UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

DEEP LEARNING FOR PREDICTING DISEASE PROGRESSION OF CLINICAL ENDPOINTS IN ALS

Lucas Barreto Silva

Mestrado em Engenharia Informática

Dissertação orientada por:
Prof. Doutora Sara C. Madeira
Prof. Doutor Nuno Garcia

2022

"Adversity is the first path to truth."
G.G. Byron Don Juan

Acknowledgments

Words cannot express my gratitude to my Family for their invaluable support and love throughout my life path, especially my Parents, Maria and António, who always guided me and made it possible for me to chase my dreams, none of this would have been possible without them, my sister Ana and my dogs Luna and Leia, who were by my side here in Portugal, my godfather Jorge and my grandparents who were always thinking about me despite being far away in Brazil.

I am also thankful to my friends and college colleagues who I spent with most of the last five years, and shared great moments, that made me who I am today. To the "Luso-Gang" group, Catarina Silva, Diego Barros, Guilherme Vieira and Pedro Santos, for the incredible journey during our bachelor, with experiences that I will not be able to forget and will cherish for life. A special thanks to my "LáEmCima" group, Afonso Assunção, Bruno Santos, Carolina Ramos, Constança Garcia, Diogo Azevedo, Francisco Madruga, Guilherme Vieira, Leandro Ribeiro, João Rodrigues, João Nogueira, Pedro Nogueira, Primo, Rodrigo Cassanheira and Rodrigo Penedo for the special moments and laughs shared. To the friends I did not mention, I want to thank for the contributions throughout the years.

I would like to express my appreciation to my advisor Nuno Garcia, for the significant insights given during this project duration, for always being available to answer any questions and helping make this work possible. To Sara Madeira, for believing and choosing me to work on this project, as well as the great ambition to research and help others. I am also grateful to Faculdade de Ciências da Universidade de Lisboa, while it was a short passage, it was very meaningful to be able to be apart of such a community.

Finally, a formal acknowledgement to Fundação para a Ciência e Tecnologia (FCT) funding through the project "AIpALS – Advanced learnIng models using Patient profiles and disease progression patterns for prognostic prediction in ALS" (PTDC/CCI-CIF/4613/2020).

Abstract

Amyotrophic Lateral Sclerosis (ALS) is a neurodegenerative disease vastly known for its rapid progression, usually leading to death within a few years, by respiratory failure. Since there is no cure currently known, the main objective is treatment in order to improve symptoms and prolong survival. A treatment that is known to be effective, is Non-invasive Ventilation (NIV), being capable of extending life expectancy and improving quality of life. Therefore it is imperative to administrate it preemptively. However Amyotrophic Lateral Sclerosis (ALS) affects most muscles of the body, so there will exist many clinical conditions that require some kind of treatment. In this work, we propose two approaches that use deep learning to predict Amyotrophic Lateral Sclerosis (ALS) disease progression, to know when a patient should be treated or not for a given clinical endpoint. In the first approach, we use the various snapshots of the patients as input without taking into consideration the temporal dependence between the available features, so instead we use each assessment of the patient as a single instance to feed the models. In this approach, we propose the use of a MLP and a CNN, to predict the outcome for the time windows available (90, 180 and 365 days), using the instances mentioned before and perform class resampling on the training sets, using SMOTE on the minority class and Random Undersampling on the majority class. In order to have more data to train the models and have a balanced set, which helps achieving a better performing model on the test set. In the second approach, we take the snapshots of the patients and group them by the patient reference, and proceed to only used those that have length of 3 and 4. On the sequences of length 3, we performed padding in order to have the same length as the sequences of length 4, by simply taking the first instance of this sequence and use it as the first and second assessment on the sequences. We proceed to use the sequences to feed them into a LSTM model, and train for the several datasets and retrieve the scores obtained. On this approach we do not perform any type of class resampling, as the first approach does. The results obtained show some promising insights, on the first approach in which we use the instances of the patients, the preprocessing performed was one of the main factors to the great results obtained. On the second approach, which the temporal sequences of length 3 and 4 are used, the results obtained are not as promissing as the first approach, however there are still improvements that could be done.

Keywords: Amyotrophic Lateral Sclerosis, Clinical Endpoint, Non-invasive Ventilation, Deep Learning, Disease Progression

Resumo

Esclerose Lateral Amiotrófica (ELA) é uma doença neurodegenerativa amplamente conhecida pela rápida progressão, geralmente leva à morte em poucos anos, entre os 3 e 5 anos, devido a insuficiência respiratória. Como não existe nenhuma cura conhecida atualmente, o principal objetivo é o tratamento de forma a melhorar os sintomas e prolongar a expectativa de vida dos pacientes. Um dos tratamentos conhecido por ser eficaz, é a Ventilação Não-Invasiva (VNI), sendo capaz de prolongar a expectativa de vida e melhorar a qualidade da mesma. Portanto é imperativo administrá-lo preventivamente. E porque não administrar este tratamento a todos os pacientes? Devido a nem todos os pacientes são elegíveis para começar o tratamento de *NIV*, e ainda existe discordância sobre quando é o momento ideal para começar este tratamento. Para tal, é imperativo procurar prever quando um paciente estará elegível para um dado tratamento num futuro próximo. No entanto, a Esclerose Lateral Amiotrófica (ELA) afeta a maioria dos músculos do corpo, desta forma existem diversas condições clínicas que requerem algum tipo de tratamento, e atenção médica.

Até à data, há diversas aplicações promissoras de modelos preditivos para a previsão de *NIV*, no entanto apenas aplicado ao tratamento de *NIV*, e dado que *ALS* afeta a maioria dos músculos do corpo, existe assim um potencial por explorar. Desta forma temos disponíveis vários datasets com diferentes condições clínicas, para a qual vamos utilizar modelos com o intuito de realizar a previsão nas mesmas janelas de tempo.

Neste trabalho, propomos o uso de *Deep Learning* para prever a progressão da doença Esclerose Lateral Amiotrófica (ELA), para saber se um paciente deve ou não ser tratado para uma dada condição clínica, dentro de janelas de tempo pré-definidas (90, 180, e 365 dias).

No entanto, antes de treinarmos qualquer modelo precisamos de realizar o tratamento de dados nos datasets disponíveis, para tal, os conjuntos de dados usados consistem em pacientes com ALS após serem diagnosticados e receberem terapia na clínica de ALS do Centro Hospitalar Universitário de Lisboa Norte, Hospital Santa Maria, de 1995 a 2021. Os dados consistem de informação estática, que contém informação demográfica sobre o paciente, e informação temporal que consiste do resultados de uma série de avaliações clínicas de testes específicos, que incluem testes respiratórios, dados neurofisiológicos, e respostas ao questionário da *ALS Functional Rating Scale*.

Como os conjuntos de dados apresentavam dados em falta em grande parte das colunas, tivemos que realizar *data imputation*, tivemos em conta o tamanho dos conjuntos de dados e tentámos remover o mínimo de linhas possível. Para tal, decidimos utilizar o *K-Nearest Neighbours Imputation (KNN Imputation)*, em colunas que tinham um número de dados em falta abaixo de um valor definido por nós, e as colunas que tivessem mais que o valor definido são retiradas. Para as que estavam dentro do valor definido iam ser introduzidos nos dados em falta, valoresc calculado por outras instâncias mais próximas em termo dos vários valores que possuem. Mais tarde, usamos estes novos conjuntos de dados para realizar *feature selection*, para obtermos as que melhor definem o conjunto de dados.

Na primeira abordagem, usamos os vários *snapshots* dos pacientes como input sem ter em consideração a dependência temporal entre as *features* disponíveis, sendo assim usamos cada avaliação do paciente como uma única instância para fornecer aos modelos. Nesta abordagem, propomos o uso de uma *MLP* e uma *CNN*, para prever o resultado para as janelas de tempo anteriormente mencionadas, usando as *snapshots* mencionadas anteriormente e realizar *Class Resampling* nos conjuntos de treino, usando o *SMOTE* na *minority class* e o *Random Undersampling* na *majority class*. De forma a termos mais dados para treinar os modelos e ter um conjunto balanceado, dado que ajuda a obter melhor desempenho do modelo no conjunto de teste.

Na segunda abordagem, tiramos os *snapshots* dos pacientes e os agrupamos pela referência de cada paciente, e passamos a utilizar apenas aqueles que possuem comprimento 3 e 4. Nas sequências de comprimento 3, realizamos *padding* para ter o mesmo comprimento que as sequências de comprimento 4, simplesmente utilizamos a primeira instância dessa sequência, como primeira e segunda avaliação das sequências. De seguida, usamos as sequências para fornecer a uma *LSTM* e treinamos para os vários conjuntos de dados, no final obtemos as avaliações de cada modelo. Nesta abordagem não realizamos nenhum tipo de reamostragem de classe, ao contrário do que acontece na primeira abordagem.

Os nossos resultados demonstram que, a primeira abordagem da MLP e da CNN foi capaz de obter resultados bastante promissores para alguns dos tratamentos disponíveis, alcançando até um valor de 92% na métrica AUC no *Clinical Endpoint C2*, na janela de tempo de 90 dias. Sendo este resultado uma surpresa, dado que esperávamos que os conjuntos de dados com melhores resultados seriam os de 365 dias, devido a terem um maior número de instâncias positivas que os conjuntos de dados de 90 e 180 dias. Relativamente à segunda abordagem com a LSTM, e com a utilização de sequências temporais, não atingimos resultados tão promissores. Devido a reduzirmos a quantidade de dados disponível para o treino dos modelos, já que utilizamos sequências de comprimento 3 e 4, e deixamos de parte o resto das intâncias dos dados, leva a não atingirmos uma performance tão boa quanto à abordagem anterior.

Palavras-chave: *ALS Functional Rating Scale, Data Imputation, Deep Learning*, Esclerose Lateral Amiotrófica, Ventilação Não-Invasiva,

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

Amyotrophic Lateral Sclerosis (ALS) is a neurodegenerative disease with no cure and a short life expectancy following the start of symptoms, with most patients dying of respiratory failure within 3-5 years. Only a few of the available treatments increase the life expectancy by a few months. Being Non-invasive Ventilation (NIV) the technique that has showed the most promise in terms of extending life expectancy and improving quality of life, there is research that suggests that starting it sooner is better. So why not treat every patient with Non-invasive Ventilation (NIV)? Not all patients are eligible to begin the NIV treatment, and there is still some disagreement over when the optimal time is to begin NIV [12]. Therefore it is a good idea to look into whether it is possible to predict whether patients will be eligible for NIV in the near future based on their progression. However there are other clinical endpoints just as important to predict, such as the need of a wheelchair. Given that ALS affects most muscles of the body, there will exist many clinical conditions that require some kind of treatment, which motivates the study of algorithms to analyse the evolution of other clinical endpoints.

1.2 Goals and Contributions

This dissertation follows the work developed by Carreiro et al. [7] which proposed the first prognostic models based on predefined time windows to predict the disease progression and the need of NIV treatment in ALS. Following this work, Pires et al. [32] proceeded to categorize patients depending on their stage of disease progression, stratifying them in 3 groups (slow, neutral and fast). Afterwards they used the data from each of the groups and build a specialized model, achieving promising results. However neither of these approaches take into consideration the temporal dependence between the available features. Regarding deep learning approaches, Muller et al. [30] made use of neural networks, mainly the Long Short-Term Memory (LSTM), to predict the disease progression of the

patients in the previously mentioned time windows and Shapley Additive Explanations (SHAP) to explain complex machine learning models.

Our goal is to predict Amyotrophic Lateral Sclerosis (ALS) disease progression, in order to know when a patient should or not receive treatment for a given clinical endpoint, within a predefined time window. To this end, we propose the use of deep learning models, to predict disease progression based on temporal clinical data. This approach is not common due to the fact that deep learning models tend to be quite complex and difficult to interpret and usually there is not enough data available.

In this work, our data consists of static and temporal information, the first being demographic data and the second being the outcomes of a series of clinical assessments of specific tests, from 1093 ALS patients observed from the early 90's until October 2021, in the form of multivariate time series. These scores and the demographic data will then be used to feed as input to our models later on.

In order to achieve a model capable of making such disease progression predictions, we propose the use of a Multilayer Perceptron (MLP) and a Convolutional Neural Network (CNN) on a first approach, and a Long-Short Term Memory (LSTM) with temporal sequences from the patients' data. One of the main advantages of deep learning approaches is having the ability to solve complex problems, which require finding hidden patterns in the data. This approach shines when there is a lack of domain understanding for feature introspection, since feature engineering is less of a concern [26].

Concerning the prediction of different clinical endpoints, such as the need of Non-invasive Ventilation (NIV) or the need of a wheelchair, we will make use of the proposed models and take into account the clinical history and temporal data of the patients, in order to predict the outcome.

In this scenario, the previous approaches to the problem at hand do not make use of the temporal data available. In our approach, the objective is to use the temporal dependence between features, in addition to the static data already used, to implement deep models capable of predicting disease progression of different clinical endpoints. The main contributions of this work are:

- The proposal and evaluation of different deep neural networks to predict disease progression of several clinical endpoints in Amyotrophic Lateral Sclerosis (ALS).
- Explanation of the results obtained, give insights on what is happening and how to further improve the scores.

- Analyze the practical utility to help with prognostic prediction, improve patient care and increase survival time.

1.3 Structure of the document

This document is organised as follows:

- Chapter 1 - Provides an introduction, explains the goals and contributions of this work and introduce the ALS case study.
- Chapter 2 - Gives an overview of Deep Learning approaches in ALS and insights of work that has already been done related to ALS.
- Chapter 3 - Is an overview of the data used to describe each ALS patient.
- Chapter 4 - Discusses the main deep learning approaches we used for this work.
- Chapter 5 - Presents and discusses the results obtained by the models proposed.
- Chapter 6 - Presents the conclusions and outlines future work and goals.

Chapter 2

Background and Related Work

2.1 Amyotrophic Lateral Sclerosis

Amyotrophic Lateral Sclerosis (ALS) is a neurodegenerative disease, that consists of rapidly progressing muscular weakness. ALS patients have a short life expectancy, and the most common cause of death being respiratory failure, given that it eventually affects most muscles, including the diaphragm [45]. ALS affects around 10 per 100.000 inhabitants, in Portugal, and worldwide between 5.9 and 39 people per 100.000 inhabitants [38], suffer of ALS.

Most treatments focus on managing symptoms, because this disease has no cure. So improving quality of life and increasing the life expectancy is the main objective of clinicians at the moment. Non-invasive Ventilation (NIV) has shown to increase the life expectancy of patients, however not all patients are eligible to start NIV treatment.

It is hard to predict when a patient should be treated with NIV, given that there is still some discordance to when the optimal time is to start the treatment, mainly because it is generally accepted that it should be started when clinical symptoms of respiratory insufficiency are evident. However, recent studies show that an earlier initiation of NIV might have greater benefits [12].

The Revised ALS Functional Rating Scale (ALSFRS-R) is the temporal information available, and consists on the outcomes of a series of clinical assessments of specific tests, from 1093 ALS patients observed from the early 90's until October 2021, in the form of multivariate time series. Consisting on a set of tests that are recommended to the patients to perform, they then can evaluate on a scale from 0 to 4, based on the difficulty of performing certain tasks. The score of 0 means that the patient cannot perform the task, and a 4 that the patient can easily perform the task assigned. In clinical practice, the ALSFRS-R is widely used to assess disease progression. Despite this scale, there is

a need for advanced machine learning approaches for clinicians to be able to use these models to aid in prognostic prediction and improve life expectancy, given the challenge of disease heterogeneity. However, there is still difficulties predicting disease progression of ALS, given the quantity of data being rather small and highly imbalanced proves to be a great challenge.

2.2 Deep Learning

In this section, we discuss the use of deep neural networks approaches on tabular data and the different types of architectures approaches that can further improve the performance of the models. We will provide an overview of architecture-based methods and regularization-based models.

2.2.1 Preprocessing

In this section, we discuss the preprocessing of Deep Learning on Tabular Data. Despite traditional machine learning approaches being generally better with tabular data and deep learning models were usually used with audio or image data. However, it is still not clear why deep learning cannot achieve the same level of predictability on tabular data, as on image classification and natural language processing.

There are several challenges when working with this type of data, such as inappropriate training data with a lot of missing values, extensive preprocessing pipelines and model sensitivity. The quality of the data is a common issue for real-world data [5]. These often include missing values, outliers, inconsistent data, and tabular data are frequently class-imbalanced, due to the expensive nature of data collection.

To deal with the class-imbalance, there are two main approaches. The first, over-sampling, this approach adds copies of instances from the minority class. Increasing the number of instances of the minority class, by replicating them until you have a more balanced class, Random Over-sampling is an example of this approach. And under-sampling, this approach deletes instances from the majority class, scaling down the number of instances of the majority class, Random Under-sampling is an example of this approach.

However, both of the approaches have their pros and cons that one should consider when dealing with this type of problem. Over-sampling has the advantage that it does not lead to information loss, given that it creates new data, but this increases the likelihood of model overfitting, because it replicates the minority class. Under-sampling has the advantage that it can help improve the run-time of the model and memory problems, when a data set is quite large, however, unlike the over-sampling it can eliminate useful

information about the data.

SMOTE (Synthetic Minority Over-sampling Technique) [8] is a technique proposed to solve the previous approaches problems. This approach is an over-sampling method, that creates synthetic data, instead of replicating the instances of the minority class. It aims at generating similar samples of the minority class, by taking into account each minority class sample and creating new samples using K-nearest Neighbours (KNN). This approach alleviates the overfitting caused by the random oversampling, given it generates new data instead of replicating instances. It does not lead to loss of information, because there is no deletion of instances of the majority class. However, SMOTE can generate additional noise to the data, and is not as practical for high dimensional data.

One of the main challenges when working with this type of data, is the extensive preprocessing needed, mainly on how to handle categorical features. Generally the first step is to convert the categories into a numerical representation, these may be quite sparse leading to a problem known as curse of dimensionality [19].

Regarding model sensitivity, deep neural networks can be particularly vulnerable to little changes in the input data [37]. Even the tiniest change in a both categorical and binary features can have a significant influence on prediction [5].

2.2.2 Specialized Architectures

The most common strategy for deep learning with tabular data is specialized architectures. With this we will provide an overview of state of the art deep neural network architectures, specifically for heterogeneous tabular data, such as the data used for this project. This group can be further divided into two sub-groups, the first being hybrid models and the second being transformer-based models.

The majority of deep neural network techniques for tabular data are hybrid models. They adapt the data and combine neural networks with proven classical machine learning algorithms, such as decision trees. In this we have two types that we will distinguish, the fully differentiable models and partly differentiable models.

This category's fully differentiable models have a unique feature, they use gradient descent optimizers to provide end-to-end deep learning for training and inference. As a result, they enable highly efficient implementations in recent deep learning frameworks that make extensive use of GPU or TPU acceleration. We will present some models that we found interesting to mention on this topic.

First we have an ensemble of differentiable oblivious decision trees [25] proposed by Popov et al. [33] - the NODE Framework for deep learning tabular data. Because all nodes on the same level utilize the same splitting function, oblivious decision trees can be easily parallelized. NODE uses the transformation with soft splits to make the entire architecture fully differentiable and benefit from end-to-end optimization.

Katzir *et al.* [21] introduced an interesting idea, which every decision tree is essentially a form of a Boolean formula, more precisely a disjunctive normal form, as Net-DNF recognizes. They employ this inductive bias to construct the architecture of a neural network that can mimic the gradient boosting decision trees algorithm's properties. It was then tested for classification tasks on data sets with no missing values and produced results that were similar to those of XGBoost [9].

Nondifferentiable techniques and deep neural networks are combined in this subset of hybrid models. The nondifferentiable element of these models is usually handled by decision trees.

Ke *et al.*[22] proposed the TabNN architecture, which is built on two principles: directly using expressive feature pairings and lowering model complexity. It extracts feature groups from gradient boosting decision trees, clusters them, and then builds a neural network based on those feature combinations. However, numerous intensive computation stages are already involved in the network's creation [5].

Similar to the TabNN, it was proposed the use of gradient boosting decision trees for the data preprocessing stage by Ivanov and Prokhorenkova [20]. The authors demonstrate that a decision tree has the shape of a directed graph. As a result, the proposed approach uses graph neural networks to utilize topology information from decision trees. Several experiments showed that the proposed architecture was superior to other solid proposals, regarding predictive performance and training time.

The other type of specialized architectures, is the transformer based models. Researchers and practitioners have proposed numerous ways for heterogeneous tabular data employing deep attention processes [5], inspired by the recent explosion of interest in transformer-based methods and their results on text and visual data.

Cai *et al.* [6] introduced ARM-net, this is a tabular data-specific adaptive neural network for relation modeling. The ARM-net framework's main idea is to dynamically and selectively simulate feature interactions with combined features. In addition, the authors offer a novel sparse attention approach for dynamically generating interaction weights given input data. As a result, users can directly model feature crosses of any order with

noisy features that have been judiciously filtered.

2.2.3 Regularization Models

This group of approaches claims that the extreme flexibility of deep learning models for tabular data is one of the key learning challenges, and that substantial regularization of learnt parameters might improve overall performance [5].

The Regularization Learning Network (RLN) introduced by Shavitt and Segal [36], which uses a learnt regularization approach, was one of the first methods in this area. In their tests, RLN's outperformed deep neural networks and produced results comparable to the gradient boosting decision trees approach, but they only compared the models using one data set containing mostly numerical data. The categorical data concerns are not addressed in the RLN study. Data sets with solely numerical data were used for the studies and example implementation. Borisov et al. [4] proposed a similar approach, in which regularization coefficients are learned exclusively in the first layer with the purpose of extracting feature importance. We found out that using the right regularization techniques is fundamental to train these models and give all the details in the following chapters.

2.2.4 Activation Functions

These type of functions are used to establish the neural network's output, such as yes or no. The obtained values are mapped between 0 and 1 or -1 and 1. The Activation functions are divided in 2 types, Linear and Non-Linear activation functions. The activation function used on our models in this work, is the Rectified Linear Unit (ReLU) activation function, this is a non-linear function. The output values are between 0 and 1, given the nature of this function, all negative values become zero. This method provide some advantages, such as simpler computation which reduces computation run times.

2.3 Related Work

In this section we focus on machine learning and deep learning methods applied to the ALS domain. There have been a lot of studies regarding ALS, where many researchers developed machine learning models in order to predict the diagnosis. That is whether or not the patient has ALS or some other condition, given that is not uncommon for misdiagnosis. There has been some promising proposals made by ALS researchers, despite the limited size data [17].

To achieve that, there are two main approaches, supervised and unsupervised learning. Based on the features of the available data and the overall study purpose, the specific

method should be carefully chosen.

The goal of supervised learning is to map inputs to outputs using training data sets. Problems regarding supervised learning can be divided into either classification or regression problems [35]. Classification approaches use the input of the training data sets, and the output variable is a category, such as "Disease Progression" or "No Disease Progression". On the other hand, regression approaches use the input, to predict a real value of a variable, such as trying to predict motor decline based on clinical observations [17].

The goal of unsupervised learning is to learn the structure of data without a well-defined output or feedback [35]. Being extremely helpful in patient stratification problems. In order to create clusters, this approach is used, current clinical criteria, which are generally based on a limited collection of clinical observations, tight thresholds, and conservative inclusion/exclusion criteria for class membership, may be preferable than clustering approaches [17].

Bereman *et al.* [2], investigated protein bio markers in a set of matched plasma and CSF (Cerebrospinal fluid) samples from patients with ALS. To discover differentially abundant proteins, intensity-based relative quantification was performed, followed by evaluation of advanced machine learning techniques to construct both diagnostic and prognostic models for application in ALS.

Zhou *et al.* [44] to investigate the utility of predictive modeling on clinical trial analysis of ALS. The approach proposed for ALS disease progression was measured by ALSFRS-R in a test set from a former clinical trial. The models used were able to achieve some promising results, for the trial sample. Reaching the conclusion that ALSFRS-R are capable of explaining a moderate degree of longitudinal change variability, which is aided by robust missing data management for baseline parameters.

Carreiro *et al.* [7] proposed the first prognostic models based on predefined time windows to predict the disease progression and the need of NIV treatment in ALS. This approach consisted in clustering temporally-related tests using hierarchical clustering with constraints, in order to obtain snapshots of the patients' condition at a given time, then used to predict whether or not the patient will require NIV within a time window.

Pires *et al.* [32] proceeded to categorize patients depending on their stage of disease progression, stratifying them in 3 groups, slow, neutral and fast. Using the patients' disease progression rate to create the previous groups. Afterwards they used the data from each of the groups and build specialized models, with the goal to improve prognostic prediction. However neither of these approaches take into consideration the temporal de-

pendence between the available features.

Martins *et al.* [29] has also made progress regarding disease presentation patterns and disease progression patterns, where the original features of the dataset available were transformed, in order to use as sequences for the algorithms used on the SPMF library, and these outputs the extracted patterns. Following the evaluation of the extracted patterns, the resulting findings showed to be very promising, with the models for all three prediction time windows.

Regarding deep learning approaches, Muller *et al.* [30] made use of neural networks, mainly the Long Short-Term Memory (LSTM), to predict the disease progression of the patients in the previously mentioned time windows and Shapley Additive Explanations (SHAP) to explain complex machine learning models. Finding that the use of SHAP explaining the deep model was helpful to determine the influence of certain features for the overall prediction. Our work is most similar to [30], which inspired the deep learning approach and preprocessing steps.

Van der Burgh *et al.* [42] use deep learning, with clinical characteristics as well as MRI data to predict survivability of ALS patients. In close to 70% of patients, clinical features successfully predicted survival category. Deep learning utilizing MRI data properly predicted 62.5% of structural connections and 62.5% of brain morphological data. But when combining all the sources of information, the deep models prediction accuracy increased to almost 85%. Demonstrating a promising advantage of the use of deep learning model in disease prognostication.

Chapter 3

Data Pipeline

3.1 Dataset

The datasets used in this work consist of ALS patients after being diagnosed with ALS and receiving therapy in the ALS clinic of Centro Hospitalar Universitário de Lisboa Norte, Hospital Santa Maria, from 1995 to 2021. The data consists on static information, containing demographic information about the patient, and temporal information that consists of the results of a series of clinical assessments of specific tests, these include respiratory tests, neurophysiological data, and questionnaire answers to the ALS Functional Rating Scale. The tests are recommended for the patient to perform, which they can proceed to evaluate on a scale from 0 to 4, based on the difficulty of performing certain tasks, where a 0 means that the patient cannot perform the task, and a 4 means that the patient can easily perform the task assigned. Table 3.1 shows the tests mentioned.

The main objective includes being able to predict within a predefined time window, whether or not a patient will need NIV. These time windows are within 90, 180 and 365 days. For that the main dataset is also divided into 3 sub-datasets for each time window. However, ALS also affects other areas of the body. Many patients might have difficulty walking or talking, so those are also clinical endpoints that are interesting to predict and further improve the quality of life. Therefore there are other possible datasets for the different clinical endpoints available aside from the NIV, such as, to predict the need of an additional communication tool, of a full-time caretaker, of a wheelchair or the indication of PEG.

3.2 Data Acquisition

As mentioned before the data was acquired of the ALS patients receiving therapy in the ALS clinic of Centro Hospitalar Universitário de Lisboa Norte, Hospital Santa Maria,

Category	**Test**	**Abbreviation**
Respiratory Tests	Vital Capacity	VC
	Forced Vital Capacity	FVC
	Airflow Occlusion Pressure	P0.1
	Maximal Sniff Nasal Inspiratory Pressure	SNIP
	Maximal Inspiratory Pressure	MIP
	Maximal Exspiratory Pressure	MEP
Neurophysiological Tests	Phrenic Nerve Response Amplitude	PhrenMeanAmpl
	Phrenic Nerve Response Latency	PhrenMeanLat
Other Physical Values	Cervical Extension	CervicalExt
	Cervical Flexion	CervicalFlex
ALSFRS-R Questions	1- Speech	P1
	2- Salivation	P2
	3- Swallowing	P3
	4- Handwriting	P4
	5- Cutting Food and Handling	P5
	6- Dressing and Hygiene	P6
	7- Turning Bed and Adjusting Bed Sheets	P7
	8- Walking	P8
	9- Climbing Stairs	P9
	10- Breathing	P10

Table 3.1: Overview of the Several Tests. This presents the temporal information of the data, that consists of the results of a series of clinical assessments of specific tests, they consist of respiratory tests, neurophysiological information, and ALS Functional Rating Scale questionnaire responses. On the ALSFRS-R, are the recommended tests for the patients to perform and self evaluate their performance on a scale from 0 to 4, based on the difficulty on performing the tests. Self evaluating with a 0 means that the patient is not able to perform the said test, and a 4 means that the patient can easily perform it.

from 1995 to 2021. However the data had to go through a processing approach, that was proposed by Carreiro *et al.* [7]. In order to build the snapshots from this type of clinical time series, the majority of the researchers follow a simple approach, which consists on the use of a pivot date, usually the day of a crucial event or test. Every evaluation of a test that was conducted between two pivot dates is sorted into the cluster that contains the pivot date that is furthest to the left, one example would be the date of hospitalization. Therefore the following test after this date is always included in the snapshot. However, this approach present several limitations, which a patient's important test may not have been evaluated, or it may have been performed more than once between two pivot dates. This would result in the discarding of a large number of test evaluations or the creation of sparser snapshots (with more missing values) [7].

In order to overcome these limitations Carreiro *et al.* [7] proposed a new way to create the patient snapshots based on clustering temporally-related tests using hierarchical clustering with constraints (using single-linkage metric) in order to obtain snapshots of the patients' condition at a given time, the same test cannot be evaluated twice because it is a component of two independent batches of tests, and all tests in a snapshot must be coherent with respect to a particular property of interest. As a result, the NIV status of each test in a given patient snapshot must be the same, given that it is known at the time of each test if the patient required NIV or not. The method proceeds to compute a single feature representing the NIV status for the snapshots, which can have two values 1 meaning the requirement of NIV, and 0 meaning no requirement of NIV.

It is required to build the learning instances that the prediction models will employ after creating the patient snapshots using the original follow-up data. These depend on changes in NIV status between snapshots within the k-day time range that has been selected. And afterwards, they created the class Evolution (E), with two possible values, 1 the patient needs to initiate treatment within the time window and 0 the patient does not need any treatment.

In figure 3.1, we can see the transformation performed on the original data into the snapshots of each patient used on this work which are divided by the reference of each patient, meaning each instance is a different evaluation performed, the instances go up to the point where a patient has a positive evolution, and does not go beyond that point.

3.3 Target Indicators

The tests performed by the patients are indicators for the doctors to whether or not to treat a patient for a certain clinical problem. For the several clinical endpoints, there are different tests used as indicators as we will further explain for each of the clinical

Original Data

REF	Date	Test 1	Date N	Test 2	Test 3	…	Test N	NIV Date
1	01/01/2010	13	06/06/2010	3	28		4	09/11/2010
2	28/05/2010	15	23/12/2010	7	25		2	Not Applied
…								
N	30/07/2018	17	25/01/2019	4	27		3	17/02/2020

Snapshots

REF	Date (Snapshot Median)	Test 1	Test 2	Test 3	…	Test N	Evolution
1	01/01/2010	13	3	28		2	0
1	28/05/2010	15	5	23		3	1
…							
N	13/05/2017	23	4	12		4	0
N	30/07/2018	17	3	27		3	1

Figure 3.1: Example of data transformation, from the original data into the patient snapshots used on this work, in this case the target clinical endpoint was the NIV.

endpoints.

For the clinical endpoint C1 that is the need for NIV, it is detailed that before the usage of the ALS Functional Rating Scale Revised (ALSFRS-R), it was used the ALSFRS, and it had the P10 test, which tested the patient's respiration. Meaning that older instances of patients, only have the ALSFRS, and for these whenever the value of the test is below 2 are considered eligible to start NIV. However on more recent patients, it is used the ALSFRS-R, in which the P10 respiration test is not applied and there are 3 new instances for testing the respiration of the patient. Therefore the P10 test is the existence of dyspnea on the patience, the P11 test is the existence of orthopnea, and the P12 is whether there is respiratory insufficiency. In this case, the patient is eligible to start the NIV treatment, when the P10 test value is 1 (dyspnea occurs while in rest, when sitted or laying) or 0, or when the P12 test value is below 3 (Intermittent use of BiPAP).

On the clinical endpoint C2 that is the need for an auxiliary communication device, the

test remained the same as previous scales, being the P1 speech test, that indicates how the patient communicates. Therefore the patient is eligible to use an auxiliary communication device, when the P1 test value is 1 (Speech combined with non-verbal communication) or 0 (Loss of useful speech).

The clinical endpoint C3 is the Percutaneous Endoscopic Gastrostomy (PEG) indication, has the C2 the test remained the same, the test being the P3 Deglutition test. Given that this treatment is quite invasive a lot of patients with PEG indication end up stalling the procedure, therefore the timing to start PEG is when the P3 value test is 1 (Need for probe for supplements) or 0 (Enteral or parenteral nutrition).

On the clinical endpoint C4 that is the need for a caregiver, there are 2 tests to consider that remained the same, the first being test P5 which consists of cutting food and managing objects and the second is test P6 which is getting dressed and personal hygiene. On the first test, there are 2 variants one for patients with gastrostomy and one for those that do not have gastrostomy, in both the value that is considered to have a caregiver is below 2 (Needs some help doing the activities). On the second test, the value decided to have a caregiver is 1 (Needs help for the personal hygiene) or 0 (Total dependence).

Lastly, the clinical endpoint C5 is the need for a wheelchair, the main testing metric is the walking test. On this one if the patient can not walk even with help it is determined that is time to start using a wheelchair, therefore if the test value is 1 (Functional movement only, no outpatient) or 0 (No useful limb movements inferior).

3.4 Preprocessing

3.4.1 Data Imputation

With that, the dataset was missing values in most of the columns, so data imputation was crucial, given the size of the dataset we tried to remove as few rows as possible. There are several ways why certain values are missing from the data, as such there are three different types of missing values: Missing Completely at Random (MCAR), Missing at Random (MAR) and Missing Not at Random (MNAR) [10], described below:

- **MCAR:** Data gaps have underlying causes that are independent of both known and unknowable patient factors. An illustration is when a blood sample tray breaks unintentionally and the lab results are lost.
- **MAR:** The fundamental causes of missing data are connected to well-known patient features. The greater likelihood that a blood sample could not be taken prior to the

commencement of dialysis in patients who started acutely is one example.

- **MNAR:** The fundamental causes of missing data are connected to unidentified patient features. As an illustration, participants with lower levels of education are more likely to leave out a question in a survey on their schooling.

To address this problem, there are several approaches we can use, the three main approaches are the use of Deletion, Imputation or Prediction [10], described as follows.

- **Deletion:** This consists in removing single instances or columns that are formed mostly by missing values, that are above a threshold defined by the user. However this method often leads to a considerable loss of information that could be essential for the model learning.
- **Imputation:** This process consists on replacing the missing data with values that are plausible by observing other instances of the whole data. The most common methods are the Mean Imputation and the Zero imputation, however these methods often lead to dataset bias [46], that would greatly affect the model performance and make it so the model would not be able to perform on a real world application. And there are methods, such as the KNN Imputation that uses the nearest neighbours to replace the missing values with data that are similar.
- **Prediction:** This method consists on using predictive models that learned from the data in order to predict the missing values. This type of process can be used with deep learning or regression models, however these are really complex to implement and are not always advantageous if one does not have enough time to spend training these models.

3.4.2 K-Nearest Neighbours Imputation

Our approach to solve this, was to use a model to predict the missing values, the model used in this case was the K-Nearest Neighbours Imputation (KNN Imputation). KNN algorithm has been proven to be quite effective [3], however we only used it on features that had less than a predefined threshold of values missing, that is specific to the datasets of each clinical endpoint. The features with a number of missing values above the defined threshold, were removed from the dataset.

However we would have to pay attention about choosing a low k value will make the results less generalizable and increase the influence of noise [31]. As opposed to that, choosing a higher k tends to make local effects less noticeable [31]. Another For that we would have to choose somewhere around the middle. There is also a need to pay attention

to whether or not we have binary classes, since it is recommended to use an odd number of neighbours to avoid ties [31].

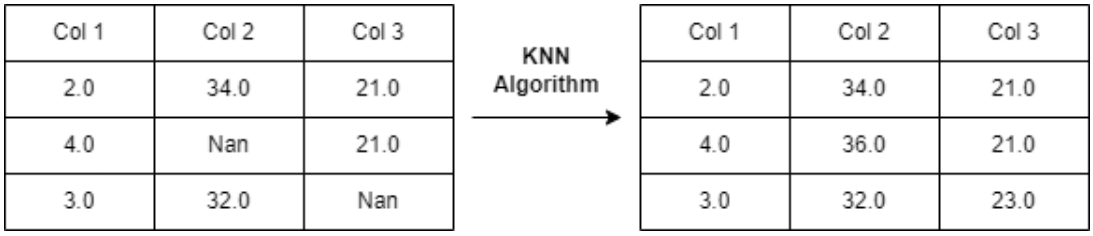

Figure 3.2: Example of K-Nearest Neighbours Imputation. This algorithm predicts the missing values by using the nearest neighboured instances to replace the missing values with data which is similar.

As shown on figure 3.2, this was one of many approaches possible to tackle this problem, however some of the other approaches result in modifying the variance of the dataset, such as the Mean Imputation and the Zero Imputation, which would turn the dataset bias to some degree [46]. Another approach that would greatly affect the performance would be to simply remove the rows that contain missing data, but given that the data available is already limited, to further reduce the amount of data which we could feed to the models was not a option.

3.4.3 Label Encoder

In order to be able to use the KNN Imputation algorithm and neural networks for prediction, there was a need to change the categorical features so they could be used on the algorithm. For that, we used Label Encoder, which consists on encoding the levels of categorical features and transform them into numeric values, given that these algorithms do not work unless we use numerical values.

3.4.4 Class Imbalance

The next step was dealing with the class imbalance, given that some of the datasets we have available are highly imbalanced, meaning that classes of data are not equally represented. This represents a problem for most classification algorithms, the reason why, is these aim at maximizing classification accuracy, this measure is biased towards the majority [27].Without dealing with this problem, models could achieve high values of accuracy, despite not being able to generalize the predictions made.

There are two main approaches to address the class disparity problem. The oversampling methods add duplicate instances from the minority class. Random Over-sampling is

an example of this method, which involves increasing the number of examples of the minority class by repeating them until you have a more balanced class.And under-sampling, which reduces the number of instances of the majority class by deleting examples from the majority class, an example of this strategy is Random Under-sampling.

To solve this problem, there are several approaches as previously mentioned, but many have the risk of overfitting the model, for that the approach chosen in this case is SMOTE (Synthetic Minority Over-sampling Technique). This approach, avoids the risk of overfitting, by generating artificial samples instead of replicating existing observations [13], but it is recommended to use SMOTE combined with Random Undersampling of the majority class, as it is believed to achieve better model performance [8]. On table 3.2, we can see the class distribution of the datasets we are using before any class resampling.

Dataset	Evolution	k=90	k=180	k=365
C1 - Need For NIV	Yes	217	740	1394
	No	4178	3655	3001
C2 - Need for an auxiliary Communication Device	Yes	159	454	859
	No	5237	4942	4537
C3 - Need For PEG	Yes	68	267	605
	No	5983	5784	5446
C4 - Need for a Caregiver	Yes	341	837	1437
	No	2750	2254	1654
C5 - Need for a Wheelchair	Yes	202	589	1242
	No	4928	4541	3888

Table 3.2: Class Distribution For Time Windows of k Days. The class imbalance tends to be at its' highest when the time window is shorter, such as the 90 day time window, and as the time window enlarges to 180 and 365 days, the class imbalance decreases.

Checking the previous table we can see that the class imbalance, tends to be at its highest when dealing with the shorter prediction time window (90 days), and tends to decrease this disparity as the time windows grow larger. This is a trend that every dataset follows, and is something to be expected given that in a larger time window there is more likelihood of a patient to have a positive disease progression.

3.4.5 Feature Selection

After that we proceed to do feature selection which the goal is to find the best set of features, to achieve a better prediction model. Evaluating the features that will give the most information to make a good prediction of the target class. There are several approaches

one could use, in order to, obtain a set of features that can explain the dataset without using all features available.

We decided to use the Maximum Relevance - Minimum Redundancy (MRMR) algorithm, in order to achieve that. The MRMR is an algorithm used for finding the ”minimal-optimal” subset of features, this finds the features that are relevant with the least redundancy to explain the data available, as is shown on the figure 3.3.

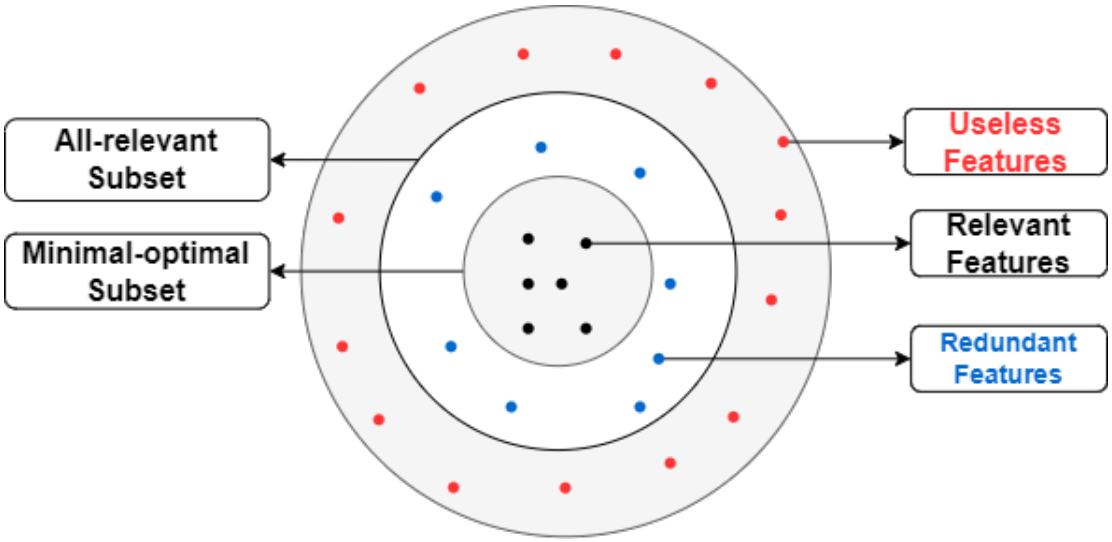

Figure 3.3: Example of Maximum Relevance - Minimum Redundancy. This algorithm finds the ”minimal-optimal” subset of features which prove to be relevant with the least redundancy to describe the data available.

We tried several values for the number of features to use on the models, after choosing a value we then trained the models and evaluated them in order to, reach a conclusion to what number of features we could achieve the best performing model. Reaching the consensus that the number of features chosen does not impact greatly the performance of the proposed models, these achieved the best results in most of the cases when paired with 15 features as inputs. However the performance increase is not enough to justify the use of this number of features and not only 3 to 5 features, but as this does not decrease the overall computational run time, so we decided for now to keep using 15 features as input.

3.4.6 Selected Features

In this section we will give insight of the features selected by the MRMR algorithm. As mentioned previously on section 3.1, there are tests used as indicators, to help clinicians know when a patient should start a given treatment. And the MRMR algorithm was able to find the features that were originally used as indicators in the great majority of the datasets. With this said, the tests used as indicators are in the top 5 features in terms of

relevance, in all most all of the datasets. This shows that the algorithm chosen for feature selection is able to find the relationship between the test used as indicator and the target value.

Chapter 4

Methodology

This section describes the methodology proposed for the prediction of disease progression of clinical endpoints in ALS. As previously mentioned, ALS is a disease characterized by the progressive loss of motor neurons in the brain and spinal cord. Patients with ALS tend to have a short life expectancy, being the most common cause of death respiratory failure. Given that most muscles are eventually affected by this disease, the patients with ALS, suffer from various conditions that need treatment. There is no cure for ALS, so most treatments focus on managing symptoms, and the main objective of clinicians is to improve quality of life and increase life expectancy. The goal is predict ALS disease progression, in order to know when a patient should receive treatment for a given clinical endpoint, within a predefined time window.

4.1 Architecture Proposal

There are many approaches that could deal with the problem at hand, for this work we proposed the use of deep learning, in order to come up with deep models that are capable of predicting disease progression of clinical endpoints in ALS. The same deep learning model can be applied to various different applications and data types, and is flexible enough to be adapted to new problems in the future. Despite the need of a large amount of data, in order to achieve a better performance than other approaches, there has been promising results with using neural networks.

Our models consist of somewhat complex neural networks. The first is a Multi Layer Perceptron (MLP), and a Convolutional Neural Network (CNN) based on a previous approach winner of second place on Mechanisms of Action (MoA) prediction competition and used on the Santander customer transaction prediction problem [39]. The last model developed is a Long-Short Term Memory (LSTM), which takes advantage of the temporal clinical data in the form of sequences, being a somewhat simple architecture.

4.2 Multilayer Perceptron

Multilayer Perceptrons are the classical type of neural network, feedforward artificial neural network that generates a set of outputs from a set of inputs. One or more layers of neurons make up these structures. These use backpropagation for training the network, this is a method of propagating the entire loss back into the neural network to determine how much of the loss is caused by each node, and updating the weight values of the nodes with the higher error rates, in order to minimize the overall loss. The technique propagates an output error from a neural network backwards through the network to identify the routes that have the biggest effects on the output [34].

Backpropagation reveals which pathways have a greater impact on the outcome and enables us to strengthen or weaken connections to make the prediction we want. MLP's are well suited to classification prediction problems in which inputs are classified or labeled, and are used for tabular data sets as is the case of the ALS data set.

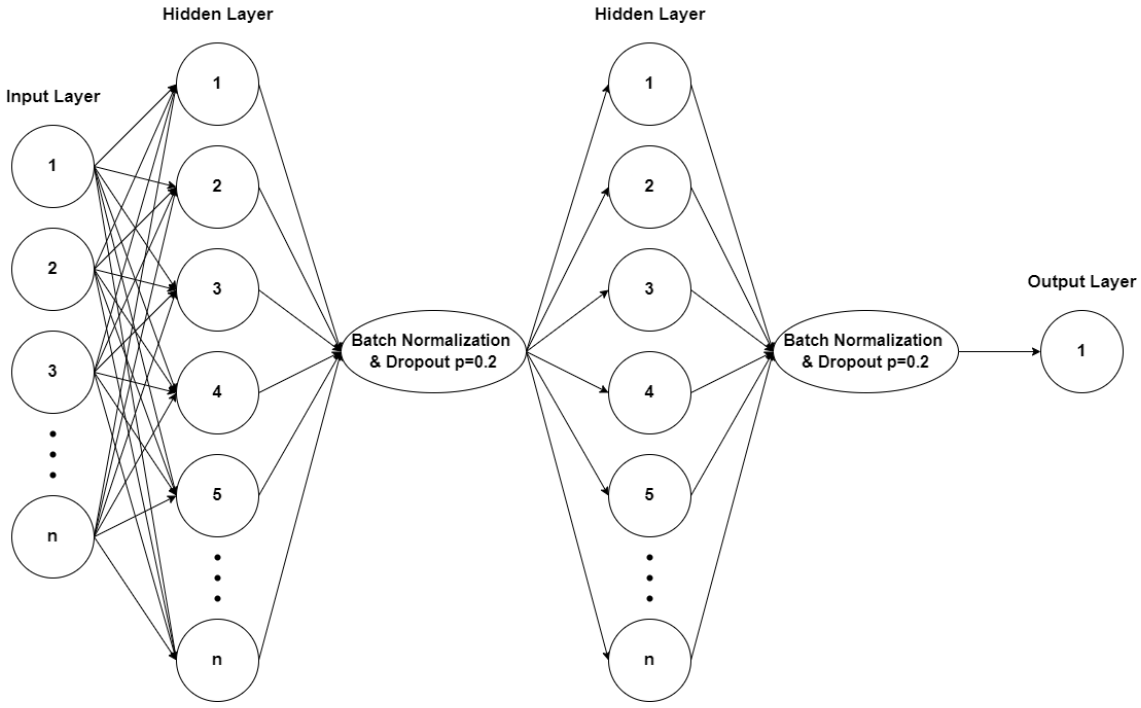

Figure 4.1: Multilayer Perceptron Proposed Architecture. This proposed architecture consists on a sequence of input neurons, into its' 3 layers. The input layer uses **n** features, and passes those to the other two hidden layers with 512 features of the batch normalization and followed by a dropout layer of 20%, both use the ReLU activation function, finally reaching the output layer, that is a continuous value between 0 and 1. Figure adapted from []

The proposed architecture for this model is shown on the figure 4.1, it consists on a sequence of input neurons as features, into its 3 layers. The first layer is the input layer and consists of the input neurons and a ReLU activation function, after this layer we have the

batch normalization layer with 512 features, followed by a dropout layer of 20%, finally connecting with the output layer. The output of this MLP is a continuous value between 0 and 1, where a 0 indicates the patient does not require the treatment, and 1 when the patient requires the treatment, for the specific clinical endpoint its trying to predict.

4.3 Convolutional Neural Networks

Convolutional Neural Networks, or more commonly known as CNN or ConvNet, are a class of neural networks that is dedicated to the processing of data with a grid-like architecture, such as images. Normally CNN's consist of three layers: a convolutional layers, a pooling layer and a fully connected layer. Although the CNN structure is effective at extracting features, it is rarely employed in tabular data since the precise feature ordering is unclear. The data can be directly reshaped into a multi-channel picture format, and the correct sorting can be learned using the FC layer through back propagation.

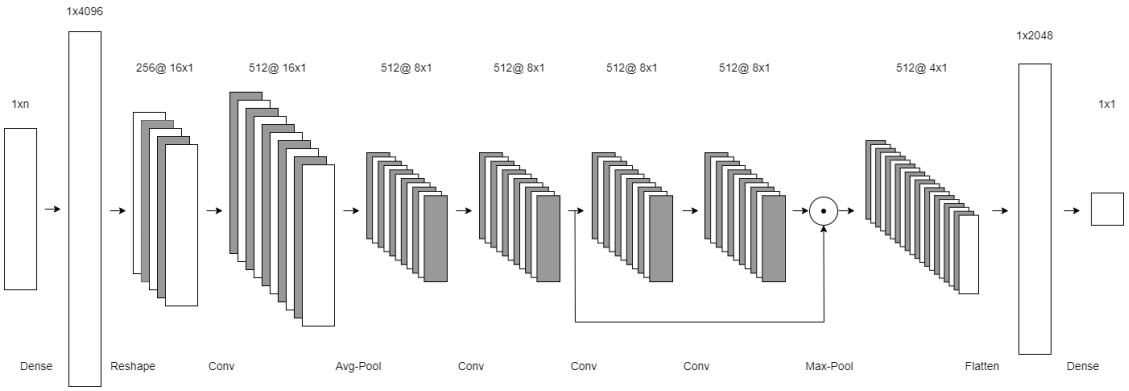

Figure 4.2: 1 Dimension Convolutional Neural Network Architecture. The FC layer improves the feature dimension, which expands the input's dimension and makes the resulting data meaningful through feature sorting. Afterwards there is a reshaping of the data into an image format, then features are extracted on the oncoming 1D-Conv layers. Finally they are used to make predictions trough a FC layer after flatten. Figured adapted from [39]

The architecture shown on the figure 4.2, first we have a FC layer is used to improve the feature dimension. This layer's responsibilities include expanding the input's dimension and making the resulting data meaningful through feature sorting. After this, there is the reshaping of the data into an image format. Features are extracted in the next multiple 1D-Conv layers with a shortcut-like link, just like in the basic CNN model. Finally, the features that were extracted are used to make target predictions through a FC layer after flatten [39].

4.3.1 Training

In order to train the models proposed more efficiently, there was a need to rescale the data, through data normalization on all features. This is a rescaling of data from its original range so that all values fall between 0 and 1. This is recommended when dealing with neural network problems, as it often results in a better performing model, and leads to a more efficient neural network training.

Furthermore each dataset was imputed to the models, while using stratified group k-fold cross-validation (CV) scheme [23]. Cross-validation is a resampling procedure, widely used on machine learning model evaluation, on data sets with limited samples. This method only uses the parameter k, that represents the number of groups the data provided is to be split into. In this case we used the parameter k equals to 5 and 10, therefore the data set provided to train the model, will then be split into k groups, then it will select one group as a test set to evaluate the model, and the rest of the groups are used as training data, and it does this k times until it used all groups as test sets in evaluation. It is important to notice that this algorithm is a variation of the Stratified k-fold cross validation, as it attempts to return a stratified fold, with no overlapping instances in the train and validation sets at the same time. Ideally, we would have used the recommended approach [18] of using the stratified 10-fold CV, given that it is relatively low bias and variance. However in some circumstances, stratification will be impossible due to a small number of groups containing a large number of samples.

In order to prevent such event from happening, it was required to do a search over the several different folds produced by the CV, to find the ones where this would not happen. To achieve this, we tried several random state values on the CV, and analysed whether or not the groups formed were stratified. Saving the values which the groups were indeed stratified, if not it would generate problems when trying to evaluate the models, because it would not be able to calculate the AUC-ROC metric if there were only instances of a single class present on the validation set. However the most imbalanced datasets, did not have enough positive class instances to produce stratified groups, and only produced groups that the validation set just had one class present. This proved to be quite challenging, so the decision was to put aside any dataset that could not produce stratified groups.

This will help to get an idea of how the model would behave on real data, without having bias on our train and validation sets.

The resampling is only applied on the training set of each fold, leaving the test set unchanged and these sets created have the same class distribution as the original data set. Therefore it was confirmed that the cross-validation method does not result in training set contamination by the test set or vice versa, meaning no patient appears in the train and test sets at the same time.

Figure 4.3: A K fold example. This approach will split the data provided into k groups, defined by the user, afterwards it will select one group as a test set and the remaining groups are used as training sets, and repeats this process until every group has been used as a test set in evaluation.

Given that these type of models require a lot of training, in order to achieve good results, the models train for a maximum of 500 epochs. But not all models end up training to that amount of epochs, because some models reach a plateau sooner than others. To prevent training models that are not improving anymore, we decided to use Early Stopping to prevent pointless training and consumption of resources. In most cases the value chosen for this parameter was between 50 and 80 epochs of training, meaning if the model has stopped improving its loss value for the specified number of epochs, it stops training that model

4.3.2 Loss Function

The main objective when training neural networks is to decrease the value of the loss function, the reason why is this metric focus on the minimizing the error the model makes. The loss function aggregates the various good and bad aspects to a single number. So when this value decreases it is a good representation that the model is improving it's performance.

There are many loss functions one can choose, but for the context of the ALS disease progression problem, as we are trying to predict between two classes, it is a binary clas-

sification problem as previously mentioned. So the better option of loss function for this is the Binary Cross-Entropy (also referred as Logarithmic Loss), and the configuration of the output layer of a node with a sigmoid activation unit.

4.4 Long-Short Term Memory

Long Short-Term Memory (LSTM) networks are a special sort of recurrent neural network (RNN) that can learn order dependence in sequence prediction problems. Traditionally, recurrent neural networks have been difficult to train. However LSTM's are not generally appropriate for tabular data sets, where results have been poor. But they can be used on time series data, as is the case of the ALS data set, so for that reason we will try this type of model to evaluate whether it is a good approach.

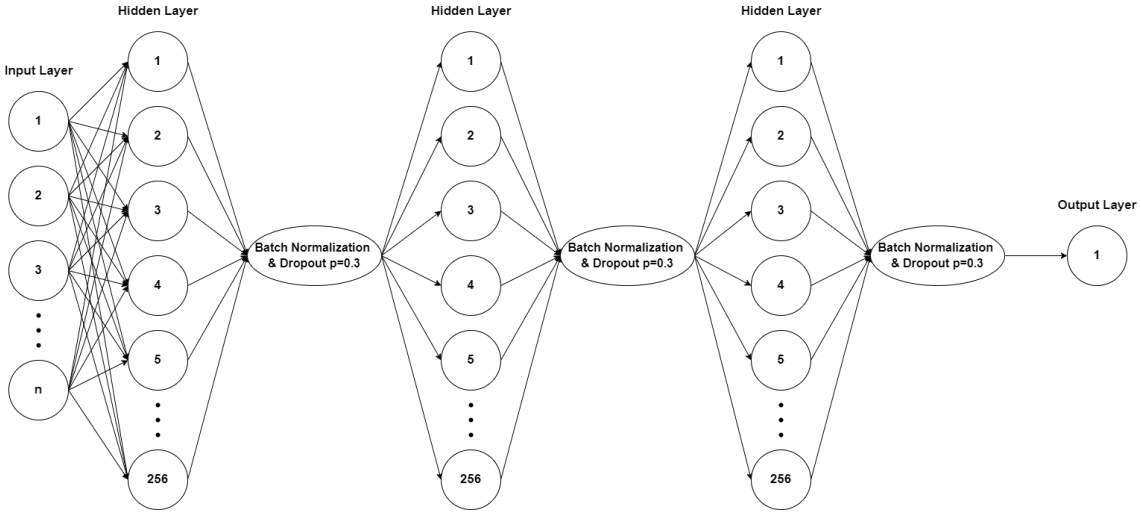

Figure 4.4: Sequence Model - LSTM Architecture. This approach makes use of the temporal dependence of the clinical features, which the input layer feeds the data into it's hidden layers, that are composed of 256 neurons, and use batch normalization and a dropout layer of 30%. Finally the output layer consists of 1 neuron that is a continuous value between 0 and 1.

This approach will use the temporal dependence of the clinical features, and for that the proposed architecture for the LSTM, as it is shown on the figure 4.4, consists on a input layer that feeds the data into it's hidden layers. The 3 hidden layers consist of 256 neurons that proceed to use batch normalization and a dropout layer of 30%, finally reaching the output layer that consists on a single neuron which the output is a continuous value between 0 and 1, in which 0 means there is no requirement for treatment, and 1 means that it requires treatment.

4.4.1 LSTM Training

The training procedure for the LSTM consists on grouping sequences by the patient reference, and then only using the sequences of length 3 and 4. We checked the mean length value of the sequences, it was around 3.6 instances, so because of the lack of data we decided to choose 3 and 4. However the sequences of length 3, needed padding in order to all sequences have the same length for the model, for that we decided to simply take the first clinical assessment available and use it as the first and second assessment on the sequences, as it is shown on the figure 4.5.

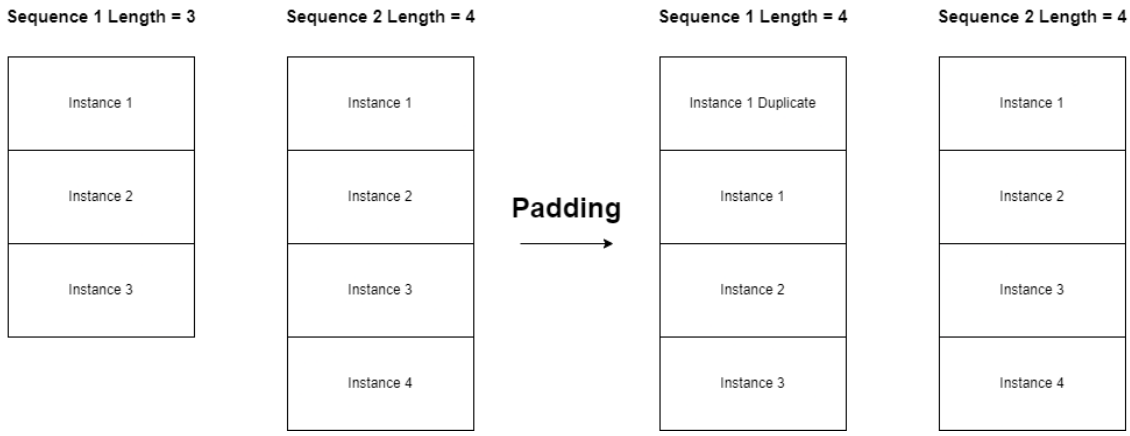

Figure 4.5: Padding Method for the Sequences of Length = 3. This approach consists on grouping the patient instances in groups of sequences of length 3 and 4. The sequences with length of 3, needed some padding, in order to be able to provide this data to train the model. For that we used the first instance of the sequence and simply use it as the first and second instances of the sequence, resulting in a sequence of length 4.

Afterwards it consists on using the Stratified K-Fold, to create the train and test sets which will be used to to train and evaluate the model. On the contrary as the MLP and CNN training procedure, this one does not have any class resampling so the sets are quite small.

4.5 Evaluation

For the evaluation of the models' performance, we use various metrics that tell us how the model is truly behaving. Given the training that is done, we need to be able to check whether the models are learning and if they are capable of generalizing to the validation data. The main metrics used for this evaluation, are the AUC-ROC Curve, the f1-score and the accuracy.

4.5.1 AUC-ROC Curve

The Receiver Operator Characteristic (ROC) curve is a binary classification issue evaluation metric. The Area Under the Curve (AUC) is a summary of the ROC curve that measures a classifier's ability to distinguish between classes. Therefore it indicates how well the model distinguishes between positive and negative classes. The greater the AUC, the better. The following figure 4.6 is an example of the different values of AUC.

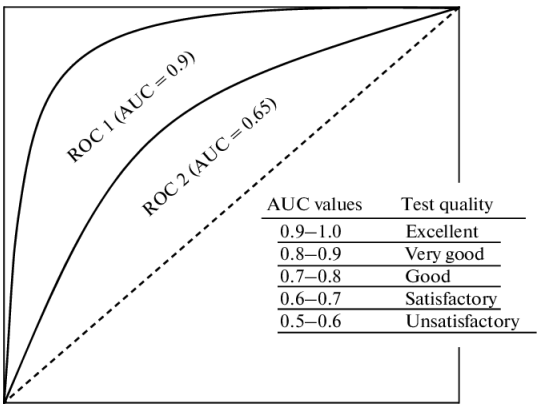

Figure 4.6: AUC ROC example. This metric summarizes the classifier's ability to distinguish between classes, the ability to distinguish between positive and negative classes. The higher the value on this metric the better performance on the model. Figure from [40].

Being this the metric we will use the most to evaluate the model's performance given the skewness of the data. As it can provide a lot of information in a single metric, making it easier to evaluate how our models behave.

4.5.2 Accuracy

Accuracy is the most common metric used to evaluate models, however is not always the best metric to use, specially when the classes are imbalanced. Such it was the case of the the disease progression class we are trying to predict, prior to the resampling done with SMOTE. Therefore the accuracy is a viable option to evaluate our model's performance on the training sets. However it is not viable to evaluate the test sets, because on this set the class is not balanced as it previously was mentioned on the section 4.

$$Accuracy = \frac{TP + TN}{P + N}$$

4.5.3 Sensitivity

Given that the Sensitivity is the model's ability to correctly identify the positive class [14], in this case it is the ability to correctly identify patients with disease progression. As

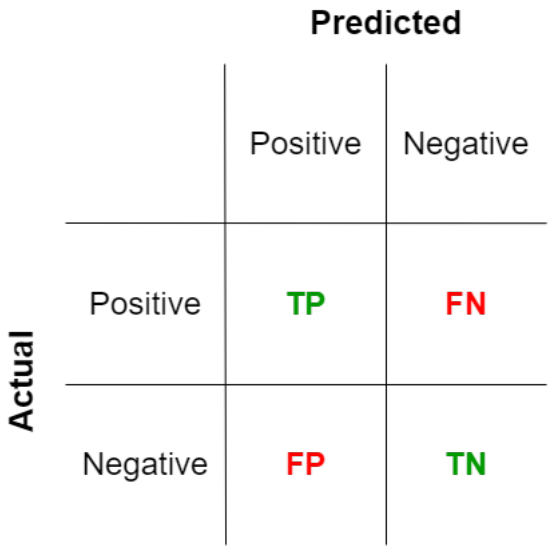

Figure 4.7: Confusion Matrix. TP, TN, FP, FN refer to the number of true positives, true negatives, false positives and false negatives.

the objective of this project is to clearly identify the positive instances, if we are able to achieve higher scores on this metric it means the model is capable to identify the minority class.

$$Sensitivity = \frac{TP}{TP + FN}$$

4.5.4 Specificity

The Specificity is the model's ability to correctly identify the negative class [14], in this case it is the ability to correctly identify patient with no disease progression. This is a metric that will no be the main objective to have the highest values however we still need to be able to identify the negative class instances in order to have a great prediction model.

$$Specificity = \frac{TN}{TN + FP}$$

4.5.5 F1-Score

This metric is one of the most used for model evaluation of binary classification problems, as it uses both precision and recall to represent the performance of the model. F1-score is better than the accuracy metric when dealing with imbalanced classes, as is the case of ours. This will be used in complement of the AUC-ROC curve, to validate how the model is behaving.

$$F1Score = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Chapter 5

Results

5.1 Baseline Results

In this section we will discuss the baseline results obtained. With the first approach mentioned on section 4.1 with the MLP and CNN models, we were able to achieve some promising results, which are reported in the table 5.1.

Some of the datasets do not have a score, due to the lack of data available, meaning that the class distribution is so sparse and the stratified group k-fold cross-validation was not able to return a stratified group. Therefore we decided to exclude those that we could not get a stratified group, in order to not get redundant scores on the AUC ROC metric.

We will then compare the models to other approaches to the problem, however the only one we can compare is with the clinical endpoint C1, as the other clinical endpoints it is the first time these are used to predict the disease progression, so there is no other available references with the same data that is comparable.

The scores show that both models were able to achieve overall great results, they behaved similarly and scored close to each other. However in the majority of the datasets the MLP performed better by a slight margin. With the scores that these models achieved on this metric, means that the models are able to distinguish between classes.
Being able to distinguish between both classes, shows that the models are quite effectively predicting the outcome. The clinical endpoint that showed the most promise is the clinical endpoint C2, that was able to surpass the 90% metric value, which is excellent results on this metric, without any hyper parameter search. The other 2 clinical endpoints 5.1, that were able to achieve good results, were the clinical endpoint C4 and C5, being able to score above 80% mark. Despite being a good score, there is still room for improvement in order to, surpass the 90% mark as the previous clinical endpoint.
The clinical endpoint C1 despite being the one with the lowest scores, still was able to achieve overall fair scores. However it is the one that needs the most improvement, given that it is the treatment that showed the most promise in improving life expectancy of patients. And given that fact there is a need to work even more on this clinical endpoint,

Dataset	Model	AUC ROC Mean		
		k=90	k=180	k=365
C1 - Need For NIV	MLP	0.761	0.746	0.792
	CNN	0.768	0.766	0.762
C2 - Need for an auxiliary Communication Device	MLP	X	0.924	0.924
	CNN	X	0.921	0.906
C4 - Need for a Caregiver	MLP	0.834	0.797	0.824
	CNN	0.850	0.782	0.800
C5 - Need for a Wheelchair	MLP	X	0.840	0.833
	CNN	X	0.827	0.823

Table 5.1: Baseline Model Mean Values for the AUC Evaluation Metric for the different Time Windows of k Days for each Clinical Endpoint. The Clinical Endpoint C3, and the time windows of 90 days of the clinical endpoints C2 and C5 do not have scores, as the class distribution was too sparse and no stratified group was found. The other models were able to achieve some promising results, with the clinical endpoint C2 being the best scoring one, obtaining above the 90% mark.

because it able to increase life expectancy of a lot of patients with ALS in the short term. Pires et al. developed models to predict the clinical endpoint C1 (Need for NIV) [32], the patient data was grouped by the stage of disease progression (Slow, Neutral and Fast), the results obtained were higher than the models we presented on table 5.1. Our models did not reach the 80% mark, where as the models Pires presented, several of the them were above 80% and one was even able to surpass the 90% mark, this shows that our models still have a lot of room to improvement.

To better understand the scores obtained, we made use of confusion matrices, to check how the models were behaving. The confusion matrices showed that the models were indeed able to predict correctly the majority of the time all positive class instances, while only miss classifying a few of the negative class instances. This shows that the models can clearly distinguish between both classes, and also be able to achieve good accuracy scores, which the models achieved in average above the 70% and 80% of accuracy, on an imbalanced set.

The following section 5.2, we will present the scores obtained by the models after doing this hyper parameter search and check whether or not we gained an increase in performance of the model.

5.2 Hyper Parameter Search

In this section we will show the results obtained when doing the parameter search. This search consisted on changing the parameter values of the preprocessing of the datasets, meaning we evaluated the models with different preprocessed methods datasets. The values changed were the number of neighbors used on the KNN imputation, the number of features used as input as well as the hidden size on the models. The objective is to optimize the scores obtained, and check if we can get better scores on some of the datasets, as this is a very costly procedure. The tables we will present, show the scores obtained by the models on the several test metrics, where each table represents a specific clinical endpoint. The following table 5.2 refers to the clinical endpoint C1 model results.

By analysing the scores of the table 5.2, we have that on the 90 day time window both models had an increase on the AUC ROC metric, where the CNN had the biggest increase from 76.8% to 79.6% when compared with the table 5.1, this is a considerable increase but there is still room for improvement, by adjusting the parameters in order to achieve better performance. However the rest of the time widows there was not an increase noticeable enough to say that there was an improvement in performance. As mentioned on the section 5.1, that Pires et al.[32] models achieved better scores than our baseline models on table 5.1, however after the hyper parameter search we were able to improve the models however not to the point to surpass Pires' models.

Something to note is that the sensitivity values, which was the ability to correctly identify patients with disease progression as mentioned on section 4.5.3, were great and close to 1 in all models, which in our case is a positive outcome given the objective we have. However the values of the specificity metric, which was the ability to identify the patients with no disease progression, were lower. On the 90 day time window, the scores obtained did not surpass the 20% mark, on the 180 day time window the scores were around the 30% mark, the best was the 365 day time window that reached the 50% mark but with sensitivity of around 80% mark.

The following table 5.3 refers to the clinical endpoint C2 model results.

By analysing the scores of the table 5.3 and comparing with the table 5.1, there was no increase in performance as the scores stayed quite similar, around the 91% mark. As this dataset already had models with great scores of AUC ROC, we did not expect any significant improvement of performance, given that the models perform quite well and achieved values extremely close to 1. The Scores obtained on the sensitivity metric were also quite close to 1, and the specificity were higher than the values obtained on the table 5.2, on the 90 day time window scores on the MLP of around 40% and on the CNN around the 30% mark, and on the 180 day time window scores on the MLP of around 60% and on the CNN around the 40% mark.

The table 5.4 refers to the clinical endpoint C4 model results.

The scores obtained on the table 5.4, on the 90 day time window the scores of AUC ROC increased by 1% on both models, the other time windows did not achieve any improvement and stayed pretty much the same. The sensitivity and specificity of the 90 time window, were close to 1 and around 0.30 mark respectively, the 180 day time window, scored around 90% and around 50% respectively. The 365 day time window achieved scores of the AUC ROC of around 80%, however it achieved scores of sensitivity and specificity around the 70% mark, which was quite different from all the other clinical endpoints which showed high values of sensitivity and low values of specificity.

We believe this happened because the 365 day time window dataset was the only that did not require any class resampling, as the disparity between classes was quite low, and the number of positive and negative instances were close to the same value. And by not having to use the SMOTE and the undersampling methods, we used all of the real data we had on this dataset. This means that perhaps the SMOTE method of creating synthetic data, can cause the model to become more sensitive rather than a mix in between sensitive and specific.

The following table 5.5 refers to the clinical endpoint C5 model results.

By analysing the scores of the table 5.5 and comparing with the table 5.1, similarly to the clinical endpoint C2, there was no increase in performance as the scores did not surpass the baseline values obtained, around the 83% in AUC ROC. However the scores are already pretty good, as the models have a decent capability of identifying the patients with disease progression as we can check the sensitivity values, and do not have the lowest values for the specificity.

An interesting note is that the majority of the models, turned out to be more sensitive than specific. Meaning the models take higher into account getting the positive instances than the negative classes, this was clear when checking the confusion matrices, where we had basically all the positive instances correctly labelled, and had the majority of the negative instances correctly labelled however with several instances being miss classified. For the context of this problem and given the lack of data available, it is a positive outcome. As well as the scores obtained on the accuracy metric, we can confirm that the models still perform quite well, where most of them are above the 80% threshold, and only on some cases this metric did not reach the 70% threshold.

With this by checking the several tables we are can notice that comparing with the results obtained with no parameter search, this process was able to increase by a slight margin the scores obtained by some models. However the parameter search done, could

be even deeper and perhaps there could have been a higher increase in results. But given that this process takes time to do, and doing it on 10 different datasets, we had to reduce the selection of parameters for the hyper parameter search. Therefore, we can also conclude that these models are very robust to different preprocessing steps.

Even so, there were some convergence on the parameters values, the most noticeable case is the number of features used as input to feed the neural networks, where the most recurrent value was 15 features as input. On a complex problem like ours, there is a need to have enough features to be able to distinguish between classes and find patterns, that might not be as easy to find, with just 5 features for example. With the other parameters, the same happened but it was not as clear, on the number of neighbours for the KNN imputation algorithm, it ended up being 4 neighbours that appeared in half of the times, as it happened on the hidden size parameter where the 512 of hidden size was the best one half of the time. Meaning the models had better performance with a neural network that was a bit more complex.

5.2.1 Parameter Convergence

As mentioned before, there were some parameters that showed some convergence such as the number of input features used. On this section will further discuss and present data that support this. We can further support this statement by presenting the top five best models of each dataset, with the following tables, which result from the hyper parameter search performed that in total sum up to 270 different models of each proposed architecture (MLP and CNN). The following table 5.6, shows the scores obtained by the models on the clinical endpoint C1.

The following table 5.7, shows the scores obtained by the models on the clinical endpoint C2.

The following table 5.8, shows the scores obtained by the models on the clinical endpoint C4.

The following table 5.9, shows the scores obtained by the models on the clinical endpoint C5.

The models shown on the previous tables support the statement of models having a better performance the more complex they are to a certain degree, which shows that the great majority of the models take advantage of having a higher number of input features used on the models, as before the number of input in which the models performed best was the 15 features. Showing that the models have an easier time distinguishing between the classes with more information available rather than 10 and 5 features, meaning the features clinicians use really have a high correlation with the outcome. It would be beneficial to go beyond the scope of values done in the hyper parameter search we performed, to understand if the highest number of input values would still achieve the best performing

models.

The following parameter value was not as clear whether or not there was any type of convergence, however by showing the tables with the top 5 best performing models it is possible to confirm whether or not there was any convergence on the number of neighbours used on the KNN imputation algorithm. By analyzing the previous tables for the KNN parameter, as mentioned on the section 5.2, the value that was the most recurrent was the number of 4 neighbours, however unlike the number of features it was not as clear. Given that, most of the models performed better with 3 or 4 neighbours on the KNN, however value of 3 neighbours was pretty close to being the most common value. Which would also be a good option, but normally we would have more values to test with, to obtain a good measure of exactly what values work better on each dataset. Although having more values to test would be beneficial, we would have to pay attention to the points made on section 3.4.1, that choosing a low k would make the results less generalizable and increase noise, however a higher k would tend to make local effects less noticeable.

On the other hand, the hidden size parameter does not seem to converge to a specific value, as the models seem to vary a lot this parameter value. This could mean that this parameter does not have as great of an impact as the other two parameters, or that the values chosen could have been lower for example instead of using 128, 256 and 512, use a hidden size value of 64 in the place of the 128.

5.3 LSTM Results

In this section we will discuss the results obtained by the LSTM models. With this approach the results obtained were not the most promising results, comparing with the previous models that were able to achieve quite promising scores. Similarly to the section 5.1 and 5.2, because of the lack of data some of the datasets do not have scores, as it was not able to have all groups with positive instances presence on the test sets.

On the following table 5.10 we will present the results obtained by the LSTM models for the clinical endpoints C1 and C2, where we performed a hyperparameter search, of the number of neighbours of the KNN imputation algorithm and the number of features used as input on the model.

The table 5.10, we can analyse the results obtained, and verify that the achieved scores were not the most promising. As most of the scores of the AUC metric were around 50% meaning they were not able to distinguish between classes and performed quite poorly, however the only one that performed slightly better was on the clinical endpoint C2 on the time window of 365 days, as it obtained a score of 73%, which it is not ideal but

when comparing with the MLP and the CNN on the table 5.3, that achieved above the 90% score, which shows that there is a lot of information that is lost when not using the whole dataset as input for the models. However these models were able to obtain more balanced scores of the sensitivity and specificity metrics, which means the models are not as inclined to get just the positive instances correctly.

The following table 5.11 we will present the results obtained by the LSTM models for the clinical endpoints C4 and C5.

By analysing the table 5.11, the scores achieved as the previous table were not the most promising. Most of the scores of the AUC were around the 50% mark on the clinical endpoint C4, so they were not able to distinguish between the classes, however on the clinical endpoint C5, the scores of the AUC were above 70% with the highest score on the 365 day time window of 76%. The time windows that got the best scores again was the 365 day time window, as we thought it would happen given there were more positive instances as the time window is larger. Comparing with the table 5.5, the scores were a obtained by the LSTM were lower but not by a big difference, which with more tweaking of the hyper parameters and collecting more data the probably could achieve similar values as the MLP and the CNN. As the table 5.10 models, the values of sensitivity and specificity were more balanced than on the section 5.2 models, most likely because there is no usage of the SMOTE to re-sample the data, which is the reason we see fit to make this happen, as it creates synthetic instances from the data available meaning it also uses the negatives instances this can create a problem that the model may be biased to being more sensitive or specific.

The models of the tables 5.10 and 5.11, like on the section 5.2 models, also showed convergence on the hyper parameters values, where the number of neighbours for the KNN imputation that achieved the best scores was with 2 neighbours, on all most of all datasets. As well as the number of inputs features used by the models they converged to 5 features used as input, which unlike the MLP and CNN from section 5.2, that more features ended up being more beneficial for the model's performance.

In order to support the claims mentioned previously, about parameter convergence we decided to present the top 5 best performing models as we did on section 5.2.1, with the MLP and CNN approach. The following table 5.12, shows the scores obtained by the models on the clinical endpoint C1.

The following table 5.13, shows the scores obtained by the models on the clinical endpoint C2.

The following table 5.14, shows the scores obtained by the models on the clinical endpoint C4.

The following table 5.15, shows the scores obtained by the models on the clinical

endpoint C5.

The tables shown provide a lot of insight to support the statements previously made, but the data shows that it was not as clear as we thought, where the models actually converged to 2 values most of the times. For the number of neighbours on the KNN algorithm, the best performing models converged to 2 or 3 neighbours, and as mentioned on section 5.2.1, we have to be careful whether or not the values are to low that makes the model less generalizable, so with that in mind we would most likely use the models with 3 neighbours used.

On the number of input features used, most models tend to converge to 5 or 10 features as input, unlike the models on the section 5.2, which most of those ended up performing better the more complex the neural networks were, on the other hand, on this approach the models do not benefit as much of having more features, where most of the best performing models have only 5 features. Even so, these models were not able to achieve as good of results as the first approach, showing that there is still some improvement that can be done.

Time Window	Model	KNN	# Input Features	Hidden Size	Metric	Mean Score
k=90	MLP	4	15	128	AUC ROC	0.777
					Accuracy	0.817
					Sensitivity	0.973
					Specificity	0.154
	CNN	3	10	256	AUC	**0.796**
					Accuracy	0.712
					Sensitivity	0.978
					Specificity	0.121
k=180	MLP	4	15	128	AUC	0.753
					Accuracy	0.689
					Sensitivity	0.912
					Specificity	0.313
	CNN	4	15	512	AUC	**0.769**
					Accuracy	0.630
					Sensitivity	0.924
					Specificity	0.293
k=365	MLP	3	15	512	AUC	**0.769**
					Accuracy	0.686
					Sensitivity	0.824
					Specificity	0.504
	CNN	4	15	128	AUC	0.764
					Accuracy	0.649
					Sensitivity	0.833
					Specificity	0.477

Table 5.2: Best scoring models from the hyper parameter search for the Clinical Endpoint C1 - Need for NIV in k days. There was a slight increase in the scores obtained by the parameter search, relatively to the baseline results, where the largest increase was on the 90 day time window with the CNN model, from 76.8% to 79.6%.

Time Window	Model	KNN	# Input Features	Hidden Size	Metric	Mean Score
k=180	MLP	3	15	512	AUC	0.916
					Accuracy	0.888
					Sensitivity	0.967
					Specificity	0.387
	CNN	3	15	256	AUC	**0.923**
					Accuracy	0.784
					Sensitivity	0.988
					Specificity	0.285
k=365	MLP	2	15	512	AUC	0.910
					Accuracy	0.868
					Sensitivity	0.951
					Specificity	0.596
	CNN	3	15	256	AUC	**0.918**
					Accuracy	0.787
					Sensitivity	0.976
					Specificity	0.392

Table 5.3: Best scoring models from the hyper parameter search for the Clinical Endpoint C2 - Need for an auxiliary Communication Device in k days. As this clinical endpoint already had excellent scores, there was no increase in the results obtained, as it was harder to improve given the baseline scores.

Time Window	Model	KNN	# Input Features	Hidden Size	Metric	Mean Score
k=90	MLP	4	5	128	AUC	0.842
					Accuracy	0.760
					Sensitivity	0.961
					Specificity	0.295
	CNN	2	15	256	AUC	**0.863**
					Accuracy	0.657
					Sensitivity	0.978
					Specificity	0.243
k=180	MLP	2	15	256	AUC	**0.787**
					Accuracy	0.728
					Sensitivity	0.875
					Specificity	0.504
	CNN	4	15	512	AUC	**0.787**
					Accuracy	0.638
					Sensitivity	0.907
					Specificity	0.425
k=365	MLP	2	15	512	AUC	**0.811**
					Accuracy	0.730
					Sensitivity	0.734
					Specificity	0.716
	CNN	2	15	512	AUC	0.804
					Accuracy	0.734
					Sensitivity	0.727
					Specificity	0.726

Table 5.4: Best scoring models from the hyper parameter search for the Clinical Endpoint C4 - Need for a Caregiver in k days. On the 90 day time window both models increased 1% relatively to the baseline results. The other time windows the scores obtained did not increase, and obtained similar scores to the baseline results.

Time Window	Model	KNN	# Input Features	Hidden Size	Metric	Mean Score
k=180	MLP	4	15	256	AUC	0.832
					Accuracy	0.780
					Sensitivity	0.952
					Specificity	0.313
	CNN	4	15	512	AUC	**0.835**
					Accuracy	0.674
					Sensitivity	0.973
					Specificity	0.251
k=365	MLP	2	15	512	AUC	**0.827**
					Accuracy	0.763
					Sensitivity	0.890
					Specificity	0.498
	CNN	4	15	128	AUC	**0.827**
					Accuracy	0.714
					Sensitivity	0.922
					Specificity	0.465

Table 5.5: Best scoring models from the hyper parameter search for the Clinical Endpoint C5 - Need for a Wheelchair in k days. The scores obtained did not show any increase of the baseline scores. However most models turned out to be more sensitive than specific, which these favor getting more positive instances correct than the negative classes.

Time Window	Model	KNN	# Input Features	Hidden Size	AUC ROC Mean	Accuracy	Sensitivity	Specificity
k=90	MLP	4	15	128	**0.777**	0.817	0.973	0.154
		4	15	512	0.753	0.855	0.965	0.148
		4	15	256	0.749	0.828	0.966	0.139
		3	10	128	0.744	0.756	0.973	0.121
		3	10	256	0.743	0.764	0.970	0.115
	CNN	3	10	256	**0.796**	0.712	0.978	0.121
		3	10	512	0.795	0.705	0.980	0.122
		4	15	512	0.795	0.625	0.984	0.106
		3	15	256	0.794	0.654	0.985	0.111
		3	15	512	0.793	0.624	0.986	0.106
k=180	MLP	4	15	128	**0.753**	0.689	0.912	0.313
		4	10	128	0.752	0.679	0.912	0.308
		4	15	256	0.752	0.707	0.908	0.326
		2	15	128	0.751	0.694	0.909	0.311
		4	10	256	0.749	0.686	0.906	0.312
	CNN	4	15	512	**0.769**	0.639	0.924	0.293
		4	15	256	0.767	0.649	0.923	0.302
		4	15	128	0.767	0.648	0.924	0.299
		3	15	256	0.765	0.646	0.925	0.293
		2	15	128	0.765	0.667	0.924	0.310
k=365	MLP	3	15	512	**0.769**	0.686	0.824	0.504
		3	15	256	0.761	0.674	0.815	0.494
		4	15	512	0.759	0.684	0.816	0.506
		2	15	256	0.758	0.669	0.814	0.489
		2	15	512	0.757	0.670	0.808	0.490
	CNN	4	15	128	**0.764**	0.649	0.833	0.477
		4	15	512	0.763	0.641	0.832	0.466
		4	15	256	0.763	0.642	0.844	0.468
		3	15	256	0.762	0.638	0.838	0.464
		3	15	128	0.760	0.634	0.833	0.461

Table 5.6: Top 5 models obtained by the hyper parameter search for the Clinical Endpoint C1 - Need For NIV.

Time Window	Model	KNN	# Input Features	Hidden Size	AUC ROC Mean	Accuracy	Sensitivity	Specificity
k=180	MLP	3	15	512	**0.916**	0.888	0.967	NAN
		3	15	128	0.912	0.854	0.975	0.352
		3	15	256	0.910	0.868	0.970	NAN
		3	10	128	0.909	0.845	0.977	0.349
		2	10	256	0.906	0.854	0.976	0.366
	CNN	3	15	512	**0.916**	0.888	0.967	0.596
		3	15	128	0.912	0.854	0.975	0.352
		3	15	256	0.910	0.868	0.970	0.598
		3	10	128	0.909	0.845	0.977	0.349
		2	10	256	0.906	0.854	0.976	0.366
k=365	MLP	2	15	512	**0.910**	0.868	0.951	0.596
		3	15	256	0.906	0.839	0.960	0.519
		3	15	512	0.906	0.869	NaN	0.598
		2	15	128	0.905	0.840	0.960	0.520
		3	15	128	0.904	0.848	NaN	0.532
	CNN	3	15	256	**0.918**	0.787	0.976	NaN
		2	15	512	0.917	0.801	0.972	0.464
		4	15	256	0.916	0.799	0.970	NaN
		2	15	128	0.916	0.791	0.974	0.452
		4	15	128	0.916	0.775	0.973	NaN

Table 5.7: Top 5 models obtained by the hyper parameter search for the Clinical Endpoint C2 - Need for an auxiliary Communication Device.

Time Window	Model	KNN	# Input Features	Hidden Size	AUC ROC Mean	Accuracy	Sensitivity	Specificity
k=90	MLP	4	5	128	**0.842**	0.760	0.961	0.295
		4	5	256	0.837	0.775	0.958	0.304
		4	10	128	0.835	0.766	0.957	0.294
		3	10	256	0.834	0.786	0.956	0.313
		2	10	256	0.834	0.768	0.953	0.290
	CNN	2	15	256	**0.863**	0.657	0.978	0.242
		4	15	256	0.863	0.678	0.978	0.257
		4	15	512	0.858	0.669	0.975	0.248
		2	15	128	0.858	0.690	0.975	0.254
		4	15	128	0.855	0.682	0.973	0.255
k=180	MLP	2	15	256	**0.787**	0.728	0.875	0.504
		2	15	128	0.787	0.717	0.878	0.488
		2	15	512	0.786	0.731	0.873	0.504
		4	15	512	0.785	0.705	0.861	0.478
		4	10	512	0.778	0.684	0.868	0.458
	CNN	4	15	512	**0.787**	0.638	0.907	0.425
		4	15	256	0.786	0.635	0.906	0.423
		4	15	128	0.784	0.624	0.912	0.419
		2	15	512	0.784	0.690	0.873	0.464
		4	10	128	0.784	0.639	0.898	0.425
k=365	MLP	2	15	512	**0.811**	0.730	0.734	0.716
		3	15	256	0.803	0.723	0.729	0.708
		3	15	512	0.803	0.726	0.726	0.708
		2	15	128	0.799	0.728	0.738	0.707
		3	15	128	0.798	0.723	0.727	0.708
	CNN	2	15	512	**0.804**	0.734	0.727	0.726
		3	15	512	0.804	0.728	0.715	0.726
		3	15	128	0.801	0.731	0.734	0.713
		2	15	256	0.801	0.731	0.726	0.720
		3	15	256	0.800	0.733	0.727	0.726

Table 5.8: Top 5 models obtained by the hyper parameter search for the Clinical Endpoint C4 - Need for a Caregiver.

Time Window	Model	KNN	# Input Features	Hidden Size	AUC ROC Mean	Accuracy	Sensitivity	Specificity
k=180	MLP	4	15	512	**0.835**	0.674	0.973	0.251
		4	15	128	0.833	0.643	0.977	0.236
		4	15	256	0.833	0.665	0.974	0.245
		2	15	128	0.830	0.710	0.965	0.268
		3	15	256	0.828	0.660	0.972	0.243
	CNN	4	15	512	**0.835**	0.674	0.973	0.251
		4	15	128	0.833	0.643	0.977	0.236
		4	15	256	0.833	0.665	0.974	0.245
		2	15	128	0.830	0.710	0.965	0.268
		3	15	256	0.828	0.660	0.972	0.243
k=365	MLP	2	15	512	**0.827**	0.763	0.890	NaN
		4	15	512	0.826	0.764	0.892	NaN
		3	15	512	0.823	0.764	0.891	NaN
		4	15	256	0.822	0.756	0.897	0.518
		3	15	256	0.822	0.753	0.896	NaN
	CNN	4	15	512	**0.827**	0.714	0.922	0.465
		4	15	256	0.824	0.705	0.921	NaN
		4	15	512	0.823	0.729	0.915	NaN
		3	15	256	0.821	0.726	0.915	NaN
		3	15	128	0.821	0.732	0.909	NaN

Table 5.9: Top 5 models obtained by the hyper parameter search for the Clinical Endpoint C5 - Need for a Wheelchair.

Dataset	Time Window	KNN	# Input Features	Metric	Mean Score
C1 - Need for NIV	k=90	3	15	AUC	0.526
				Accuracy	0.734
				Sensitivity	0.817
				Specificity	0.238
	k=180	2	5	AUC	**0.581**
				Accuracy	0.602
				Sensitivity	0.562
				Specificity	0.609
	k=365	2	5	AUC	0.510
				Accuracy	0.553
				Sensitivity	0.422
				Specificity	0.598
C2 - Need for an Auxiliary Communication Device	k=90	2	5	AUC	0.542
				Accuracy	0.807
				Sensitivity	0.868
				Specificity	0.462
	k=180	2	5	AUC	0.583
				Accuracy	0.589
				Sensitivity	0.553
				Specificity	0.618
	k=365	2	5	AUC	**0.731**
				Accuracy	0.751
				Sensitivity	0.808
				Specificity	0.668

Table 5.10: Hyper Parameter Search of the LSTM models on the Clinical Endpoints C1 and C2. The results from this model were not the most promising, which most models were not able to achieve the 60% mark on the AUC metric, however the only one to surpass the 70% mark was on the 365 day time window on the clinical endpoint C2.

Dataset	Time Window	KNN	# Input Features	Metric	Mean Score
C4 - Need for Caregiver	k=90	2	15	AUC	0.501
				Accuracy	0.758
				Sensitivity	0.827
				Specificity	0.185
	k=180	3	5	AUC	0.493
				Accuracy	0.610
				Sensitivity	0.317
				Specificity	0.673
	k=365	2	10	AUC	**0.560**
				Accuracy	0.720
				Sensitivity	0.508
				Specificity	0.760
C5 - Need for a Wheelchair	k=180	2	5	AUC	0.720
				Accuracy	0.745
				Sensitivity	0.760
				Specificity	0.739
	k=365	2	5	AUC	**0.764**
				Accuracy	0.753
				Sensitivity	0.734
				Specificity	0.789

Table 5.11: Hyper Parameter Search of the LSTM models on the Clinical Endpoints C4 and C5. On the clinical endpoint C4 the models stayed around the 50% mark meaning they are not able to distinguish between classes, however the clinical endpoint C5 models were slightly better and achieved above the 70% mark.

Time Window	KNN	# Input Features	AUC ROC Mean	Accuracy	Sensitivity	Specificity
k=90	2	5	**0.542**	0.807	0.868	0.462
	4	15	0.532	0.779	0.866	0.213
	2	10	0.530	0.789	0.865	0.304
	4	10	0.526	0.781	0.864	0.210
	2	15	0.525	0.768	0.864	0.258
k=180	2	5	**0.583**	0.589	0.553	0.618
	2	10	0.547	0.555	0.512	0.585
	2	15	0.510	0.519	0.465	0.555
	3	10	0.507	0.519	0.458	0.552
	3	5	0.506	0.513	0.460	0.551
k=365	2	5	**0.731**	0.751	0.808	0.668
	2	10	0.726	0.736	0.822	0.635
	3	5	0.711	0.728	0.801	0.629
	3	10	0.709	0.724	0.805	0.621
	4	5	0.702	0.720	0.795	0.615

Table 5.12: Top 5 models obtained by the hyper parameter search of the clinical endpoint C1 - Need For NIV.

Time Window	KNN	# Input Features	AUC ROC Mean	Accuracy	Sensitivity	Specificity
k=90	2	5	**0.542**	0.807	0.868	0.462
	4	15	0.532	0.779	0.866	0.213
	2	10	0.530	0.789	0.865	0.304
	4	10	0.526	0.781	0.864	0.210
	2	15	0.525	0.768	0.864	0.258
k=180	2	5	**0.583**	0.589	0.553	0.618
	2	10	0.547	0.555	0.213	0.297
	2	15	0.510	0.519	0.465	0.555
	3	10	0.507	0.519	0.458	0.552
	3	5	0.506	0.513	0.460	0.551
k=365	2	5	**0.731**	0.751	0.808	0.668
	2	10	0.726	0.736	0.822	0.635
	3	5	0.711	0.728	0.801	0.629
	3	10	0.709	0.724	0.805	0.621
	4	5	0.702	0.720	0.795	0.615

Table 5.13: Top 5 models obtained by the hyper parameter search of the clinical endpoint C2 - Need for an auxiliary Communication Device.

Time Window	KNN	# Input Features	AUC ROC Mean	Accuracy	Sensitivity	Specificity
k=90	2	15	**0.501**	0.758	0.827	0.185
	4	15	0.499	0.756	0.826	0.183
	4	5	0.498	0.755	0.826	0.175
	3	10	0.497	0.757	0.826	0.180
	3	5	0.497	0.761	0.826	0.172
k=180	3	5	**0.493**	0.610	0.317	0.673
	3	10	0.486	0.603	0.297	0.669
	2	10	0.484	0.594	0.283	0.668
	2	15	0.483	0.596	0.286	0.667
	4	5	0.481	0.602	0.300	0.666
k=365	2	10	**0.560**	0.720	0.508	0.760
	3	15	0.549	0.706	0.508	0.755
	2	15	0.547	0.703	0.456	0.753
	3	10	0.547	0.708	0.508	0.754
	3	5	0.544	0.708	0.508	0.752

Table 5.14: Top 5 models obtained by the hyper parameter search of the clinical endpoint C4 - Need for a Caregiver.

Time Window	KNN	# Input Features	AUC ROC Mean	Accuracy	Sensitivity	Specificity
k=180	2	5	**0.720**	0.745	0.760	0.739
	2	10	0.712	0.736	0.760	0.706
	2	15	0.708	0.728	0.758	0.689
	3	15	0.701	0.722	0.755	0.681
	3	5	0.701	0.723	0.751	0.682
k=365	2	5	**0.764**	0.753	0.734	0.789
	3	5	0.753	0.742	0.736	0.751
	2	15	0.751	0.742	0.735	0.750
	3	10	0.750	0.738	0.733	0.746
	4	5	0.748	0.736	0.731	0.742

Table 5.15: Top 5 models obtained by the hyper parameter search of the clinical endpoint C5 - Need for a Wheelchair.

Chapter 6

Conclusion and Future Work

In this section we will conclude the discuss of the results obtained. We proposed the use of neural networks, such as the MLP, the CNN and a LSTM, to tackle the prediction of disease progression of the different clinical endpoints available within predefined time windows of 90, 180 and 365 days.

On the first approach proposed the results are promising, achieving up to 92% in AUC, for the clinical endpoint C2 in the 90 day time window. The results were somewhat expected, however the fact that the best resulting models of a certain clinical endpoint, were within the 90 day time window, which surprised us. We expected that the results would get better as the time windows were larger, given that the number of positive instances increased and the disparity between classes lowered, this was not the case whatsoever.

On the other hand, the LSTM models performed as we thought they would where the largest time window would be the best scoring, because it had more positive instances than the other time windows. However the scores achieved were not as promising as the MLP and CNN models, since using sequences of length 3 and 4, reduces drastically the data available for the model to learn from, which translates to a lower performance. To improve this, an approach is to use even more sequences of different lengths and find a padding technique that fits best to improve the sequences we have.

There is still a lot that can be done to improve the results obtained, such as a deeper hyper parameter search and to further explore the hyper parameter search there are still several parameters we did not experiment, that could potentially further improve the results achieved. On the first approach we only explored 3 different parameters, with 3 different values each, this is not a nearly enough to consider it as concluded. Ideally, we would have liked to have at least 5 parameters with minimum of 5 to 6 values each, this would give us for each dataset around 75 different models tested, right now we have 27 on each dataset. And we still were able to slightly improve some of the results, with a deeper hyper parameter search there might be some improvement to achieve.

For the models we proposed there is a need to obtain more data to train this type of models that are really dependent on how much data you can feed into it, where we did everything not to throw away data that could be essential to have good results. However we believe that these type of prediction models will be very useful once fully tested and used on a practical context with real patients. It might prove to be an effective tool to help clinicians reach a decision on when to treat a patient, and the patients' well being, with the objective to improve quality of life and increase the life expectancy of the patient.

Bibliography

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6. Ieee, 2017.

[2] Michael S Bereman, Joshua Beri, Jeffrey R Enders, and Tara Nash. Machine learning reveals protein signatures in csf and plasma fluids of clinical value for als. *Scientific reports*, 8(1):1–14, 2018.

[3] Santaniello A. Beretta L. Nearest neighbor imputation algorithms: a critical evaluation. *BMC Med Inform Decis Mak*, 16(74), 2016.

[4] Vadim Borisov, Johannes Haug, and Gjergji Kasneci. Cancelout: A layer for feature selection in deep neural networks. In *International Conference on Artificial Neural Networks*, pages 72–83. Springer, 2019.

[5] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey, 2021.

[6] Shaofeng Cai, Kaiping Zheng, Gang Chen, HV Jagadish, Beng Chin Ooi, and Meihui Zhang. Arm-net: Adaptive relation modeling network for structured data. In *Proceedings of the 2021 International Conference on Management of Data*, pages 207–220, 2021.

[7] André V. Carreiro, Pedro M.T. Amaral, Susana Pinto, Pedro Tomás, Mamede de Carvalho, and Sara C. Madeira. Prognostic models based on patient snapshots and time windows: Predicting disease progression to assisted ventilation in amyotrophic lateral sclerosis. *Journal of Biomedical Informatics*, 58:133–144, 2015.

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

[9] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[10] Moniek C.M. de Goeij, Merel van Diepen, Kitty J. Jager, Giovanni Tripepi, Carmine Zoccali, and Friedo W. Dekker. Multiple imputation: dealing with missing data. *Nephrology Dialysis Transplantation*, 28(10):2415–2420, 05 2013.

[11] P.A.M. Dirac. The lorentz transformation and absolute time. *Physica*, 19(1-–12):888–896, 1953.

[12] Ludolph AC Dorst J. Non-invasive ventilation in amyotrophic lateral sclerosis. pages 1–1, 2019.

[13] Georgios Douzas, Fernando Bacao, and Felix Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1–20, 2018.

[14] John J Dziak, Donna L Coffman, Stephanie T Lanza, Runze Li, and Lars S Jermiin. Sensitivity and specificity of information criteria. *Briefings in Bioinformatics*, 21(2):553–565, 03 2019.

[15] R.P Feynman and F.L Vernon Jr. The theory of a general quantum system interacting with a linear dissipative system. *Annals of Physics*, 24:118–173, 1963.

[16] Michal S Gal and Daniel L Rubinfeld. Data standardization. *NYUL Rev.*, 94:737, 2019.

[17] Vincent Grollemund, Pierre-François Pradat, Giorgia Querin, François Delbot, Gaétan Le Chat, Jean-François Pradat-Peyre, and Peter Bede. Machine learning in amyotrophic lateral sclerosis: Achievements, pitfalls, and future directions. *Frontiers in Neuroscience*, 13:135, 2019.

[18] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[19] J.T. Hancock and Khoshgoftaar. Survey on categorical data for neural networks. *Big Data*, 7:1, 2020.

[20] Sergei Ivanov and Liudmila Prokhorenkova. Boost then convolve: Gradient boosting meets graph neural networks. In *International Conference on Learning Representations*, 2021.

[21] Liran Katzir, Gal Elidan, and Ran El-Yaniv. Net-dnf: Effective deep modeling of tabular data. In *International Conference on Learning Representations*, 2020.

[22] Guolin Ke, Jia Zhang, Zhenhui Xu, Jiang Bian, and Tie-Yan Liu. Tabnn: A universal neural network solution for tabular data. 2018.

[23] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[24] Leslie Lamport. *LaTeX - A Document Preparation System: User's Guide and Reference Manual, Second Edition*. Pearson / Prentice Hall, New York, 1994.

[25] Pat Langley and Stephanie Sage. Oblivious decision trees and abstract cases. In *Working notes of the AAAI-94 workshop on case-based reasoning*, pages 113–117. Seattle, WA, 1994.

[26] Bengio Y. LeCun Y. and Hinton G. Deep learning. page 436–444, 2015.

[27] Charles X. Ling and Victor S. Sheng. *Class Imbalance Problem*, pages 171–171. Springer US, Boston, MA, 2010.

[28] Yuanfei Luo, Hao Zhou, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. Network on network for tabular data classification in real-world applications. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2317–2326, 2020.

[29] Andreia S. Martins, Marta Gromicho, Susana Pinto, Mamede de Carvalho, and Sara C. Madeira. Learning prognostic models using diseaseprogression patterns: Predicting the need fornon-invasive ventilation in amyotrophic lateralsclerosis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1, 2021.

[30] Marcel Müller, Marta Gromicho, Mamede de Carvalho, and Sara C. Madeira. Explainable models of disease progression in als: Learning from longitudinal clinical data with recurrent neural networks and deep model explanation. *Computer Methods and Programs in Biomedicine Update*, 1:100018, 2021.

[31] Yohan Obadia. The use of knn for missing values, 2017.

[32] Sofia Pires, Marta Gromicho, Susana Pinto, Mamede Carvalho, and Sara C. Madeira. Predicting non-invasive ventilation in als patients using stratified disease progression groups. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 748–757, 2018.

[33] Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *International Conference on Learning Representations*, 2020.

[34] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[35] Claude Sammut and Geoffrey I Webb. *Encyclopedia of machine learning and data mining*. Springer Publishing Company, Incorporated, 2017.

[36] Ira Shavitt and Eran Segal. Regularization learning networks: Deep learning for tabular datasets. In *NeurIPS*, pages 1386–1396, 2018.

[37] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[38] Celine Tard, Luc Defebvre, C Moreau, D Devos, and V Danel-Brunaud. Clinical features of amyotrophic lateral sclerosis and their prognostic value. *Revue neurologique*, 173(5):263–272, 2017.

[39] tmp. 2nd place solution - with 1d-cnn, 2020.

[40] Oxana Trifonova, Petr Lokhov, and A.I. Archakov. Metabolic profiling of human blood. *Biomeditsinskaya Khimiya*, 60:281–294, 05 2014.

[41] A. M. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 01 1937.

[42] Hannelore K. van der Burgh, Ruben Schmidt, Henk-Jan Westeneng, Marcel A. de Reus, Leonard H. van den Berg, and Martijn P. van den Heuvel. Deep learning predictions of survival based on mri in amyotrophic lateral sclerosis. *NeuroImage: Clinical*, 13:361–369, 2017.

[43] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

[44] Nina Zhou and Paul Manser. Does including machine learning predictions in als clinical trial analysis improve statistical power? *Annals of Clinical and Translational Neurology*, 7(10):1756–1765, 2020.

[45] Zhang-Yu Zou, Chang-Yun Liu, Chun-Hui Che, and Hua-Pin Huang. Toward precision medicine in amyotrophic lateral sclerosis. *Annals of translational medicine*, 4(2), 2016.

[46] Eugenio Zuccarelli. The dos and don'ts of imputation, 2020.