *Journal of Magnetic Resonance (San Diego, Calif.: 1997)*, *222*, 44–52. https://doi.org/10.1016/j.jmr.2012.05.023

Rozenberg, G., Back, T., & Kok, J. N. (2012). Handbook of Natural Computing. *Handbook of Natural Computing*, *1–4*, 1–2051. https://doi.org/10.1007/978-3-540-92910-9

Sadoc, J. F., & N.Rivier (Eds.). (1999). *Foams and Emulsions*. Kluwer Academic Publishers.

Sadoc, J. F., & Rivier, N. (1997). Foams and Emulsions. *Proceedings of the NATO Advanced Study Institute on Foams and Emulsions, Emulsions and Cellular Materials*.

Schramm, L. L. (2014). *Emulsions, foams, suspensions, and aerosols : microscience and applications* (2nd ed.). Wiley-VCH.

Skurtys, O., & Aguilera, J. M. (2007). Applications of Microfluidic Devices in Food Engineering. *Food Biophysics*, *3*(1), 1–15. https://doi.org/10.1007/s11483-007-9043-6

Stevens, E., Antiga, L., & Viehmann, T. (2020). *Deep Learning with PyTorch*. Manning.

Stevenson, P., Mantle, M. D., Sederman, A. J., & Gladden, L. F. (2006). Quantitative Measurements of Liquid Holdup and Drainage in Foam Using NMRI. *AIChE Journal*, *53*(2), 290–296. https://doi.org/10.1002/aic

Trater, A. M., & Alavi, S. (2005). Use of non-invasive X-ray microtomography for characterizing microstructure of extruded biopolymer foams. *Food Research International*, *38*, 709–719. https://doi.org/10.1016/j.foodres.2005.01.006

Weaire, D., & Hutzler, S. (1999). *The Physics of Foams*. Oxford University Press.

Weber, B. (2020). *Data Science in Production: Building Scalable Model Pipelines with Python*.

Wilson, A. J. (1989). *Foams: Physics, Chemistry and Structure*.

Wu, Y., Holland, D. J., Mantle, M. D., Wilson, A. G., Nowozin, S., Blake, A., & Gladden, L. F. (2014). A Bayesian Method to Quantifying Chemical Composition Using NMR : Application to Porous Media Systems. *22nd European Signal Processing Conference*, 2515–2519.

Ziovas, K., Sederman, A. J., Gehin-Delval, C., Gunes, D. Z., Hughes, E., & Mantle, M. D. (2016). Rapid sphere sizing using a Bayesian analysis of reciprocal space imaging data. *Journal of Colloid and Interface Science*, *462*, 110–122. https://doi.org/10.1016/j.jcis.2015.09.066

Ziovas, Kostas. (2021). *MSc Thesis Porject*. https://github.com/kziovas/msc-thesis-project-informatics

Zuniga, R., & Aguilera, J. (2008). Aerated food gels: fabrication and potential applications. *Trends in Food Science & Technology*, *19*(4), 176–187. https://doi.org/10.1016/j.tifs.2007.11.012