Norwegian University
of Life Sciences

Master's Thesis 2024 30 ECTS
Faculty of Science and Technology

A Parameter Estimation Approach in Simulated Neural Data Using Metamodelling Approaches

Raúl Andreas Sanjinés Morató
Data Science (M.Sc.)

Acknowledgments

I extend my wholehearted gratitude to my family, my mom Tania, my grandmother Esthercita, my uncle Carlos who unwaveringly supported me all my life. I want to thank my supervisors, Fadi and Eirik; without their patience and support, this submission would not have been possible. Lastly, I want to express my gratitude to Alexandra, who stood by me in the ups and downs for the duration of the whole degree and more.

Raúl Andreas Sanjinés Morató
Ås, October 2023

Abstract

Data from dynamic stochastic systems is highly complex, making it hard to analyze. Neural network simulations based on such systems have parameters with intricate relationships with their outcomes necessitating deep parameter space exploration and sensitivity; these processes can be computationally demanding and hard to interpret. In this case, the objective was to estimate parameter labels; approximate parameter space from simulated neural activity in the shape of pair of spike trains. This study uses a metamodelling approach to compare two different sampling approaches when applying the Random Forest algorithm. In the sampling, normal sampling, and synthetic minority oversampling technique (SMOTE) with imbalanced labeled data were compared. Results show that, when using normal sampling the classification gives low prediction accuracies on the test data close to random chance. Using random forest with imbalanced labeled data and SMOTE provided an improvement on the performance of the model across classification metrics, with an average accuracy of 80%, and precision and F1 score with a similar 80% average value. This paves the way for future research involving complex synthetic neural data while attempting to estimate a parameter space.

Contents

List of Figures

Nomenclature

HC-PLSR	Hierarchical Cluster-based Partial Least Squares Regression
LFP	Local Field Potential
MSA	Multi-Surrogate Approximation
NEST	Neural Simulation Tool
OLS	Ordinary Least Squares
PLSR	Partial Least-Squares Regression
SMOTE	Synthetic Minority Over-Sampling Technique
SVC	Support Vector Classification

Chapter 1

Introduction

1.1 Background

The brain is one of the most complex systems known. Our understanding of it and its functions has evolved through many research methods including imaging technologies like Magnetic Resonance Imaging (MRI), anatomical dissections, and computational simulations [1, 2]. New developments in the field aim to better the understanding of complex brain functions like memory, learning, and the drivers of behavior, and how the connectivity in the brain, synaptic strengths, and other aspects of it influence such functions. Research in this field is aimed not only at understanding the intricacies of the brain and its processes, but also to help develop models for simulating neurological disorders which has the potential of aiding in diagnostic tools and test hypothesis for therapeutic strategies.

Simulations are based on models and they come in various shapes and sizes, from simple mechanistic models of two neurons, to massively complex networks of populations of neurons. Tools like the Neural Simulation Tool (NEST) attempt to bridge the gap between neural simulations and real data. While the simulations attempt to achieve results that come the closest to a measurable reality, they may become too computationally expensive or too intractable to interpret. In order to be able to ascertain if a model actually approaches reality, much data is needed both from the model and from reality. This has several limitations ranging from lack of patient neurological data or lack of samples of when the model was applied (each instance of a model running is a simulation). Depending on the complexity of the model, new simulations can use up a lot of computational resources or take a lot of time. Moreover the sensitivity of the models must be taken into account [3]; as many models include or are formed by complex differential equations, small changes in initial conditions can lead to big changes in the output. There is a need for methodologies that can help infer the relationships between the outputs of a model and the parameters that originated them in an accurate and data-efficient way.

1.2 Problem Statement

As researchers continue gathering more in vivo data, generating synthetic data from models, and developing more theories on the brain, there is a need for a simplified framework that can allow the elucidation of the relationships between parameters and data. Current models can become too complex to scale and to interpret, needing approaches that can keep the depth of the analysis while simplifying the process. For this purpose, a data-intensive metamodelling approach can be applied. Metamodelling in this case refers to "model of a model", which uses the outputs of a model to make another, simpler one.

The precise estimation of network parameters from synthetic neural activity data has many challenges. The complex non-linear nature of neural models mean that small variations in initial conditions can lead to entirely different outputs, making the input-output relationship elusive. This highlights the need for a robust methodology that can help in parameter estimation.

With only having a limited amount of data, one of the challenges faced is model parameter estimation from neural activity synthetic data. To create simulations, NEST takes as input series of parameters to define networks of populations of neurons and their interaction. When exploring a specific parameter space, all other parameters are kept constant while the one under the scrutiny is varied across a defined range; a deterministic relationship between parameters and outputs is challenging to determine due to complexity of the model and the nature of it.

1.3 Research Questions

Synaptic strengths representing the connectivity of neuron populations and their efficacy of synaptic transmission between neurons play a crucial role in the behaviors and dynamics in neural network simulations. This parameter can vary significantly and, additionally, small variations in it can cause significantly different outputs. Typically these synaptic strengths are continuous variables within models; simplifying them into categorical values could potentially aid in making models more efficient and interpretable. Accurate parameter estimation is important for understanding and modelling the dynamics of neural networks; the focus of this research is to be able to accurately estimate model parameters from complex data. Specifically, the aim is to extract parameters (or parameter ranges) from spike train data generated by neuronal network simulations.

- How does the categorization of synaptic strengths into "high" and "low" impact the predictive accuracy and interpretability of machine learning models in identifying key dynamics within neural networks?
- Can a metamodelling approach improve the selection and tuning of machine learning algorithms for modelling synaptic strength dynamics in neural network simulations, leading to more efficient generalization across different network configurations?

1.4 Objectives

The main objective is to use a metamodelling approach to evaluate and compare the effectiveness of various classification algorithms for predicting parameters from spike train data derived from neuronal network simulations.

- Identify the best-performing model.
  - Use metrics in addition to accuracy, as precision and F1 score to assess the models' performance
- Provide a methodological framework for applying these techniques in computational neuroscience.

1.5 Scope and Limitations

The data scope of this thesis is centered on the data of 10.000 simulation sets. This includes the parameters (network simulation kernel), and spike trains for two populations of neurons (one excitatory and one inhibitory). All simulations were performed exclusively in the Neural Simulation Tool (NEST), not using other neural simulators for this study. No additional simulations were performed. The scope of the machine learning approaches used was centered on the comparison of Support Vector Classification (SVC) and the Random Forest algorithm; this was compared in two scenarios of classification tasks. For this purpose target data underwent two data labeling scenarios: balanced quartile-based labeling, and imbalanced quartile-based binary labeling.

Chapter 2

Theory

This chapter will introduce the main theoretical topics involved in this study. Firstly, fundamental concepts about the neuron, neuron networks, and the modelling of them will be presented. This will be followed by a short overview of machine learning; this section will focus on supervised learning for classification, with an emphasis on the Random Forests algorithm. Then, the concept of metamodelling will be introduced; this will be done from the definition of metamodelling to its current uses in computational neuroscience. The chapter ends with a section dedicated to the explanation of useful model evaluation metrics, with a focus on metrics for classification tasks.

2.1 Fundamentals of Neuronal Network Dynamics

2.1.1 Neuron Dynamics

Neurons are a basic structural and functional units of the neural system; they are designed to process and transmit information via electrical and chemical signals. The structure of a neuron consists of the cell body (soma), dentrites and axon; this structure enables an efficient flow of information between neurons and their networks. Figure 2.1 shows a schematic view of the neuron and its parts [1].

Dendrites are the neurons' receivers of signals, they gather input from other neu-

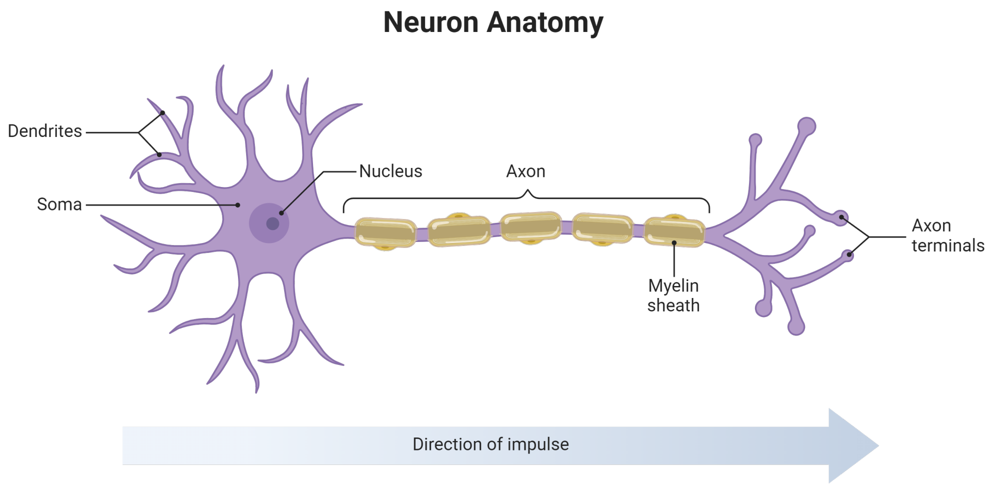

Figure 2.1: Schematic view of the main parts of a neuron, Licenced and Created with BioRender.com

rons and relay it to the soma for processing. From the soma extends the axon, responsible for transmitting the processed signal away from the cell body and toward other neurons, muscle cells, or glands. The axon is encased in a myelin sheath that enhances and preserves the signal. The main signaling between neurons happens with action potentials, these are threshold events in the shape of electrical impulses. This event has many stages as seen in figure 2.2.

The starting point of this process is the resting potential, a state where the neuron is not transmitting nor receiving any signals. In this phase, the neuron holds a negative internal charge relative to the outside of the neuron, typically around -70 milivolts (mV). When this neuron receives a stimulus from another, this balance can change increasing the internal charge; if the stimulus is strong enough to reach a threshold (usually around -55 mV), an action potential initiates. If the stimulus fails to reach the threshold, the neuron returns to the resting potential and no action potential is triggered. If the threshold is reached the action potential begins with an initial rising phase. There is a quick influx of positively charged sodium ions into the cell, changing the relative internal charge of the neuron to the outside until it reaches a peak at around +40 mV. After the peak comes a falling phase where there is an outflow of positively charged potassium ions from the neuron, followed by an undershoot phase where the membrane potential drops slightly below the resting potential, creating a temporary hyperpolarized state. This keeps the neuron from responding to signals for a short period of time, after

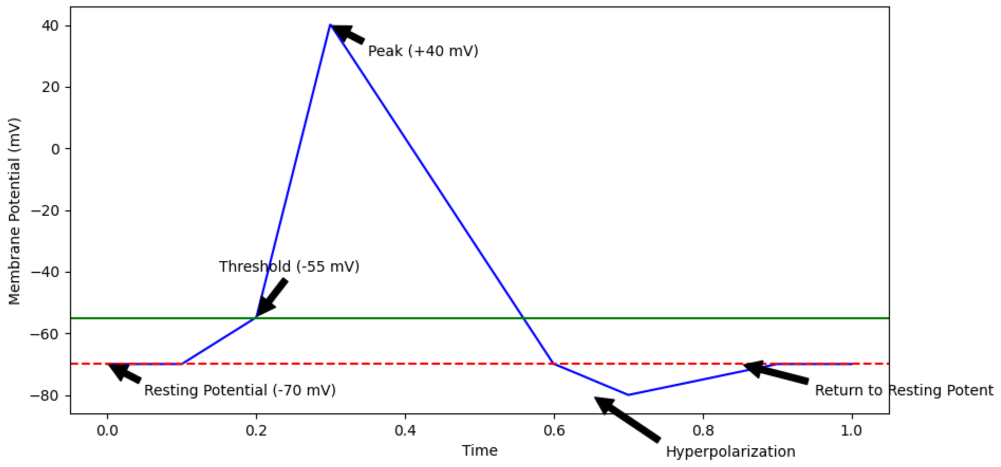

Figure 2.2: Schematic representation of an action potential

which the neuron returns to its resting potential. This entire sequence of events represents a single action potential that happens just in a few milliseconds [1] [2].

The connection between neurons happens through specialized links called synapses; when an action potential reaches the end of the axon (pre-synaptic terminal), it triggers the release of neurotransmitters. These chemical messengers bind to specific receptors on the post-synaptic neuron, causing excitation or inhibition depending on the neurotransmitter and receptor involved. Action potentials are one of the ways neurons communicate with each other, another form of communication is the local field potential (LFP) which represents synchronized, subthreshold neuron activity. The LFP measures net synaptic activity in a region, more related to neuron input, while action potentials measure neuron outputs [2].

2.1.2 Neuron Modelling

Modelling neurons and their networks help researchers to better understand how information is processed by neurons. A limitation in understanding or interpreting brain activity is its high complexity. Neuron modelling provides a framework that can help explore and test hypotheses, develop insights into brain processes, and generate predictions for empirical testing. The benefits of modelling neurons are the reduction of complexity, the integration of experimental data, and hypothesis testing and prediction [1].

There are different approaches to modelling neurons and their networks. Mechanistic models describe the functioning of brain cells and their connections. The Hodgkin-Huxley model developed in the 50s is an example of this; it explains how action potentials are generated with the movement of sodium and potassium ions across the cell membrane [4]. In contrast to mechanistic models like the Hodgkin-Huxley model, descriptive models aim to characterize the relationship between inputs and outputs of neurons or neural networks without modelling their underlying biological mechanisms. These models are usually based on statistical techniques or machine learning algorithms that capture the input-output mapping from empirical data. An example of a descriptive model is the integrate-and-fire neuron; this model describes the neuron's membrane potential in terms of the synaptic inputs and the injected current it receives [5].

As well as having models with different approaches, neural modelling can be performed at different detail levels, each having different uses and limitations. An ultra-detailed model could provide information about ion channels in the neuron, setting the scale of the measurements in the order of picometers, and could be useful to explore the mechanisms involved in this process, but will fail to provide useful information when analyzing networks of neurons. A more generalized, firerate-based model, can provide great insight into the interaction of populations of neurons, yet, may fail to characterize individual neuron activity. The level of detail chosen will depend on the research's needs, with a constant dichotomy of choosing a realistic detailed model or a less realistic but more interpretable one. [2]

Developments in computational neuroscience lead to the development of novel ways of defining networks and models to tackle specific issues. One of these network setups is the Brunel Network, which is commonly used to explore the dynamics of sparsely connected networks of excitatory and inhibitory neurons [6]. A Brunel network is made of 2 sparsely, randomly connected populations of neurons, one excitatory and one inhibitory. It can help understand how different neural firing patterns (like synchronous and asynchronous states) can emerge from simple network structures. The Brunel model is useful for studying the balance of excitation and inhibition and how this affects the global dynamics of neural circuits [6]. Another novel approach to modelling comes in the shape of the Potjans-Diesmann Model. This model is a more complex and layered network, representing a multilayer and structured network setup. It specifically models cortical microcircuits, incorporating multiple layers that correspond to different layers in the cortex (like layers 2/3, 4, 5, and 6). Each layer contains both excitatory and inhibitory neurons with connectivity parameters derived from anatomical studies [7]. This model is highly relevant for understanding cortical processing, and the interactions between different cortical layers, providing insights into more complex brain functions and information processing dynamics.

Simulating the complex dynamics of neurons and their networks requires powerful computational tools. One such tool that is extensively used in computational neuroscience is the NEST simulator: a flexible and scalable simulator for modelling large-scale neural systems [8] [9]. An example can better illustrate the parameters used in NEST, in this case, a simple two-population network configuration derived from the Potjans-Diesmann model will be considered. This configuration involves 2 neuron populations: one excitatory and one inhibitory. Here is a simple description of the most important parameters needed to define the network:

- C_{YX} - Connection Probability: Represents the connection probability between neurons in the network. It defines the proportion of neurons in population Y that are connected to neurons in population X
- J - Reference Synaptic Strength: Denotes the synaptic weight, which is the strength of the connection between neurons. It influences the effectiveness of communication between neurons, measuring how strong or weak a synaptic connection is compared to a standard reference. By multiplying J by the respective g_{YX} values the actual synaptic weight can be determined.
- g_{YX} - Synaptic strength: Describes the relative strength of synaptic connections between neurons of different types, indicated by the labels g_{ee}, g_{ei}, g_{ie}, and g_{ii}:
-
  - g_{ee}: Excitatory to Excitatory
  - g_{ei}: Excitatory to Inhibitory
  - g_{ie}: Inhibitory to Excitatory
  - g_{ii}: Inhibitory to Inhibitory

  This parameter quantifies how influential a signal from one population of neurons is on another, depending on whether the source and target are excitatory or inhibitory. An analogy can make this concept easier to grasp. Imagine there is a soccer game, a Local team and a Visitor team (representing the excitatory and inhibitory populations respectively). Passes between teammates represent the connectivities g_{ee} and g_{ii}, the "attacks" on the opposing team are the connetivities g_{ei} and g_{ie}.

2.1.3 Neural Summary Statistics

Neural data, either simulated or in vivo recordings, is complex as it represents neural communication such as spike trains describing action potentials and LFPs. To be able to make sense of this information statistical methods can be used to summarize the data, identify patterns, and highlight trends.

Spike train data in its simplest shape is a recorded sequence of neuronal electric activity, from which characteristics such as the firing rate can be calculated and used to characterize the recorded activity. The main calculations to be made are:

- **Mean Firing Rate**
  - It represents the average number of spikes per unit time, representing the overall activity level of the neuron[10]. It is calculated as:

$$\frac{N_{spikes}}{T}$$

- **Covariance of Interspike Intervals**
  - It measures the variability in time between consecutive spikes, reflecting the consistency or variability of spike timing. It is calculated as:

$$\frac{1}{n-1}\sum_{i=1}^{n}(ISI_i - \overline{ISI})^2$$

- **Fano Factor**
  - It calculates the variance-to-mean ratio of spike counts in a given interval. This metric is used to assess the dispersion of spike counts relative to a Poisson process. It is calculated as:

$$\frac{\mathrm{Var}(N_{spikes})}{\mathbb{E}(N)}$$

2.2 Machine learning classification fundamentals

Machine learning involves the development of algorithms and models that enable computers to learn by relying on patterns and inference from data. There are three main types of learning: supervised, unsupervised, and reinforcement learning. Unsupervised learning deals with unlabeled data; in this case, the algorithms try to find patterns or relationships in the data without preset output labels. Reinforcement learning, on the other hand, involves trial and error and receiving rewards or penalties. In this case the environment is designed to maximize a reward signal and this approach is used extensively in robotics, gaming, and other decision making scenarios.

Supervised learning is based on the use of labeled data to predict outcomes for new, previously unseen data. This type of machine learning needs a comprehensive dataset that is accurately labeled to work correctly. In practice, supervised learning is often used for regression (predict a continuous variable) and classification (predict the label of a variable). Algorithms that are typically used in this context are varied, including Decision Trees, Support Vector Machines, and Random Forests [11]. Table 2.1 contains a general overview and comparison of the mentioned methods.

Table 2.1: General Overview of Robust Classification Algorithms

Feature	**Decision Trees**	**Random Forests**	**Support Vector Machines (SVC)**
Basic Concept	A tree structure where each node represents a feature decision, leading to a classification or regression outcome [12]	An ensemble of decision trees that improves classification accuracy and controls overfitting [13].	A classifier that finds an optimal hyperplane which categorizes new examples [14].
Strengths	Simple to understand and interpret. Can handle both numerical and categorical data [12].	Reduces overfitting compared to individual decision trees. Handles high dimensional spaces well [13] [15].	Effective in high dimensional spaces [14]. Versatile with different kernel functions [16].
Weaknesses	Prone to overfitting especially with complex trees. Sensitive to noisy data.	Computationally expensive, less intuitive than single decision trees.	Requires careful choice of kernel and regularization parameters. Can be inefficient with large datasets.
Use Cases	Good for data exploration, suitable for scenarios where interpretability is important.	Suited for problems where robustness and accuracy are critical. Can handle diverse types of data.	Particularly good for classification of complex but small- or medium-sized datasets.
Typical Application	Medical diagnosis, credit scoring, customer segmentation.	Bioinformatics, environment modelling, multi-class prediction.	Image classification, bioinformatics, text categorization.
Performance	Fast training but can create overly complex trees that do not generalize well [12].	Training is slower due to building multiple trees, but predictions are robust.	Training time can be long; highly effective with the right kernel.

2.2.1 Decision Trees and Random Forests

Decision Trees are a type of supervised machine learning algorithm that represents decisions and their possible consequences in a tree-like structure, an schematic view of this algorithm can be seen in figure 2.3. Tree-structured classifiers are built by recursive splits of a set into 2 subsets (nodes) iteratively [12]. In decision trees, the node splitting follows the optimization of an objective function aimed at maximizing Information Gain [11]. The algorithm begins at the root node (entire dataset), then it splits the datasets based on an attribute; common attributes are Gini Impurity, Entropy, and Classification error. After the initial split at the root node using one of the attributes, the algorithm repeats the process on each of the split subsets using the same attribute. This process is repeated for each new split and continues until:

1. All elements in a subset are of the same class
2. The subset has reached a minimum size
3. The tree has reached maximum depth

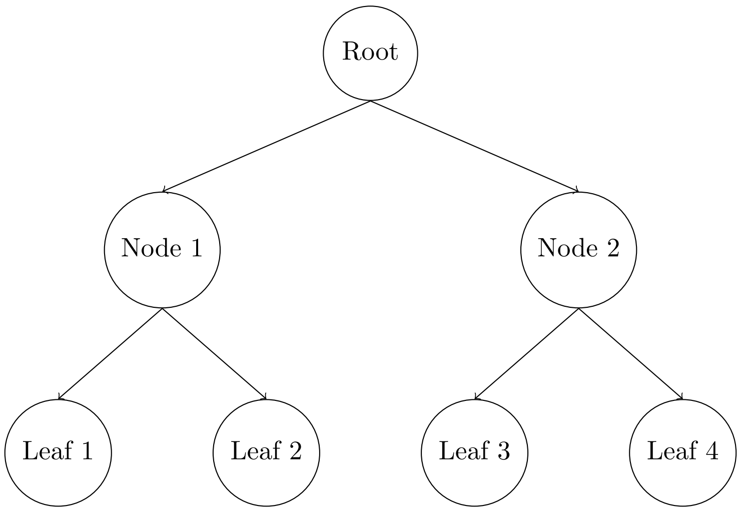

Figure 2.3: Simplified schematic of a Decision Tree.

Decision Trees are easy to interpret and are a robust model to handle non-linear relationships, however, they are sensitive (small changes in data can lead to different trees) and can cause overfitting. In order to tackle these issues, the ensemble learning method Random Forest can be used. This method enhances the decision tree method by creating many decision trees and aggregating their predictions [13, 15]. Each of the decision trees is built using random samples of data, and as the number of trees increases, the overall error rate converges. The accuracy of this method will depend on how strong each tree is and the correlation between trees [13]. A schematic representation of this algorithm can be seen in figure 2.4. While this method has many advantages and solves some of the issues of decision trees, it can be computationally expensive and sensitive to irrelevant features.

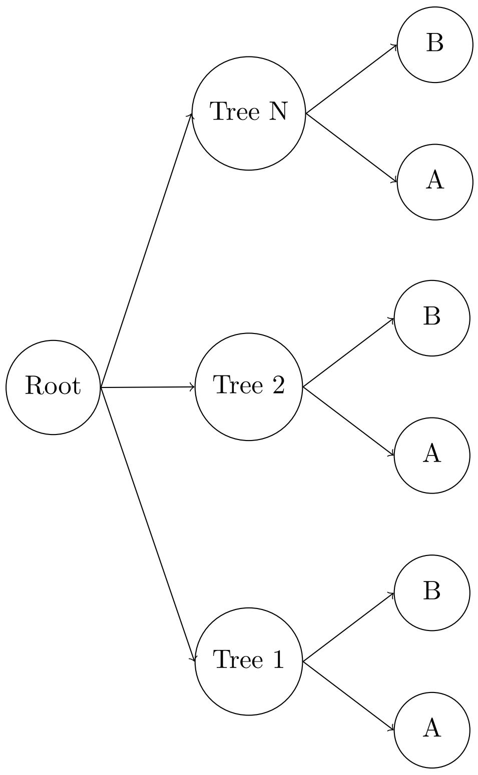

Figure 2.4: Simplified schematic of a Random Forest with multiple decision trees

2.2.2 Best practices

In order to improve the performance of models there are some best practices that need to be applied:

- Data preparation: this step includes the handling of missing layers, outliers and noise in the data. This can be done by omitting that data or using imputation techniques (such as using the mean to fill missing values). When dealing with noisy, numerical data with high outlier content, scaling techniques such as min max scaling and normalization can be used. When having categorical data it is also useful to use encoding techniques such as one-hot encoding as this helps in the performance of models.

- Feature engineering: this is an important step aiding the models as it can identify, create, or transform attributes that are relevant for the model and can increase its predictive power.
- Model evaluation: using accuracy on its own to gauge the performance of the model is not enough. Precision, recall, and the F1 score help interpret and explain models' predictions.

2.2.3 Handling of Imbalanced Datasets

Sampling is a process of selecting a subset of data from a larger dataset for many purposes, in this case for training models. Sampling can be random with each datapoint having an equal chance of being selected or, stratified where the data is divided into subgroups and samples taken from each subgroup to ensure adequate data representation. These sampling methods are sufficient when dealing with balanced datasets, yet would not yield good results when applied to imbalanced data.

When the number of observations in one class is significantly larger than observations in other classes, models tend to be biased towards the majority class. This results in poor classification performance for the minority class. In order to overcome this problem, some sampling strategies can be implemented:

- oversampling the minority class
- undersampling the majority class
- synthetic data generation

From the synthetic data generation approach, the method of Synthetic Minority Over-sampling Technique (SMOTE) stands out as a robust method that helps balance class distribution. This is done by generating synthetic samples rather than resampling the existing data. The steps that this process follows are:

1. Random sample selection from the minority class

2. The k nearest neighbors identification
3. Synthesis of new samples
4. Repetition

The samples created by this method are not just repeated copies of this data but are calculated based on the feature space dynamics of the minority class, making these new datapoints useful in the creation of decision boundaries for classification models [17].

2.3 Metamodelling

Metamodelling is a data-driven technique used in fields like computational biology and engineering. In essence, it serves as an abstraction layer above traditional modelling, providing a broader perspective that facilitates the examination, evaluation, comparison, and manipulation of models. This idea is represented best by the concept of "models of models"; a metamodel is basically a model trained with outputs from another model. These metamodels can give insights into the dynamics of the original models, such as approximating how changing the input parameters can affect the outputs without needing to fully understand all the complex details of the original system [18].

In computational biology, metamodels are useful for studying dynamic biological models. These models can be extremely complicated, with many interconnected parts and non-linear interactions, making them hard to analyze directly. Metamodels offer a way to understand these complex systems in a simpler form.

Metamodelling techniques have attracted interest across disciplines like engineering, systems biology, and computational neuroscience as a means of creating surrogate models for complex systems [19, 19, 20, 20, 21, 22, 23, 24, 25]. As data-driven approximations, metamodels offer more desirable computational properties and can run significantly faster than original models [25].

One of the objectives of metamodelling is to map the relationships between input parameters and output variables of an original model, providing a simplified yet accurate representation of its behavior. One commonly used approach for this is Ordinary Least Squares (OLS) regression, where each output variable is regressed individually on the input factors [23]. However, for highly nonlinear or non-monotonic relationships, OLS regression may not perform well. Partial Least Squares Regression (PLSR) is a more suitable alternative, as it can handle collinearities and nonlinearities better [21, 23]. In contrast, an advanced technique called Hierarchical Cluster-based PLSR (HC-PLSR) has shown promising results in systems biology [24]. In HC-PLSR, the data is first separated into groups using fuzzy clustering. Then, each group is analyzed separately with PLSR, producing regional regression models in addition to a global model. This approach can better capture complex, nonlinear input-output mappings, and has been applied successfully in various biological systems' models, such as gene regulatory networks, circadian clocks, and cardiac cells [24] [22] [3]. The results indicate that HC-PLSR can outperform traditional methods, especially for highly nonlinear or non-monotonic systems, and can provide insight into parameter interactions and sensitivities [3].

In addition to predicting model outputs from input parameters (classical, or forward metamodeling), it can also be used in the inverse direction, predicting input parameters from model outputs or empirical data (inverse metamodeling) [23]. This approach can potentially enable fast, non-iterative prediction of model parameters. However, inverse metamodelling requires a one-to-one correspondence between model inputs and outputs, which may not always be the case [23]. When multiple input combinations produce similar outputs, inverse metamodelling becomes challenging. One potential solution is to use Principal Component Analysis (PCA) on the model outputs alone, which can provide insights into the output patterns [23]; however, the PCA scores cannot be uniquely related to the model inputs without additional information.

Machine learning techniques, such as support vector machines, have been used in computational neuroscience because of the advantages they provide in handling high dimensional data, even with no linear boundaries between classes. For example, a detection and classification learning approach was developed to analyze spike wave forms and noise [26]. The classifiers worked with different scenarios of noise data and were effective in classifying spikes. Despite this, SVMs face challenges handling overlapping spikes and future research is needed to try different SVM based methods.

2.4 Evaluation metrics

The selection of metrics is an important step when using a metamodelling approach. In supervised learning, a confusion matrix is used as a way to show and compare the cases of correct and incorrect classification across classes. Accuracy, precision, recall, and F1 score are metrics used to evaluate how well a machine learning model performs at classification tasks and can be calculated from the confusion matrix.

Accuracy refers to the percentage of predictions the model got right across all classes, measuring the overall effectiveness of the classifier. It is the ratio of true predictions (true positives and true negatives) and the total number of predictions.

Precision refers to the true predictions, defined as ratio of true positive predictions and all positive predictions made by the classifier. While high precision means that the classifier is most likely correct when predicting a positive class, it alone does not provide interpretability of missed positive predictions (false negatives) [11].

Recall is a sensitivity measure defined as the proportion of actual positives correctly identified (true positives). This is an important metric in context where false negatives are not desired. This metric gives an evaluation of the model on how effective it is to capture true positives.

Another metric used is the F1 score, which is a statistical measure that balances precision and recall. It is calculated by taking the harmonic mean of these two metrics and its useful as a metric that reflects both false positives and false negatives, being more informative in case of imbalanced class distribution [11].

Chapter 3

Methodological Framework

This chapter will present the methods used in this research to reach the goal of estimating parameters (parameter ranges as labels) from outputs of a model (simulated neural activity). Firstly, an overview of the steps taken will be described in the Design of Reserach. Then follows the Data Generation and Exploration section, which includes a detailed explanation as to the origin of the simulation data with the specific parameters used within the NEST simulator; this section will include an exploration of the data used. This section will be followed by the Data Preparation, which will explain the steps taken to adapt the data to best fit the models to be used. Finally, the Application of Metamodelling techniques to compare the models will be explained, including the metric selection for the classification performance assessment.

3.1 Design of Research

This research is structured around a methodology that begins with the generation of data using the NEST simulator, followed by data preparation and analysis, finishing with the evaluation of the classification performance. Figure 3.1 shows the sequence of steps in this process.

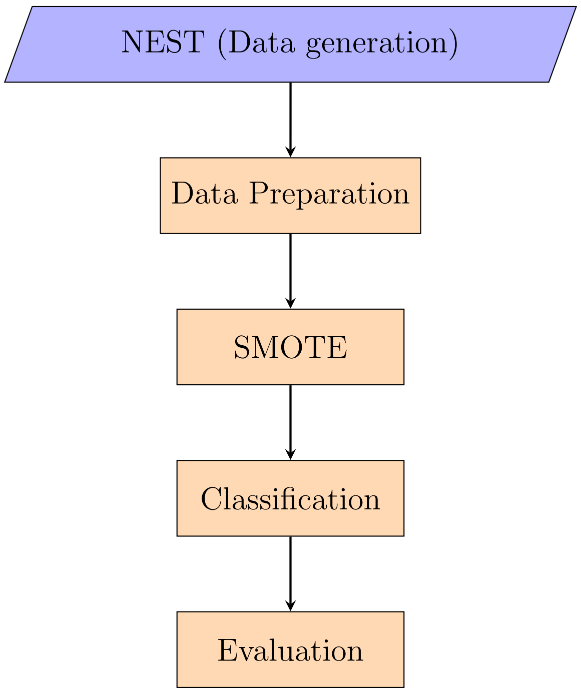

Figure 3.1: Flow diagram of the research steps

3.2 Data Generation and Exploration

This section aims to show the nature of the data used in this thesis. The data was generated using the NEST simulator (version 3.4), producing 10,000 simulation sets that mimic a specific two-population network configuration derived from the Potjans-Diesmann model. This model targets the fourth cortical layer, including both inhibitory (L4I) and excitatory (L4E) neuron populations. The data from these simulations (coupled with the parameters that originated them) were stored in 80 h5py files, totaling 5.12 GB.

Each of the files contains 125 sets of parameters (totaling 10.000 sets) and each set of parameters is coupled with 100 simulations. The data output for each simulation is made of two spike trains: one excitatory and one inhibitory, both containing timestamp arrays. The total simulation time was 2500ms.

From the files, the data contents that are relevant are:

- Static parameters: These are parameters that remain constant throughout all the simulations and that affect how the neuron populations interact, such as J, Cyx

- g_{YX} parameter space: the connectivity parameter g_{YX}, composed of 4 sub-parameters, dictates how the neuron populations interact and is the target of the metamodels prediction.
- Spike trains: Each parameter g_{YX} corresponds to 100 pairs of excitatory and inhibitory spike trains, representing the resulting neuronal activity of both populations for each run of the simulations.

In addition to this information, the h5py files contain parameter information (most of it repeated across all simulations), that define the neurons, the network, and the simulation details. Data is extracted using the h5py library, iteratively extracting the network parameters and the excitatory and inhibitory spike trains. To save time and space in the loading of the data, the important process variables are put into pickle files:

- g_{YX} parameters
- Excitatory spike trains
- Inhibitory spike trains

These pickle files will be later used to populate the dataframes used.

The Network and Simulation parameters used to generate the data are in tablea 3.1 and 3.2.

The data used for this study was extracted from the original h5py files and used to populate the dataframes further used. In table 3.3 there is an overview of the contents of the raw data when loaded into a Pandas DataFrame.

The data exploration shows that the data is highly complex, parameters uniformly distributed, and behaviors mostly have a gaussian distribution. In the following sections we will explore more in depth the g_{YX} parameter space and the spike train data.

- G_{YX} parameter space

This section will explore and describe the dataframe where the data within the g_{YX} parameter space will be stored. Ths data is loaded to a dataframe named df_g_{YX} from the pickle file previously described and has 4 columns.

Table 3.1: Network Parameters

Parameter	Value	Brief Description
N_E	21,915	Number of excitatory neurons
N_I	5,479	Number of inhibitory neurons
k_ext_E	2,100	External input to excitatory population
k_ext_I	1,900	External input to inhibitory population
C_YX	0.050, 0.135, 0.079, 0.160	Connection probabilities between neurons
d_E	1.5 ms	Synaptic delay for excitatory population
d_I	0.75 ms	Synaptic delay for inhibitory population

Table 3.2: Neuron Parameters

Parameter	Value	Brief Description
tau_m	10.0 ms	Membrane time constant
t_ref	2.0 ms	Refractory period
C_m	250.0 pF	Membrane capacitance
E_L	-65.0 mV	Resting membrane potential
V_th	-50.0 mV	Threshold potential
V_reset	-65.0 mV	Reset potential
tau_syn_ex	0.5 ms	Excitatory synaptic time constant
tau_syn_in	0.5 ms	Inhibitory synaptic time constant
J	87.81 pA	Reference synaptic strength

Table 3.3: Comparison of DataFrames df and df_g_{YX}

	Dimensions	**Columns**	**Data Stored**
df	1.000.000 rows with 2 columns	Spikes_E, Spikes_I	spike trains; each entry in the dataframe is an array storing timestamps
df_g_{YX}	10.000 rows with 4 columns	ee, ei, ie, ii	Subparameters, each entry is a real number

Table 3.4: Statistical Summary of Connectivity Parameters

Statistic	ee	ei	ie	ii
Mean	1.248380	-7.815476	1.247278	-7.807148
Std Dev	0.432327	3.024907	0.430247	3.005878
Min	0.500645	-15.894849	0.500557	-15.920007
Max	1.999423	-2.289307	1.999436	-2.264191

To begin the exploration and to observe the distributions of this parameter, a boxplot was created. In figure 3.2 we can see the distribution of the values within the subparameters in *df_*g_{YX}. Parameters EE and IE are referring to the excitatory population; it is clear from figure 3.2 that they share an almost identical parameter space. The differences that these share are evident in table 3.4 as comparing the maximum values for each parameter, where the limits of each quartile are, and other significant statistical measures differ. A similar case occurs with the distribution of values within the parameters IE and II. The distributions are similar in shape and scope, yet the specific values they hold differ. Table 3.4 contains the statistical description of all the parameters.

- Spike trains

Spike trains are the result of the simulations, in this case, they come in the shape of arrays that store timestamps that capture the activity of the respective populations. Each simulation consists of two spike trains, one excitatory and one inhibitory, with a simulation time of 250 ms, meaning all values stored in the arrays must be real numbers between 0 and 2500. From the 1000000 simulation samples 12132 show no activity in either population.

The first step to be able to work adequately with spike trains is to extract information from them. In this case, this step is performed at the same time as the

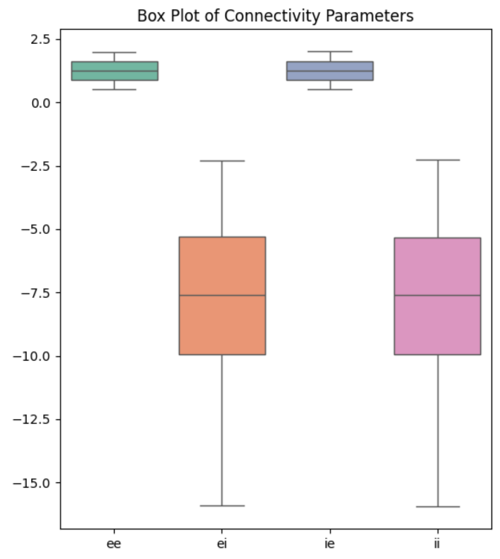

Figure 3.2: Box plot of the connectivity subparameters

population of the main spike train dataframe, for each array the following metrics will be calculated: mean firing rate, covariance of interspike intervals, mean interspike intervals, and fano factor. The resulting dataframe will contain additional columns for detecting empty spike trains.

3.3 Data Preparation

The data preparation section follows the best practices described in section 2.2.2 for a classification approach. In that sense, empty spike trains will be removed from the analysis along with their parameters in a global preparation step. As "input" for the classification model used there are the summary statistics from the spike trains, capturing the activity of the range of simulations studied. As "output" there is the parameter g_{YX}, containing the 4 subparameters ee, ei, ie, ii; it is the combination of these subparameters. The data needs to be adapted to the input format of the models in use. This preparation step will include data transformation and feature engineering separately for the input (spike trains) and output (parameters).

Another global preprocessing step is the data split. In this case the training dataset consists of 80% of the samples and the test size 20% for all aproaches.

3.3.1 Input preparation

Input for the models used are the data from the spike trains. Having each pair of spike trains in an individual row in the main dataframe (that has two columns) allows for quick and efficient feature engineering using common Python libraries, as Pandas, and NumPy. The output has 10 000 rows, and so the input must match this dimension; this makes necessary not only to characterize each pair of spike trains (each individual simulation) and also grouping the simulations and aggregating the data from 100 simulations each. This will lead to the input matching the dimension of the output, while a careful feature engineering is able to capture the general impact the parameters have on the spike trains at the moment of simulation.

For each spike train, there were calculations made to obtain: spike counts, mean firing rate, the covariance and variance of the interspike intervals, and the fano factor. These are calculated as described in section 2.1.3.

For the data aggregation, for every 100 rows and for the features previously mentioned, the following statistics were calculated: the mean, variance, standard deviation, median, skewness, and curtosis.

The end result of this processing is a dataframe with 10 000 rows and 60 columns storing a comprehensive aggregation of the features of each simulation set. The next step involves scaling; in this case, the standard scaler was used across all columns.

3.3.2 Output preparation

The target output variable in this research are the connectivity subparameters, in this case stored in the dataframe df_g_{YX}. As seen in the Data section, the values stored in the basic dataframe are continous numerical values. The raw data of the subparameters was first adjusted by the synaptic strenght constant and the connection probabilities matrix, as well as adjusting the parameters according to the number of neurons in the respective population.

In order to overcome the high complexity of approaching the raw data, the labeling approach was chosen using two variations. The first variant is normal quartile based labeling, this means that values got a label according to the quartile they belonged. Using this approach it makes evident the uniform distribution of all subparameters, as the labeling is balanced. The second approach for labeling involves grouping quartiles to simplify the problem even more, from a 4 class classification problem to a binary one. To get this imbalance, quartiles 1, 2 and 3 were labeled as class 0 and quartile 4 as classs 1.

3.4 Application of Metamodelling Techniques

To deal with this classification problem, for both sampling approaches (normal sampling, and SMOTE), the initial steps taken were similar, like using a pipeline, but the difference enters at the moment of the creation of the training dataset.

For the balanced labeled data approach, the procedure is as follows:

1. Pipeline Definition: A pipeline is defined with the classifiers to be compared (SVC and Random forests).
2. Fitting the Pipeline: The pipeline is then fitted with the training data, where the aggregated summary statistics from the spike trains are the input features ”X”, and the labeled synaptic strength parameters are the target ”y”.
3. Prediction: After fitting, the pipeline is used to make predictions based on the test X data. The model generates output labels based on what it has learned from the training data.
4. Accuracy Calculation: The accuracy of the model is calculated by comparing the predicted labels with the actual labels in the test Y data.

For the imbalanced labeled data approach, the procedure is as follows:

1. Pipeline Definition: An imbalanced pipeline is defined specifically to address the imbalance in the class distribution, using as classifiers SVC and Random forests.
2. Applying SMOTE: Before fitting the pipeline, SMOTE is applied to the training dataset. This generates synthetic examples of the minority class to offset the imbalance in the minority class distribution.

3. Fitting the Pipeline: After enhancing the dataset with SMOTE, the pipeline is fitted with the balanced training data, where the aggregated summary statistics from the spike trains are the input features "X", and the labeled synaptic strength parameters are the target "y"

4. Prediction: Once the pipeline has been fitted, it is employed to make predictions based on the test X data.

5. Accuracy Calculation: Finally, the accuracy of the model is assessed by comparing the predicted labels with the actual labels in the test Y data.

Comparative Analysis of Classification Models

The comparison of results across all classification attempts will be done with the following metrics: precision, recall, and F1 score. These metrics are obtained via scikit-learn's $classification\_report$

Chapter 4

Results

4.1 Classification results without augmentation

Here in table 4.1 are presented the results of the classification predictions of the Random Forest algorithm when compared with SVC. The target variable has the suffix "*_cat*" as it represents the categorized variable used in this stage. Results show that precision and F1 scores in Random Forests are higher compared to SVC, yet the overall performance of both methods is low. Both SVC and Random Forests exhibit close-to-random-chance prediction precisions, meaning that the models are making incorrect label predictions more frequently than if they were just guessing. A similar situation can be seen with the F1 scores. Random forests F1 score is slightly more consistent across targets and labels, but it still shows low performance. Comparing the individual performances of the classification models respective to each target and each label can provide additional insight regardless of the low performance. Parameter targets related to the excitatory population (g_{ee}_cat, and g_{ie}_cat) show similar patterns in the individual F1 scores in the SVC method across labels, with some exhibiting extremely poor performance. Label "H" in g_{ee}_cat and labels "L" and "M" in g_{ie}_cat show F1 scores of 0.17, 0.12 and 0.14 respectively, meaning high false positive rate combined with high false negative rate. In contrast, label "L" in g_{ee}_cat and label "V" in g_{ie}_cat show F1 scores of 0.33 and 0.35 respectively, showing a better performance. Comparing the F1 score across labels for SVC and Random forest show that overall the latter is more stable yet both have poor performance. Comparing precision scores in the same context show little variability between overall precision scores with values around random chance for a 4 class classification problem. Comparing precision and F1 scores for the parameters related to the inhibitory population (g_{ei}_cat, and g_{ii}_cat) show that those models have similar poor performance.

Table 4.1: Classification Results by Model and Category without data augmentation

Target	Label	SVC		RF	
		Prec.	F1	Prec.	F1
g_{ee}_cat	H	0.26	0.17	0.24	0.24
	L	0.24	0.33	0.25	0.27
	M	0.22	0.19	0.26	0.27
	V	0.27	0.20	0.27	0.25
g_{ie}_cat	H	0.24	0.22	0.25	0.25
	L	0.26	0.27	0.26	0.27
	M	0.22	0.25	0.20	0.20
	V	0.25	0.21	0.26	0.25
g_{ei}_cat	H	0.22	0.23	0.24	0.26
	L	0.30	0.12	0.29	0.27
	M	0.25	0.14	0.24	0.24
	V	0.25	0.35	0.25	0.25
g_{ii}_cat	H	0.25	0.22	0.22	0.23
	L	0.25	0.24	0.26	0.27
	M	0.25	0.27	0.22	0.23
	V	0.27	0.28	0.23	0.22

Table 4.2 shows a comparison of the model accuracies across targets. Despite minor differences in accuracy, Random forests and SVC both show low performance.

Table 4.2: Classification Accuracies by Model and Category without data augmentation

Target	Method	Average Accuracy
g_{ee}_cat	SVC	0.24
	Random Forest	0.26
g_{ie}_cat	SVC	0.24
	Random Forest	0.24
g_{ei}_cat	SVC	0.25
	Random Forest	0.25
g_{ii}_cat	SVC	0.26
	Random Forest	0.24

4.2 Classification results with data augmentation

Table 4.3 shows a comparison of the precision and F1 scores of Random Forests and SVC across targets and labels. In this case, Random Forests shows higher precision and F1 scores across all results and labels. SVC, on the other hand shows an imbalance in the performance across labels. From the table, the comparison between the two approaches in the excitatory parameters (g_{ee}_cat, and g_{ie}_cat) reveals that Random forest has a superior performance averaging 80% in the performance metrics. SVC shows a performance on par with the imbalance of the data; classification precision in label 0 across targets tends towards 75%, which corresponds with label 0 consisting of the "L", "M", and "H" categories in the encoding. This is in balance with SVC showing average of 25% in the precision. It is important to note that the SVC algorithm shows a higher than random chance of F1 score for the minority label 1. This can mean either low false positives or low false negatives only for the minority class.

The average accuracies of both methods are shown in table 4.4 . SVC shows low accuracy across targets, with values ranging from 0.44 to 0.54. However, random forests show much better performance averaging 0.8 across targets, outperforming SVC.

Table 4.3: Classification Results by Model and Category with SMOTE

Target	Method	Label	Precision	F1-Score
g_{ee}_*coded*	Random Forest	0	0.78	0.80
		1	0.81	0.79
	SVC	0	0.73	0.58
		1	0.25	0.33
g_{ei}_*coded*	Random Forest	0	0.80	0.81
		1	0.80	0.79
	SVC	0	0.74	0.64
		1	0.28	0.35
g_{ie}_*coded*	Random Forest	0	0.80	0.82
		1	0.82	0.80
	SVC	0	0.74	0.50
		1	0.26	0.37
g_{ii}_*coded*	Random Forest	0	0.79	0.82
		1	0.84	0.80
	SVC	0	0.75	0.53
		1	0.25	0.36

Table 4.4: Classification Accuracies by Model and Category with SMOTE

Target	Method	Average Accuracy
g_{ee}_*coded*	SVC	0.48
	Random Forest (SMOTE)	0.79
g_{ei}_*coded*	SVC	0.54
	Random Forest (SMOTE)	0.80
g_{ie}_*coded*	SVC	0.44
	Random Forest (SMOTE)	0.81
g_{ii}_*coded*	SVC	0.46
	Random Forest (SMOTE)	0.81

Chapter 5

Discussion

Changing the problem from a 4-class problem into a binary classification one helped increase the learning capabilities of the random forest classification model, that is, when SMOTE is applied.

Applying advanced machine learning techniques can help in parameter extraction by being able to differentiate different kinds of neuronal activity and the relationship to the parameters that can have originated from. Data from neuronal network simulations is extremely complex, that is why different approaches must be explored and also different combinations of approaches, in this case it was the combination of SMOTE with an ensemble algorithm that provided the better accuracy, yet, 80 percent accuracy suggests there is much more room for improvement in future research. It also must be taken into account that advanced machine learning techniques must be coupled with a mindful feature engineering process.

The methodology of metamodelling can help also neurologists as it can help if they require a specific data analysis approach for interpreting patient data. As neurobiologists deal with much patient data with many aflictions (like Alzheimer and Parkinson disease); the approach presented here could be adapted to the interpretation of spiking neural network data aiding in interpretability when coupled with other techniques. Using this approach could help in characterizing edge behaviors.

Research on the area is constantly updating and growing. From using HC-PLSR in 2012 to a robust approximate bayesian computation approach in latter years, data intensive approaches have gained popularity and efficiency.

5.1 Interpretation of results

The accuracies of both SVC and Random Forest with regular sampling suggest that neither approach effectively distinguishes between the 4 classes of the subparameters, while the Random forest + imbalanced labeling + SMOTE method is the one that has the highest accuracies across labels and across subparameters, averaging 80 percent accuracy. By artificially generating samples from the minority class, SMOTE helps the model generalize and be able to increase its prediction accuracy, precision, and F1 scores. Results can be summarized as:

1. SCV and Random forest with quartile based labeling: Both methods show very low accuracy, 25% for a 4 class problem suggests a performance near random chance. This shows that the models are not learning from the data based on the engineered features used.
2. SVC with imbalanced labeling and SMOTE; the performance is similar as in the previous case, the changes in sampling had no effect in improving the 75% accuracy that, for this imbalanced labeling, is an accuracy near random chance.
3. Random forest with imbalanced labeling and SMOTE: the average accuracy rose to 80% when using this combination of techniques.

Comparing the performance of the classifiers used in this study shows that even using robust machine learning methods coupled with data analysis best practices and feature engineering can fail to capture the complexities of the data without the right sampling approach. Using SMOTE increased the performance of the Random Forest algorithm, yet further steps may be needed to help the model generalize better from the dataset and increase its accuracy.

The other models, having a performance of 25, 25, 75 percent respectively fail to capture the data's complexity, but this could be improved by different feature engineering or sampling techniques in future research. Such low performance could be improved by more data but the addition of data would be more useful to improve the accuracy of the Random forest + SMOTE as it is already more suited to interpret data.

Performance metrics in imbalanced datasets can be misleading, and a high overall accuracy mean not reflect the algorithms ability to correctly classify the minority class. Using SMOTE can help solving this issue but the inclusion of actual additional data representing the minority class may be able to boost the performance.

5.2 Reflection on Research Questions

- How does the categorization of synaptic strengths into "high" and "low" impact the predictive accuracy and interpretability of machine learning models in identifying key dynamics within neural networks?

This helps in recognizing parameters closer to the edges, as they show a characterizable behavior by the models. The dynamics within neural networks are complex and more data and studies are needed to properly characterize them; the approach taken in this research helps in providing an initial framework to analyze this data. Turning continuous synaptic strength values into binary categories can also help in the interpretability of the relationships of neuron dynamics and the parameters. An application for such capabilities may be in medical diagnosis as edge behavior associated with pathologies may be detected, as well as the detection of the effectiveness of treatments. However, categorizing synaptic strengths into 2 labels may result in loss of information and oversimplification of the neural dynamics.

- Can a metamodelling approach improve the selection and tuning of machine learning algorithms for modelling synaptic strength dynamics in neural network simulations, leading to more efficient generalization across different network configurations?

Yes, but more research and data are needed. The mere comparing of different algorithms may provide insights of the data, as some may work better than others that can be considered a result on itself. Each model approach responded differently, giving a high variability in the effectiveness of the models. This variation is what makes metamodelling a relevant framework for working with this kind of data. More complex configurations of networks may face more or different issues when predicting parameters from behavior; that is field for further research.

Chapter 6

Conclusion

Modelling complex dynamic systems is a complicated task, time and resources intensive. Characterizing such a model and predicting its behavior from the data it generates, as well as reverse engineering the parameters that gave way to a certain behavior of the model can be useful for researchers. The latter approach was the focus of this research, aiming to bridge the gap between inputs and outputs of a model to be able to make more precise simulations, narrowing the parameter space that would be used to generate a particular behavior of the system.

The benefit of using this approach is that it allows for a first interpretation/prediction stage based on the data that is fed to the metamodel. This approach is designed for dealing with simulated data from the NEST simulator and focusing on the interactions of two populations; one excitatory and one inhibitory. In this case the output of those simulations are spike trains in the shape of timestamps. Based on the extracted and aggregated characteristics of these spike trains belonging to a set of labeled parameters, the metamodel based on Random Forests with imbalanced labelling and SMOTE is able to have an 80 percent overall accuracy in classifying neural activity with labeled network parameters. Being able to have such classification can help researchers explore edge cases (when the parameters are in the fourth quartile of their distribution); this can also be due to these parameters showing more extreme behavior, which is captured by the metamodel.

When analyzing data from such a complex system as NEST, which attempts to emulate the even more complex brain, one runs into many difficulties as stochastic complex systems' behavior is problematic to model and to predict. The dynamics the parameters play in the simulations to give way to the spike train behavior are obscured by the many distinct combinations that can occur and by the nature of the system itself. Just applying bare machine learning methods, however advanced as they may be was not enough to elucidate the behavior of the system; powerful classification models failed to capture the data's behavior for predictions above random chance. Even when applying advanced sampling techniques, the performance has much more room to improve. It is important to note that since each set of parameters were coupled with 100 simulations data had to be aggregated so that the dimensions of the inputs and outputs of the metamodel fit. This aggregation was done on extracted features of the spike trains, from mean firing rate to the fano factor. Different data engineering and/or data aggregation techniques may be applied to either the parameters or the spike trains in future research.

Research could be focused on time series analysis or non-linear dynamics that may provide deeper insight into neural interactions. More advanced machine learning techniques may also be explored; neural networks in the many varieties could be better suited to deal with the nature of the data.

Bibliography

[1] T. Trappenberg, *Fundamentals of Computational Neuroscience*, OUP Oxford, 2010.

[2] David Sterratt, Bruce Graham, Andrew Gillies, and David Willshaw, *Principles of computational modelling in neuroscience*, Cambridge University Press, 2011.

[3] Kristin Tøndel, Jon Olav Vik, Harald Martens, Ulf G Indahl, Nicolas Smith, and Stig W Omholt, “Hierarchical multivariate regression-based sensitivity analysis reveals complex parameter interaction patterns in dynamic models,” *Chemometrics and Intelligent Laboratory Systems*, vol. 120, pp. 25–41, 2013.

[4] Alan L Hodgkin and Andrew F Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *The Journal of physiology*, vol. 117, no. 4, pp. 500, 1952.

[5] Anthony N Burkitt, “A review of the integrate-and-fire neuron model: I. homogeneous synaptic input,” *Biological cybernetics*, vol. 95, pp. 1–19, 2006.

[6] Nicolas Brunel, “Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons,” *Journal of computational neuroscience*, vol. 8, pp. 183–208, 2000.

[7] Tobias C Potjans and Markus Diesmann, “The cell-type specific cortical microcircuit: relating structure and activity in a full-scale spiking network model,” *Cerebral cortex*, vol. 24, no. 3, pp. 785–806, 2014.

[8] Marc-Oliver Gewaltig and Markus Diesmann, “Nest (neural simulation tool),” *Scholarpedia*, vol. 2, no. 4, pp. 1430, 2007.

[9] Marc-Oliver Gewaltig, Abigail Morrison, and Hans Ekkehard Plesser, “Nest by example: an introduction to the neural simulation tool nest,” *Computational systems neurobiology*, pp. 533–558, 2012.

[10] Wulfram Gerstner, Werner M. Kistler, Richard Naud, and Liam Paninski, *Variability of spike trains and neural codes*, p. 168–201, Cambridge University Press, 2014.

[11] Sebastian Raschka and Vahid Mirjalili, *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*, Packt publishing ltd, 2019.

[12] Leo Breiman, *Classification and regression trees*, Routledge, 2017.

[13] Leo Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[14] Corinna Cortes and Vladimir Vapnik, "Support-vector networks," *Machine learning*, vol. 20, pp. 273–297, 1995.

[15] Khaled Fawagreh, Mohamed Medhat Gaber, and Eyad Elyan, "Random forests: from early developments to recent advancements," *Systems Science & Control Engineering: An Open Access Journal*, vol. 2, no. 1, pp. 602–609, 2014.

[16] Dustin Boswell, "Introduction to support vector machines," *Departement of Computer Science and Engineering University of California San Diego*, vol. 11, 2002.

[17] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[18] Cesar Gonzalez-Perez and Brian Henderson-Sellers, *Metamodelling for software engineering*, Wiley Publishing, 2008.

[19] Dong Zhao and Deyi Xue, "A multi-surrogate approximation method for metamodeling," *Engineering with Computers*, vol. 27, pp. 139–153, 2011.

[20] Anja Stene, "Metamodelling of simulation results from brunel's neural network model using local multivariate regression (hc-plsr)," M.S. thesis, Norwegian University of Life Sciences, Ås, 2020.

[21] Harald Martens, Kristin Tøndel, Valeriya Tafintseva, Achim Kohler, Erik Plahte, Jon Olav Vik, Arne B Gjuvsland, and Stig W Omholt, "Pls-based multivariate metamodeling of dynamic systems," in *New Perspectives in Partial Least Squares and Related Methods*. Springer, 2013, pp. 3–30.

[22] Kristin Tøndel, Ulf G Indahl, Arne B Gjuvsland, Stig W Omholt, and Harald Martens, “Multi-way metamodelling facilitates insight into the complex input-output maps of nonlinear dynamic models,” *BMC systems biology*, vol. 6, no. 1, pp. 1–21, 2012.

[23] Kristin Tøndel and Harald Martens, “Analyzing complex mathematical model behavior by partial least squares regression-based multivariate metamodeling,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 6, pp. 440–475, 2014.

[24] Kristin Tøndel, Ulf G Indahl, Arne B Gjuvsland, Jon Olav Vik, Peter Hunter, Stig W Omholt, and Harald Martens, “Hierarchical cluster-based partial least squares regression (hc-plsr) is an efficient tool for metamodelling of nonlinear dynamic models,” *BMC Systems Biology*, vol. 5, no. 1, pp. 1–17, 2011.

[25] Jan-Eirik W Skaar, Alexander J Stasik, Hans Ekkehard Plesser, Gaute T Einevoll, and Kristin Tøndel, “Metamodelling of a two-population spiking neural network,” *bioRxiv*, pp. 2022–09, 2022.

[26] R Jacob Vogelstein, Kartikeya Murari, Pramodsingh H Thakur, Chris Diehl, Shantanu Chakrabartty, and Gert Cauwenberghs, “Spike sorting with support vector machines,” in *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2004, vol. 1, pp. 546–549.

Appendix A

Appendix 1: AI Usage

I am aware of the AI Usage regulations at NMBU as found here. This appendix will explain which AI tools were used and how.

- Perplexity: This tool was used in combination with search engines to find relevant literature. Perplexity works as an AI powered search engine, I was able to ask very specific questions or provide more context to my searches when using this tool, as well as expanding the searches on the results it already provided. Perplexity gives the sources, links to the sources, and a short overview or outline of the contents included. The content created by Perplexity was also important to understand if the sources it provided were relevant at a glance, and all the sources used in the work provided by Perplexity were accessed, reviewed, and, when possible, downloaded. Some examples of the uses are:
  - "What was the first paper to show SVM?"
  - "What are use cases for metamodelling applied in parameter estimation on synthetic neural data?"
    - what sources are the most relevant in parameter estimation in the context of neural data coming from the NEST simulator?
- ChatGPT: GPT4 was used for many tasks; to quickly outline and summarize long or complex sources, explain hard to understand terms in them, and find relationships (either similitudes or differences) between pieces of information. It was also used to generate quick outlines of subsections. Content output from ChatGPT was fact-checked and reviewed.

  - "From this text insert text 1 here, provide me with a short overview of the key points"
    * "The term insert term in your assessment is not clear, explain it in simpler terms"
    * "From the context of your previous response, what is the link of that information with insert text 2 here and inser text 3 here in the context of (i.e. metamodelling, classification in machine learning, etc)"
    * "From your previous response and in the same context, provide a comprehensive yet simple outline that would encompass the information I provided"

- Grammarly: Grammarly was used extensively for grammar correction and sentence structure correction suggestions. This tool works on text I have written, with autocorrect (not accurate all the time) and sentence structure suggested changes (some I have applied, some I have not). Grammarly also has generative AI, but this feature was not used in this work.