Prediction of the Outcome of Asphyxia Patients After Hypothermic Treatment with Continuous Variables

Selin Acan
selin.acan@student.uva.nl

July 20, 2023, 112 pages

Academic supervisor: Dr. A.S.Z. (Adam) Belloum, a.s.z.belloum@uva.nl
Daily supervisor: UMC Utrecht/Dr. Richard Bartels, r.t.bartels-6@umcutrecht.nl
Host organisation/Research group: UMC Utrecht/Divisie Vrouw & Baby

Universiteit van Amsterdam
Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Master Software Engineering
http://www.software-engineering-amsterdam.nl

Abstract

Asphyxia is defined as the sudden and extensive oxygen shortage and reduced cerebral blood flow caused by birth abnormalities. The oxygen shortage primarily affects how blood is distributed to vital organs. As an outcome of this whole situation, newborn babies have to deal with neurological disorders in their future lives such as seizures. Asphyxia manifests itself in both the Netherlands and all around the world up to 23% of infant mortality rate. In this research, we tried to identify whether or not we can predict the outcome of asphyxiated babies using the observations taken at 6 hours to 4 days after birth. By the time this thesis is published, no prior similar research which aims to directly identify the problem using measurements over a set of continuous variables is seen in the literature. However, there are closely related researches that can be used to gain a more in-depth understanding of the issue. They are explained in detail in section 6. We tried 2 different approaches to tackle the problem. We failed on the first approach, however, this approach revealed a lot of information about the characteristics of the data and helped us succeed in the next one.

We developed a modified version of the advanced neural network structure AlexNet and fed the network with the generated recurrence plots. We achieved an average of 93.3% accuracy. Each observation is evaluated individually and hence the detailed evaluation metrics are provided separately, in section 4. Our study concludes that it is possible to aid doctors with machine learning models in their diagnosis of asphyxiated patients without waiting for the (Magnetic Resonance Imaging) MRI Scores. We also provided at which point the patient's severity level starts to change. Identifying these time points can be valuable for clinicians, as it helps in understanding the temporal patterns and making informed decisions about patient management and treatment strategies.

Contents

Chapter 1

Introduction

1.1 What is Asphyxia and Its Current Treatment

Asphyxia is defined as the sudden and extensive oxygen shortage and reduced cerebral blood flow caused by birth abnormalities [43]. The oxygen shortage primarily affects how blood is distributed to vital organs. To compensate for this situation, a newborn baby's body releases excitatory molecules with water into the cerebral cells causing inflammation and leading to permanent damage and dysfunction. This process is named as acidosis which can be described as an increase in acidic concentration in blood and tissues [33]. After the primary energy failure, the baby's blood flow is restored and it experiences a short recovery period allowing the clinicians to start the therapeutic treatment. The only known treatment for the disease is therapeutic hypothermia which has to be applied as early as the first six-hour window after birth to be effective since the next energy failure occurs after six to forty-eight hours after the primary failure. According to the paper [56] as an outcome of this whole situation, newborn babies have to deal with neurological disorders in their future lives such as seizures.

1.2 Global Rates of the Asphyxia Cases

The calculated percentage of child deaths with an age lower than 1 year within a thousand samples is defined as the infant mortality rate (IMR) [66]. According to the paper [56], deaths related to asphyxia correspond to 23% of the overall global IMR. The ratio of IMR associated with asphyxia is higher in the less developed countries by 78% when compared to high-income countries that have only 2% of IMR. High IMR in less developed countries is due to poor conditions like complications during birth and malnutrition. Moreover, although 2% is considerably low in developed countries, about 20% of this number further develops hypoxemic injury and there are over 400,000 children globally facing long-term disability due to this birth complication. According to the paper [43], neurological disorders originating from asphyxia in neonatal patients ranked in second place all around the world with an 8.5% IMR. Considering these numbers asphyxia can be considered as a significant global concern.

1.3 Manifestation of the Problem in the Netherlands

According to the paper [32] which represents the findings from "Perinatal Registery", out of approximately 180 thousand births in a year in the Netherlands, around 200 of those babies experienced asphyxia. Moreover, Flanders, from the same study [32], also showed an interesting result with around 65 thousand annual births including 75 of them being severely affected by the outcome of asphyxia. Moreover, studies reveal that the chances of survival increase, and the risk of developing a neurological disorder decreases with hypothermic treatment. The paper [32] shows a study between 2018 and 2021 in the Netherlands, and Flanders, which identifies the neurological differences between thermic and non-thermic patients. The study does not contain neonatal babies who are born with congenital disorders or who experience metabolic problems. The mortality rate concerning the patients who are selected from the sample satisfying these conditions was 31.8%. The survivors were not so lucky as well since out of those remaining patients 21 of them experienced cerebral palsy and 19 had developmental issues after 2 years and two of them unfortunately lost their hearing. The surprising part was not limited by this, out of 308 samples the ratio which covers severe neurological disorders and death was 45.5%.

Furthermore in the following combined study [84] carried out by VUMC and Wilhelmina Children's Hospital in Utrecht, there was a significant difference between patients who had developmental problems who had undergone hypothermic treatment and not. The study included a questionnaire testing skills in different domains such as problem-solving capacity, emotional and communication skills, and motor abilities. The information is collected for 4 years starting in 2018. This showed researchers that the observations collected from asphyxiated patients require further investigation.

1.4 Long and Short-Term Outcomes of the Disease

More than 50% of newborns who needed to take the hypothermic treatment and had to undergo resuscitation at birth showed a fast recovery and had no significant signs of encephalopathy. The observed outcome and functioning of the mentioned children were normal concerning their peers. This is the main reason why doctors still find it appropriate to perform neuroprotective treatments for patients suffering from moderate to severe asphyxia. However recent studies published by the time this thesis is written showed that, even though these patients did not develop neurological disorders like encephalopathy, their results were not good and clear. In the long term, these patients showed an incremented risk of learning difficulties in school-age at around 8 years. It is important to investigate this issue in more depth since it raises concerns about the potential risk of more subtle disabilities [5] such as:

- Problems in motor skills such as cerebral palsy, irritability, or explosiveness.
- Blindness or deafness.
- Problems in cognitive skills such as autism and learning/remembering difficulties.
- Problems in IQ Levels
- Problems in neuro-psychology
- In more severe cases death, encephalopathy, or seizures. These are usually experienced in a shorter term.

1.5 Clinical Biomarkers Used in The Detection of Asphyxia

There is no single biomarker that will provide an accurate detection of asphyxia. Below you will find the most popular bio-markers used in industry to detect the occurrence of this disease [56, 77, 31, 16].

1.5.1 The Apgar Score

Neonatal patients with an Apgar score of below 3.5 carry a high risk of death and severe disability increasing in a non-uniform manner. The paper [77] also shows that 1 out of 3 surviving patients with a 0 Apgar score had shown normal developments after their period in the intensive care unit. It takes only 10 minutes to get the Apgar results and they are used pretty commonly to qualify a neonatal patient for hypothermia treatment.

1.5.2 The Sarnat Scale

The Sarnat scale is used to detect the severity level of asphyxia in neonatal patients. The scale provides 3 severity levels: Mild, Moderate, and Severe, which will also form our classes in this research. An EEG is used periodically to access continuous variables. Three electrodes are placed on the baby's head. This way continuous variables are accessed. The main advantage is that it is easy to perform and information is accessed in real-time. An abnormal EEG at 48 hours after birth is an indication of a problem with the baby's development. A severely abnormal EEG at 36 hours and 48 hours after birth indicates a severe injury. The score obtained is also used for the qualification of neonatal patients for hypothermia treatment. The paper [77] notes that patients with symptoms of moderate severity level for less than 5 days developed normally while patients with moderate level for more than a week resulted in disabilities or death.

1.5.3 Blood Tests

Acid-base balance test is commonly used since a process called metabolic acidosis is an important qualification for neonatal patients to undergo hypothermia treatment. Although the level of lactacidemia

indicates the severity of asphyxia, it alone does not give accurate information on the duration of asphyxia. Therefore the results of this test are combined with the following blood test results:

- *Low arterial cord PH* is also a strong indication of neonatal mortality.
- *Liver enzymes like myoglobin and creatine kinase-Mb* are important for the diagnosis of asphyxia.
- *Aspartate transaminase (AST) and alanine transaminase (ALT), alkaline phosphatase and ammonia levels* also give useful information on the severity of asphyxia [77]. The main advantage of using these enzymes is that they are relatively cheap and accessible globally.

Recent studies also light up some biomarkers that can be used to predict asphyxia in patients. Such markers are; plasma neurofilament light protein, interferon, sercetoneurin, osteopontin, monocyte chemotactic protein-1, macrophage inflammatory protein 1a, vascular endothelial growth factor (VEGF), leptin, adiponectin, and erythropoietin.

1.5.4 Magnetic Resonance Imaging (MRI)

MRI provides a way to detect hemorrhage and calcification. Around 38% of patients who have undergone hypothermia treatment have intracranial hemorrhage (ICH) and therefore MRI scans are important to take into account.

Our study will be using MRI Scores as patient labels for the time series information. The labels are extracted using a novel MRI Scoring form [80], which will be further detailed in the later sections.

1.6 Biomarkers Used to Predict Short-Term Outcome of Neonates After Asphyxia

1.6.1 Neuron Specific Enolase in Serum

The paper [10] explains a study conducted on 50 patients to reveal the relationship between neuron-specific enolase [NSE], which can be found in the serum, and the brain damage and long-term outcome of infants who have undergone asphyxia. The results of the study suggest that when compared with the control group, Apgar scores and arterial cord blood PH were lower and serum NSE and arterial cord blood base deficit were higher in asphyxia patients, which indicates the existence of the relationship.

1.6.2 Thompson Scoring

The paper [13] describes a study performed on 145 post-asphyxiated patients with low-Apgar scores combined with a low, moderate, and severe Thompson Score on days 1, 3, and 7. The results of the study indicate that there is a correlation between the outcome of the post-asphyxiated patients and the day 1 Thompson Score, which is convenient and practical in terms of assigning the patients a direct numeric score instead of putting them into categories.

1.6.3 Human Umbilical Cord Blood CD34-Positive Cells

According to the paper [35] which conducted a study on 45 patients of which 20 experienced asphyxia (11 of them were stage 1 and 9 of them were in stage 2 and stage 3 based on the Sarnat Scale) and the remaining 25 kept as a control group, it is found that the absolute CD34+ cell count (p=0.02), relative CD34+ cell count (CD34+%) (p¡0.001), TLC (p=0.01), and NRBC count (p=0.02) were significantly higher in asphyxia patients than in healthy control group patients. The mentioned metrics have a strong correlation with the mortality and the degree of severity of the patients in terms of prognosis of the short-term outcome of asphyxia within a 1-4 week period.

1.6.4 Amplitude Integrated Electroencephalography (aEEG) Patterns

The study in the article [30] is performed on a subgroup of asphyxia patients who are enrolled in the hypothermia treatment. The outcome is classified as good if the assessment of cognitive and physical development, key indices for growth and progress, is above 70 and poor if otherwise. The study confirms that aEEG patterns that have recovered within the first 24-hour window evaluate a normal outcome and can be used as a prognostic value.

How Will These BioMarkers Contribute?

According to the articles [71, 6] a combination of both pre-clinical variables and MRI results provide the most accurate results. The accuracy of MRI scores raises from 85% to 90% when combined with clinical variables.

1.7 Research Question

To tackle the issue presented in the earlier sections, we tried to answer the following research question:

Can we predict the outcome of Asphyxia patients between 6 hours and 6 days after birth (with intervals of 2 hours) based on continuous data?

The outcome of the project is defined when compared with an MRI at around day 6 depending on the condition of the neonatal patient. The usage of the developed tool is not yet to steer treatment, but instead to give parents a more accurate prognosis before the MRI. The project in the future can be expanded to also steer treatment with e.g. expensive stem cell therapy.

Since the diagnosis is already made, the longer monitoring window will enable the parents of patients to have a clear decision on what percentage of the family should proceed with the treatment, what are the chances of success and what is the rate of failure, and if it is worth of it to go under that intense and expensive treatment. This will also give the patient a chance to try less intense treatment options before moving on with the hypothermia treatment if the outcome is not as severe. This outcome will be obtained in a much faster manner with the use of AI tools without waiting for the MRI.

1.8 Research method

In this section, the current approach to be followed will be explained, what kind of relevant information is necessary as an outcome of our literature study to achieve this goal and an overview of the time required for the successful completion of the project will be provided.

Previous approaches related to the topic are analyzed to gain an understanding of what possible solutions exist and how a novel approach can be proposed for the solution of the problem at hand. Since there was no previous research done on this specific topic, two novel methods are initially thought to be suitable for the solution of this problem. These two methods belong to the class of

- Generative Techniques
- Discriminative Techniques

When the original dataset was first received, its initial analysis showed us that it requires augmentation for optimal training. To apply data augmentation to time series data, where the label information may change at any point in time, it seems it would be suitable to develop a generative adversarial network for the regeneration and classification of signals in the dataset. A generative adversarial neural network (GAN) is constructed with CNN-based generative and discriminative components. Our initial guess was that it will allow us to both increase the sample size and accuracy of the built model. According to the articles mentioned in sections 3.3 and 3.4, all the possible bio-markers are collected and considered as input variables to build the model from data provided by the University Medical Center Utrecht. The received dataset was not preprocessed so the preprocessing, feature selection, and optimal data representation techniques are applied and explained in detail in this thesis. We decided to do this process manually to have higher control over the information, avoid bias and make it easier and faster. The dataset used in our research consists of 187 patients and it is appended with metavision parameters and clinical annotations of the EEG. Since the GAN model did not achieve satisfactory accuracy for short-duration time series data and had convergence problems due to large input sizes for long-duration time series data, we changed our direction to discriminative methods and, due to demonstrated success of CNNs for the classification of two-dimensional image data, the time series signal in the dataset are converted into 2D images. Since temporal differences of sample points are the most important features in time series data, recurrence plots are found to be the most appropriate conversion method for this purpose. Successful results over experimental evaluations exhibited the suitability of this approach as illustrated in the experimental works section. Consequently, AlexNet which is shown to be a successful image classification architecture in competitions is selected to be the deep learning model to be applied in our project also. This model is later used to predict the outcome of the patients. Our results are validated by comparing the outcome with the MRI of the patient on day 6. The achieved results will

provide us with information on the classification of Asphyxia patients based on measurements between 6 hours and 6 days after birth (with intervals of 2 hours) based on continuous data. This way the patients do not have to wait for the MRI result and the parents will have a clear vision and a clear future plan on whether they should proceed with the treatment or go for alternative options. Overall the completion of the project took around 3.5 months, which you can find a more detailed plan in the later sections.

1.9 Expected Results of Project

With the use of this data modeling tool, we aim to help clinicians foresee the severity level of the diagnosed disease and predict the probability of losing the patient before it happens. This way we can help the families and the clinicians with the decision of determining appropriate treatment.

1.10 Required expertise for this project

In the below table, you can find information on the expertise that is needed for the successful completion of this project. Next to each expertise, you can also find the current level of proficiency I have.

Expertise Needed	Level Of Proficiency			
	None	Low	Moderate	High
Python Programming				x
AI Tools for Data Modelling and Prediction				x
Data Preprocessing				x
Feature Extraction From MRI Images			x	
Good Clinical Practice(GCP) Training				x

Table 1.1: Required Expertise For The Project vs My Current Proficiency Level

1.11 Risks

Below you can find a list of risks and limitations for the project:

- The capability of AI Tools for appropriate modeling can be limited.
- The amount of data may not be enough for appropriate modeling and prediction.
- The project may exceed the foreseen deadline.
- Capabilities of computational tools (both in hardware and in software) to handle large amounts of training data. (Ex: Storage Space, etc)
- Different devices may be affected by different events at the time of the measurement which may introduce some bias or inconsistency to the sensor values.
- When dealing with multimodal signals, it is almost inevitable to receive a complete sample set with no missing values. The existence of such deficiencies may reduce the quality of datasets.
- Complex interactions between different modes of data representation give rise to variations and disparities, possibly caused by inconsistencies between training and testing datasets. Representing the multimodal datasets, with their inherent heterogeneity, poses a significant challenge in effectively utilizing the complementary and overlapping information across various modalities.

1.12 Contributions

The main difference in our approach will be the bio-markers used and the interval of collected continuous data. Although most of the studies mentioned in the previous sections are working with high accuracy they are done between 0-6 hour window and developed mainly for detection of the disease while our study will take place between 6 hours and 6 days after birth with intervals of 2 hours and it will be used to predict the outcome of Asphyxia patients after they are cooled down to reveal neurological abnormalities that may show itself in later ages of the patient.

1.13 Project Timeline

Below you can see the detailed project timeline :

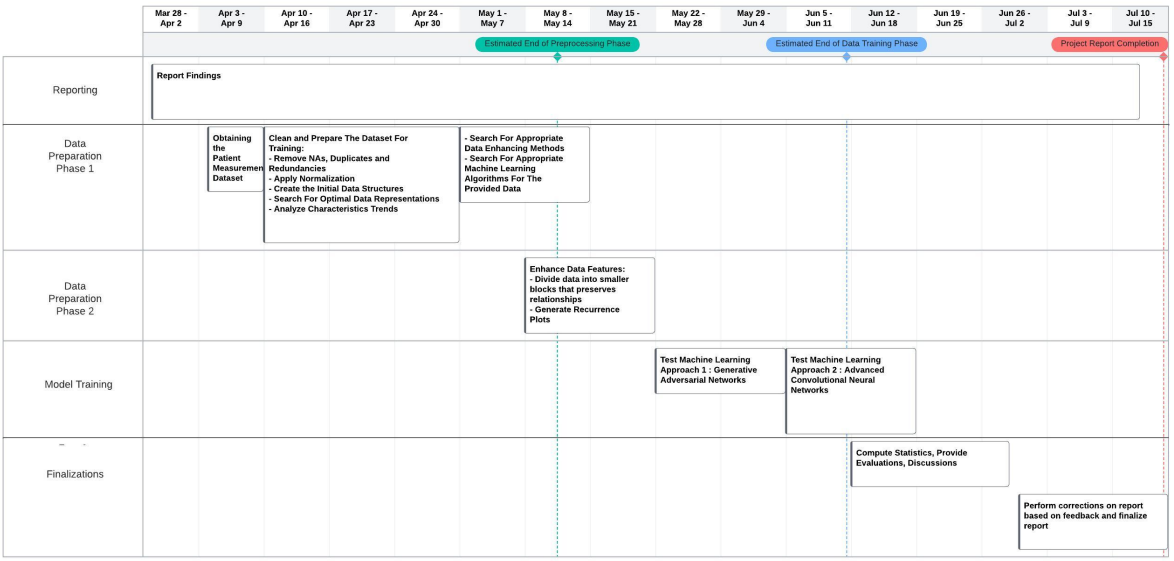


Figure 1.1: Figure displaying the milestones and goals I have set throughout the project on a weekly basis.

I have reported my findings throughout the whole project duration, which is why I spent a relatively small amount of time on additional reporting activities. I achieved the pre-set milestones and deadlines with +4-5 days for all defined phases. In terms of project planning and scheduling, we can conclude that the project succeed.

1.14 Outline

The thesis format follows the below scheme:

- Section 2 provides the necessary background information needed to be able to understand the context of the thesis and the technical terms used.
- Section 3 provides information about my research and the following Section 4 shows the outcomes I have obtained as a result of the completed research.
- Section 5 highlights the important points I learned from this study.
- Section 6 provides information about different previous approaches which tried to tackle a similar problem.
- Section 7 provides information about my conclusions and prepares the report for future work to be done.

Chapter 2

Background

This chapter will prepare the readers with the required background information to be able to follow the application details, reasoning, and technical terms referred to in this thesis. Information presented in this chapter will cover all terminology and the definitions used throughout the thesis, however, the context of information will be generic. Application and content-specific information will be provided in the Research chapter to present a more understandable link between state-of-the-art practices and presented in this thesis.

2.1 Preprocessing

2.1.1 Data Mining and Importance of Data Representation

As the complexity of a system increases, the importance of data representation becomes at least as significant as the structure of the trained neural network. Many studies [18] focus on performance optimization and generalization of models, but in reality, the highest contributor to the success of a machine learning application lies within the preprocessing and encoding phase. These stages are generally application dependent and come with their challenges like the *curse of dimensionality*. The curse of Dimensionality can be described as the unstable nature of increasing data dimensions and respectively increasing computational power in an exponential manner which causes challenges in processing and analysis. Theoretically, as the data dimensions increase, the information embedded in the data also increases. However, this also increases noise and redundancy in data, which makes the remaining computational processes practically challenging.
Hughes [36] explained this phenomenon with the following words:

With a consistent number of training data, classifiers demonstrate an initial increase in the foreseen power as the number of dimensions expands. However, once a specific threshold of dimensions is surpassed, the performance of the classifiers starts to decrement. As a consequence of this explanation, this concept is known as the Hughes phenomenon.

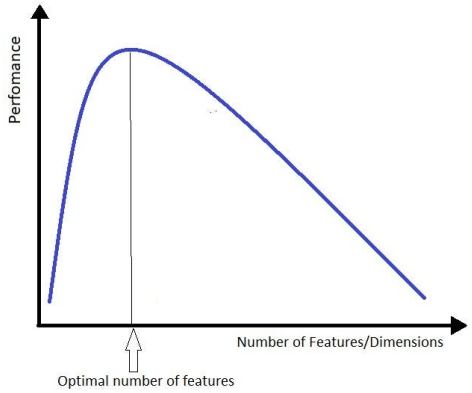


Figure 2.1: Figure displaying Hughes Principle: Curse of Dimensionality. Image From: [42]

Therefore feature selection is important both for time and space complexity, particularly when it comes to time series concerning multiple variables.

Many machine learning models expect uniform input representation which is homogeneous and has component values between 0 and 1. In an ideal situation where the dataset contains homogeneous samples within the same scale, this would cause no problems. However, in a real-world scenario, data is most of the time heterogeneous with outliers and incomplete sample values, introducing some form of diversity. Outliers must be determined and cleaned while the incomplete sample values be replaced by an appropriate interpolation method. In more complicated cases, the variables may have dependencies on each other which might not be clear to see in the first look. The datasets prepared for medical applications can be given as an example. Like in our case, generally include heterogeneous variables. This means that both numeric and categorical information exists at the same time. The most basic example of this can be given from Age, which is specified in integer format, and Gender, which is represented in a symbolic format. Feeding a neural network with this dataset as it is will not cause any compilation problems since neural networks can tolerate the variance and diversity in feature types via automated scaling properties. However, this usually results in longer train time and lowered generalization abilities. These issues may not be so obvious in smaller networks however, becomes significant as the network size increases and brings the necessity to apply some form of normalization in the preprocessing. According to an estimate provided in the paper by Maharana [45], the pre-processing phase often constitutes a significant portion, ranging from 50% to 80%, of the whole classification steps. This emphasizes the crucial role of pre-processing in model construction.

Data preprocessing can be defined as the initial formatting, cleaning, and scaling of the original dataset to optimize the model outcome. The input and the output of a preprocessing step must also obey some standards. Assuming that we are dealing with a transformation, T, from an original, unmodified sample set A to a formatted, modified new sample set B, [45], the transformation T must preserve the characteristics of each sample in the two sets. The transformation should be in the format but not in the meaning. This can be formulated as follows:

$$B_{ij} = TA_{ik} \tag{2.1}$$

Moreover, we can derive that the set A must not be equal to set B. This means that the transformation must be able to remove at least 1 of the related issues seen in the original set. We should also end up with a more helpful sample space after the transformation. It should make everything more smoothly, not more challenging.

2.1.2 Data Augmentation in Medical Applications

In many cases where the researchers face performance issues, the first thing they try to improve is the number of samples in their train dataset. As a rule of thumb in machine learning, the increase in the

sample space usually means an increase in the learning abilities of the model. Many researchers developed several data augmentation techniques for different kinds of data from natural language to images, to tackle the problem, which are still popularly in use today. *Data augmentation* can be described as the method of creating additional data originating from the existing data which follows the characteristics and the behavior of the source. The augmentation can be as simple as rotating and cropping images to more advanced techniques like applying transformations and adding noise to existing samples in machine learning-related applications such as image recognition. Natural language processing also commonly requires data augmentation. In NLP applications data augmentation is transferred to a more advanced level since the order of samples plays a critical role in the meaning of the word. In simpler words, our essential goal when opting for data augmentation is to be able to introduce diversity to the train set, hoping that the model learns more variations of the sample space and adjusts to a more generalized state. In this section, some popular data augmentation techniques that are also employed in this research work will be introduced.

> Note: This concept is referred to in section 3 in the generic preprocessing discussion.

2.1.2.1 Borderline SMOTE

During the development of prediction models, there are instances when datasets contain unbalanced class variables. To address the class imbalance in such datasets, the *Synthetic Minority Over Sampling Technique (SMOTE)* is proposed. SMOTE generates synthetic samples by leveraging the nearest neighbors, thereby illustrating minority scenarios. This widely used technique is often employed for data augmentation in classification problems. The proposed technique showcased herein demonstrates a system that combines regression and data augmentation. Synthetic data is produced from the original dataset by randomly selecting nearest neighbors, depending on the desired level of oversampling [9].

One of the limitations of SMOTE is that it fails to consider the spatial arrangement of neighboring majority class data when synthesizing minority class data, which can result in overlapping class samples. To overcome this challenge, researchers have focused on selectively performing oversampling or reinforcing the borderlines and their adjacent vicinity within the minority class. The typical steps involved in SMOTE are as follows:

Firstly, the m closest neighbors are identified among the samples belonging to the minority class. Following that, a set called "DANGER" is constructed by selecting more than half of these m neighbors, especially including the samples from the minority class that corresponds to the majority class. Following that, the algorithm identifies the s nearest neighbors from the elements in the DANGER set. The values of these neighbors are subsequently multiplied by the space between the sample and its closest neighbor. To further modify the result, a random number in the range of 0 to 1 is introduced as a scaling factor. The adjusted values are then added to the original sample. Lastly, the synthetic samples are generated using [17]:

$$\text{synthetic}_j = p_j + r_j^* dif_j, \text{ j=1,2...,s} \tag{2.2}$$

where p corresponds to the DANGER set, r provides a random number between 0 and 1, dif is for the difference in length for the s nearest neighbors of the selected sample. The final state of the sample set contains a balanced number of samples from each class and provides a possible solution to the imbalanced learning problem.

2.1.2.2 Image Transformations

Several widely used techniques for augmenting image data include applying transformations that map the image pixels to different positions or manipulating the intensity values. This process involves selecting one image from the original dataset, performing the transformation, and returning the modified image to the dataset to increment its total size. These methods are straightforward to apply, yet effective, in improving the performance of trained models. Among the 149 articles reviewed in the paper [19], 93 applied a basic augmentation approach.

The following are commonly employed image augmentation techniques:

- Geometric transformations: encompass a range of techniques that often maintain parallel relationships within an image. These transformations include scaling, translation, rotation, reflection,

shearing and perspective transforms, such as skewing. When an image contains contours, these geometric transformations are also applied to the associated contours. The parameters for these transformations can either be predefined or randomly sampled, offering flexibility in adjusting the transformation effects.

- Cropping: involves randomly selecting patches from an original image and reintroducing them into the dataset, thereby augmenting its size. This technique is commonly employed to tackle class imbalances by generating patches from under-represented classes, thus promoting a more balanced distribution within the dataset.
- Occlusion: In a manner related to cropping, this technique involves the removal of patches from an image to generate an augmented version with altered content.
- Intensity operations: These operations manipulate pixel/voxel values within an image, enabling adjustments in brightness or contrast. Popular methods encompass gamma correction, linear contrast adjustment, and histogram equalization.
- Noise injection: The augmentation of noisy images involves the addition of Gaussian noise, where image intensities are randomly sampled from a Gaussian distribution. Other types of noise, such as uniform noise from a uniform distribution, and salt and pepper noise, where pixels are randomly set to black or white, are also employed.
- Filtering: By applying convolution with a specialized kernel, intensities within an image can be adjusted based on the values of neighboring pixels. This process allows for image sharpening, blurring, or smoothing, while also enhancing the edges of objects within the image.

2.1.2.3 GAN-based Augmentation

GAN-based image synthesis presents several key challenges, particularly to ensure training stability and generate images of exceptional quality, clarity, and high resolution. To overcome these hurdles, researchers have introduced numerous GAN variants with improved network architectures and refined mathematical optimization techniques. These advanced GAN networks facilitate domain adaptation, empowering trained models to effectively transfer and adapt knowledge from one data source to another. This capability allows for the creation of new data while preserving the model's ability to embed adaptations and shifts between different sources of data [19].

2.1.3 Data Augmentation of Time Series and Why It is Different Than Others

Utilization of data augmentation [28] in deep neural networks is a valuable technique that involves generating artificial data. This process effectively reduces classifier variance and minimizes errors, contributing to improved model performance. This technique has proven effective in enhancing the generalization capabilities of deep neural networks across various computer vision tasks, including image recognition and object localization. While deep learning thrives on large training sets, obtaining such extensive data may not always be feasible in real-life scenarios.

One well-established approach to data augmentation is the generation of synthetic data. For instance, slight rotations can be applied to images without altering their representative feature information. Additional techniques encompass injecting random noise, performing slicing or cropping, applying scaling transformations, introducing temporal warping, and employing frequency warping. While these methods work well for image augmentation, they do not yield satisfactory results when applied to time series datasets. The main challenge lies in the difficulty of assessing the impact of ad-hoc transformations on the nature of a time series, as it is not as visually apparent as in images.

To address this issue, generative models provide an alternative approach by leveraging feature distributions within the sample space to form new patterns [39]. To fulfill this objective, researchers have suggested plenty of techniques, which include a variety of statistical models such as Gaussian trees, as well as handcrafted mathematical models. The popularity of generative models for time series augmentation has been on the rise, with notable examples being Generative Adversarial Networks (GANs). Another approach involves decomposition methods, which extract features such as trend components and independent components from the dataset and generate new patterns based on these extracted features. The advantage of these methods lies in their ability to preserve the underlying distribution of time series data, unlike random transformations that may inadvertently alter the distribution.

> Note: This concept is referred to in section 3 within an specific to preprocessing discussion.

2.2 Related Artificial Intelligence Techniques

Artificial Intelligence (AI) refers to the ability of machinery to exhibit expertise in specific tasks. The realm of AI encompasses a wide array of examples, real-time translation of natural languages, automated financial trading systems, and the development of self-driving cars. Any technique capable of emulating human intelligence can be classified as part of AI. It has many subclasses which can be seen in the following figure:

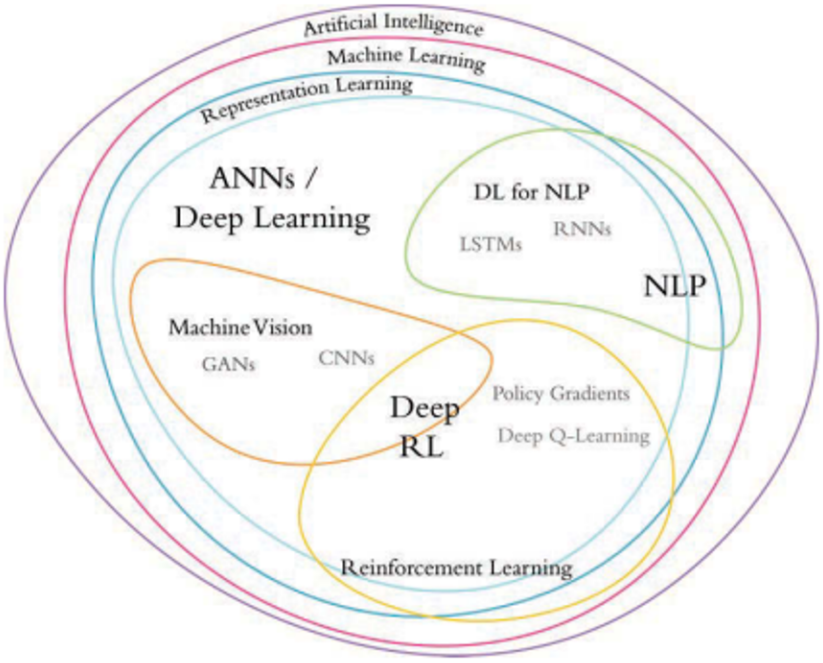


Figure 2.2: Figure showing the relationship between different artificial intelligent subclasses. Image Retrieved from: [1]

2.2.1 Machine Learning

Machine learning [12] can be described as a subfield of AI dealing with the development of computational algorithms to learn from the surroundings to achieve a set of design goals within a utility framework. There is more than one definition of machine learning. According to the author of the paper [46], Arthur Samuel, machine learning can be expressed as " the field of study that empowers computers to acquire knowledge and enhance their performance without explicit programming instructions". The methods employed in this field can be applied to many other fields such as :

- Pattern Recognition
- Finance
- Healthcare
- Computer Vision
- Entertainment . . .

It is a very problem-specific field and the algorithms developed highly vary depending on the issue targeted, the variables of the problem, the models that may better fit than the others, etc. These algorithms can change and adapt their architecture by gaining experience via repeatedly performing the same tasks to fit better to the specified goal. This process is called *training* that may be implemented in two commonly known ways, namely supervised and unsupervised learning. In supervised learning,

the model is allowed to see samples with their labeled outcomes. In other words, a supervisor informs us about the correct labels of data samples in the training dataset. In unsupervised learning, class labels of data samples are unknown initially and they are determined by spatial distribution of data samples concerning each other. Consequently, any learning algorithm as a result adapts itself not only to understand the architecture of training samples but also to generalize its configuration for the unseen data to achieve the desired outcome. Training a model with machine learning offers two key advantages :

- It can replace the workload required by humans by automating the repetition required in training.
- It can learn complex structures in the sample space in a better way than what a human observer can do.

Supervised Learning

Supervised learning can be described as one of the machine learning tasks which links an input sample to an output label considering the provided relation between the two by learning a function. These algorithms require external assistance, just like being supervised by a teacher, that is why they are referred to as "supervised" learning algorithms. The input samples are usually split into two sets by some ratio. Some popular algorithms use 70:30 or 80:20. The larger set is used for training where the labels of data samples are expected to be predicted and the success is measured according to predefined performance metrics. The other set is used for testing. The discriminative learning model that is explained in detail in the coming sections also falls under this category.

Unsupervised Learning

Unlike supervised learning described above, there are no correct labels for data samples and there is no teacher. The models try to form clusters or groups that behave similarly to the input samples without having any access to the output labels. These labels are left to be discovered by the algorithms. The unseen data is classified by comparison to the previously learned features and it is placed to the most resembling class.

Semi-Supervised Learning

This approach is a mixture of the two previously mentioned approaches: Supervised and Unsupervised Techniques. It is a good option to select in cases where there are unlabeled samples and extracting the labels is tiresome work. Some of the generative deep learning methods fall under this category.

2.2.2 Basic Deep Learning Architectures

> Note: This concept is referred to in sections 3. 6 and 4 within the employed machine learning model and along with discussions on already published approaches.

Deep learning [64] is a subfield of machine learning that aims to capture intricate data abstractions through the use of multi-layered neural networks. These networks comprise complex structures or nonlinear transformations and are capable of modeling high-level abstractions of the sample space. Deep learning involves the employment of advanced architectures and mostly non-linear transformations within these multi-layered networks, resulting in a neural network with a substantial number of layers and parameters. Commonly, deep learning methods utilize neural network architectures, that contain specialized units for filtering the input data at different scales and a classification unit that determines the class labels based on the filtered features. In summary, deep learning harnesses a series of cascading layers of nonlinear processing units to extract and transform features from the data.

Deep learning, is a specialized field within machine learning, that focuses on constructing models with multiple layers of neurons. These layered structures enable the modeling of complex data abstractions and facilitate non-linear transformations. Neural networks are simple models of learning and information processing procedures inside a human brain through constructing and remembering the links and relations between input samples. The term "artificial neuron" describes a simple model of its counterpart, the organic neuron. Neural networks modify the values of their architectural elements, such as connection weights and filter parameters to model the alterations in the input and the availability of highly successful optimization algorithms for adjustment of model parameters makes it possible to provide the best fitting solutions. It usually takes 3 layers to construct a basic neural network.

- The input layer receives the input data.

- The hidden layer(s) process the input through various computations.
- The output layer transmits the computed output.

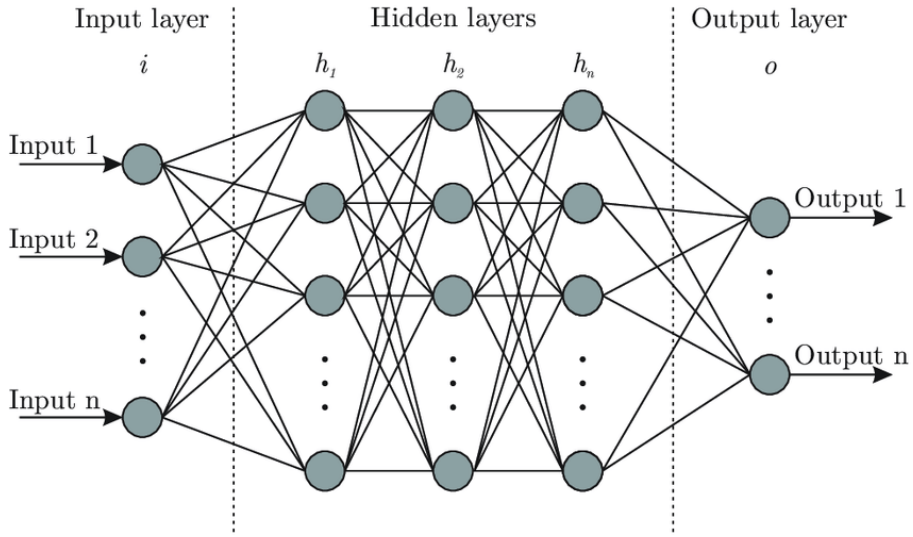


Figure 2.3: Figure showing the structure of a neural network. Image From: [8]

Deep learning networks typically consist of five or more layers, adhering to the following structure:

- An initial input layer is designed to feed the input data into the network.
- Three or more hidden layers that specialize in learning representations from the input data. Among the various types of hidden layers, *Dense* layers are particularly popular due to their versatility. In Dense layers, the neurons are considered to be *fully connected*, enabling unidirectional information flow. Neurons in higher layers receive data from the preceding layers.
- A final output layer is devoted to the values or predictions generated by the network.

As mentioned previously, the level of abstraction increases as the number of layers increases. With this increment non-linear recombinations received from lower layers are represented in a more and more abstract manner.

2.2.3 Deep Learning in the Context of Machine Vision

Computer vision [25] can be described as a field of machine learning specific to images and video. This field teaches machines to visually understand and interpret information by translating visuals through, extracting features and other important characteristics of the data learned during the training phase. This capability empowers models to analyze images and videos and leverage those interpretations for predictive or decision-making purposes. When contrasting deep learning with conventional machine vision techniques, the most significant difference arises in the context of feature extraction.

In traditional methods [21], it is the responsibility of the vision engineer to determine the specific features to seek to detect a particular object within an image. Moreover, they must identify the appropriate set of features for each class. However, this approach becomes impractical when dealing with a large number of classes. The information we look for, such as edge and texture information, most of the time must be fine-tuned by the data specialist. As a solution, deep learning methods provide a technique known as *"End to End Learning"* where the algorithm taught what to search for within the given classes. During the training phase, it identifies what are the most significant and prominent features in each class.

2.2.3.1 Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GANs) [23] are a cutting-edge approach that holds promise for both semi-supervised and unsupervised learning. Their main strength lies in their ability to implicitly model high-dimensional data distributions. In its simplest form, GANs contain two networks, namely the generator and the discriminator, designed to compete with one another. A simple example from the paper [23] explains the concept more visually:

In the context of art, one of the networks can be expressed as an "Art Fraud" trying to create realistically fake products. In the context of GANs, this is known as a generator, G, trying to create almost real fake data samples. On the other hand, in the context of art, the other network can be expressed as an "Art Expert" trying to identify and discriminate between real and fake products. In the context of GANs, this network is referred to as a discriminator, D, which accepts both fake and authentic data samples and tries to separate them. They are trained at the same time and compete with each other. The discriminator has both fake and real samples, however, the generator does not know anything about the real images and the only way it can fake the behavior of authentic data is by its communication with the discriminator. The performance of the discriminator is hence obtained via an error signal which is generated by considering whether the image is coming from a real or synthetic stack. The same error signal, via the discriminator, can be used to train the generator, leading it toward being able to produce forgeries of better quality.

If we express the behavior of the generator as a link between some representative space, called a latent space, to the input sample space, then we can formalize the behavior of the generator as follows:

$$\text{G: } G(z) \rightarrow R^{|x|} \text{ where z } \varepsilon R^{|z|} \tag{2.3}$$

is a sample from the latent space,

$$\text{x } \varepsilon R^{|x|} \tag{2.4}$$

is an image and

$$|.| \tag{2.5}$$

denotes the number of dimensions. The discriminator, D, can be described as a function computing a probability on the supplied image, resulting in whether or not it is authentic or not as its output is close to one or close to zero:

$$\text{D: } D(x) \rightarrow (0, 1) \tag{2.6}$$

Since these two models are competing with each other, at some point in the training when the discriminator reaches its optimal performance, it can be stopped and the generator can be given a competitive advantage by training it more. This is to lower the accuracy of the discriminator to prevent overfitting. Similarly, if the generator reaches its optimal performance, meaning that it can generate real-like samples, this will confuse the discriminator causing it to provide predictions with lower accuracies. The cost of training is evaluated using a value function,

$$_{D}max_{G}minV(G, D) \tag{2.7}$$

that depends on both the generator and the discriminator. Here, this definition describes the objective of maximizing the discriminator's success while minimizing the generator's performance by faking the discriminator. The training involves solving this equation where,

$$V(G, D) = E_{p_{data(x)}} logD(x) + E_{p_{g(x)}} log(1 - D(x)) \tag{2.8}$$

where $E_{pdata(c)}$ stands for the expected probability of the real data samples x, $D(x)$ stands for the discriminator's success for real data samples x, $E_{p_{g(x)}}$ stands for the expected probability of the generator's success on generating real-like fake samples, and $(1 - D(x)$ is the failure rate of discriminator due to the fake samples of the generator.

We can prove that for a fixed generator, there is a unique optimal discriminator,

$$D^*(x) = p_{data(x)}/(p_{g(x)} + p_{data(x)}) \tag{2.9}$$

and the generator, G, is optimal when

$$p_{g(x)} = p_{data(x)} \tag{2.10}$$

causing discriminator confusion and resulting in lower discrimination accuracy. In an ideal situation, we would like to train the discriminator to an optimal point comparing it to the performance of the generator but this may not be applicable in practice as explained below.

Training a Generative Adversarial Network (GAN) poses several challenges:

- Ensuring convergence of the model pairs.
- Preventing the generative model from "collapsing" and producing similar samples for different inputs.

- Avoiding the discriminator loss from quickly converging to zero, which results in sending in gradient inaccurate updates to the generator.

To address these issues, heuristic approaches can be employed to deal with these difficulties. GANs [78] play a significant role in the advancement of generative models. GANs, known as a potent category of generative techniques, excel in the task of producing data that can be easily comprehended and interpreted. They are particularly suitable for high-dimensional feature space, as the generator model architecture is not limited to any specific dimension. The flexibility of GANs allows developers to incorporate various structures and loss functions, greatly enhancing the degree of freedom in model design. By employing adversarial training, GANs facilitate the production of varied and meaningful data, steering clear of direct replication or simplistic averaging of real data. In essence, GANs offer a promising solution for generating data that is not only creative but also holds significance for human interpretation. A figure illustrating the architecture of a GAN is given below.

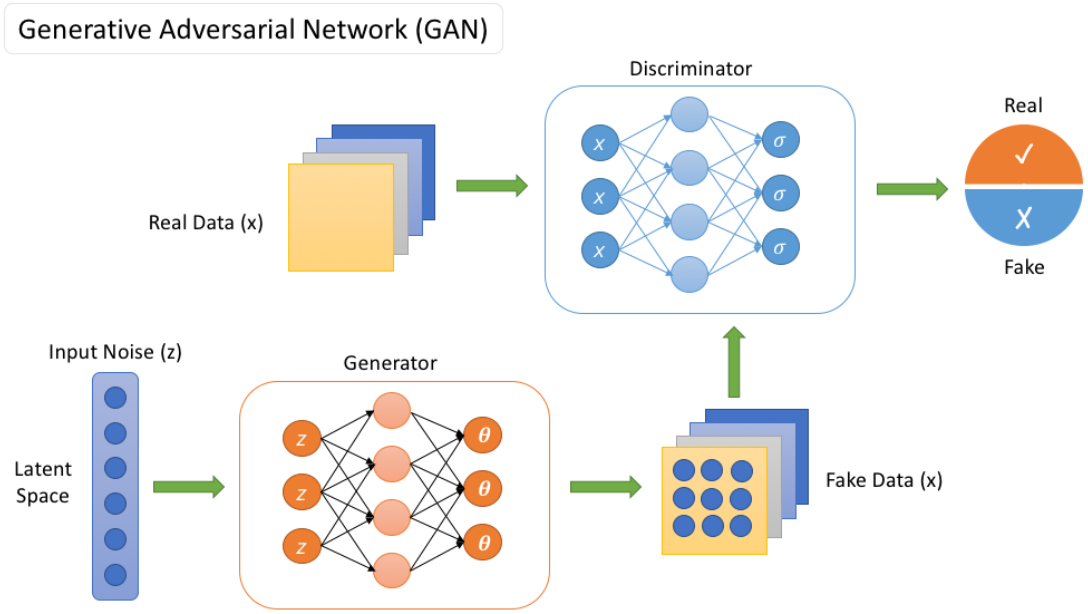


Figure 2.4: Structure of a generative adversarial network (GAN). Image From: [76]

2.2.3.2 Convolutional Neural Networks (CNN)

Convolutional neural networks (CNNs) belong to the category of feed-forward artificial neural networks, distinguished by their utilization of convolution operations in one or more of their layers. These operations play a crucial role in extracting and capturing meaningful features from the input data. They draw inspiration from biological neural networks and combine neural network architectures with discrete convolution structures to automate feature extraction. As a result, CNNs are specifically designed for recognizing and analyzing two-dimensional data such as images and videos.

When the input data for a CNN is flattened into one-dimensional vectors to serve as inputs, this flattening process leads to a significant loss of meaningful visual image structure. Another consideration is the increase in computational complexity when applying this process.

On the other hand, when dealing with moderately sized images, such as a 200x200-pixel full-color RGB image, the number of parameters to be optimized dramatically increases. For instance, each neuron in a dense layer would have 120,000 parameters when considering three color channels and 40,000 pixels. If we include an average number of neurons, such as 64, this results in an astonishing 8,000,000 variables in just the first layer of the network. When dealing with multiple data points in a full image, this naive approach usually results in low accuracy scores due to a very large number of variables as input to the gradient-descent-based optimization algorithms.

Convolutional layers in CNNs consist of sets of kernels or filters that examine the image in smaller patches. This process, known as filter convolving, involves weights within the kernels being learned through a gradient-descent-based backpropagation algorithm. Typically, kernels have a size of 3x3. In

the context of full-color RGB images, a kernel that spans the same number of pixels would entail three times more weight variables. This arises from the fact that RGB images consist of three color channels (red, green, and blue), necessitating additional weights to account for the information contained within each channel. As the kernel traverses and convolves over the image, a weighted sum is calculated at each position using the equation $wx + b$. The resulting value, denoted as z, is passed through an activation function such as $tanh$ or $ReLU$ to obtain the final activation value, a.

Unlike fully connected layers, convolutional kernels have shared weights across all inputs. This weight sharing reduces the number of weights in a convolutional layer significantly compared to a fully connected layer. As a result, a convolutional layer can possess significantly fewer weights, often differing by orders of magnitude.

In most networks, there are multiple filters in a convolutional layer. Each filter allows the network to identify representative features at a given layer. These filters scan the image, detecting specific patterns, shapes, and colors they are tuned to recognize. The outcome of the filtering process is referred to as an activation map. The activation maps generated by the preceding convolutional layers are passed on as inputs to the subsequent convolutional layers. By adding more layers and deepening the network, the degree of abstraction and the number of extracted complex features and patterns increase. These features can range from simple lines, colors, and textures to whole objects as the network becomes deeper.

The number of filters within a convolutional layer, just like the neuron count in a dense layer, is a configurable hyperparameter that can be fine-tuned to enhance the network's effectiveness. Considerations when determining the number of filters include the complexity of the data and the problem being solved. It should be noted that the appropriate number of kernels for a particular layer can vary considerably. In the initial layers, the focus is on recognizing basic features, while in subsequent layers, the emphasis shifts to detecting more complex compositions and patterns formed by these features. As a rule of thumb, later convolutional layers tend to have more kernels relative to early convolutional layers in machine vision applications.

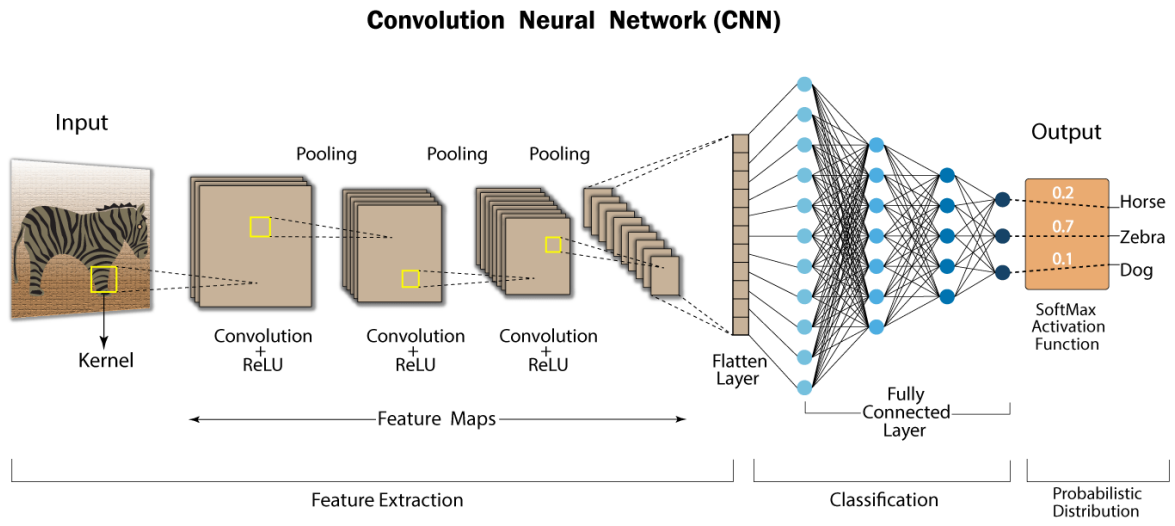


Figure 2.5: The structure of a convolutional neural network. Image from: [68]

2.2.4 Recurrent Neural Network Architectures

Natural Language Processing (NLP) [26] is a field of study that explores the application of computers to comprehend and process human languages, enabling them to perform various tasks. The objective of NLP, from a scientific standpoint, is to model the cognitive processes involved in understanding and generating human languages. By developing computational models, NLP seeks to unravel the underlying mechanisms behind human language comprehension and production. On the other hand from an engineering point of view, NLP deals with human-computer interaction and the construction of such algorithms to facilitate this process. Some popular application areas of NLP contains:

- Machine Translation
- Sentiment Analysis

- Lexical Analysis
- Social Computing . . .

Natural language is a symbolic and discrete system meticulously designed to communicate meaning or semantics. By its inherent nature, it is constructed to convey information through symbols and discrete units, enabling the exchange of complex thoughts and ideas between individuals. The physical form of natural language which we interact with also referred to as *"text"*, is always in a symbolic representation. *"Speech"* on the other hand, is a continuous latent holding the linguistic characteristics of NL, just like text. Machine learning applications employed by traditional methods need human expertise when it comes to designing feature spaces and this causes a significant bottleneck for the engineers. Additionally, the formed models usually are insufficient to successfully achieve an abstraction that would decompose complex descriptive features in the symbolic language data and hence limited computational power in terms of representation. One possible solution to tackle this could be increasing the depth of neural networks via deep learning, taking advantage of its layered structure. Deep learning has made notable strides in the field of natural language processing (NLP) with two key innovations: sequence-to-sequence learning and attention modeling. These advancements have significantly improved the performance and capabilities of NLP systems, enabling more accurate and effective language processing tasks.

Sequence-to-sequence learning, also known as seq2seq, is introduced to solve complex language problems where the model receives a sequence of samples and outputs a sequence of samples. This approach presents the concept of recurrent nets to perform encoding and decoding with an end-to-end mode. On the other hand attention modeling tries to divide the complex problem into sub-problems, also known as areas of attention, and identifies each of them sequentially. The main goal is to resolve the problems coming along with the encoding of long sequences. The advancements in both approaches allowed researchers to employ the best systems based on statistical learning and local visualizations of words through distributed embedding translations and optimized network performances. This allowed them to achieve accuracy levels almost as good as humans, if not better. However, they need a lot more training samples and computational power needs than what humans require.

2.2.4.1 Recurring Neural Networks (RNN)

Recurring Neural Networks (RNNs) [34] fall under the class where the input sample to be processed is in sequential form. In a convolutional neural network, which we explained in the above section, the processing starts at the input layer, flows in between the hidden layers, and finally exits through the output layer. The neurons at the same level do not communicate or send or receive information with each other. This brings limitations on specific cases where the input samples have relations with themselves. For example, the words in an average sentence are linked with each other and we require the earlier words to predict the later words. Since the words are not independent of each other, they require a network structure like RNNs. By leveraging the concept of recurrence, recurrent neural networks (RNNs) incorporate the output from the previous time step into the computation of the current output. Consequently, the input of the network encompasses not only the data from the input layer but also the output of the hidden layers derived from the preceding time step. This recursive nature empowers RNNs to effectively capture temporal dependencies and handle sequential data.

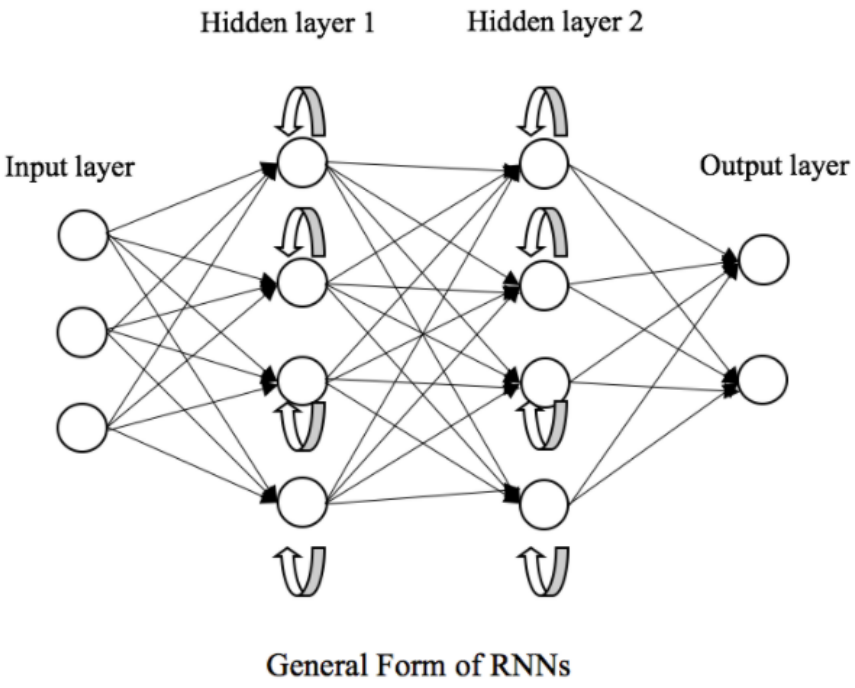


Figure 2.6: The structure of a recurrent neural network. Image From: [22]

As opposed to CNNs, the parameters at different layers in RNN are shared. They are not discrete and different in each layer. This means that as we move along in the RNN hierarchy, we see the same processing operation performed using different inputs. This way the network no longer has to spend an additional effort on relearning the parameters.

A gradient [20] can be described in terms of partial derivatives of a function concerning its free variables, and also can be thought of as a slope function, concerning the given inputs. It calculates the degree of change in the function value concerning the changing inputs. As the gradient increases, the slope becomes more upright and the trained model starts to learn quicker. A slope of zero indicates that the model has stopped learning. Hence, in a machine learning context gradient can be expressed as a function that calculates the change in model weights in correspondence to the change in error. Unfortunately, RNNs face some gradient issues. The gradient issue occurs when the value reaches the minimal and maximal extremes during training, making it very hard for the model to learn. As a result, the following challenges arise:

- Sub-optimal performance
- Reduced accuracy
- Lengthy training duration

If the gradient values become too big, exceeding the foreseen safe limits, the calculated slope will start to increase in an exponential behavior resulting in the exploding gradient problem. This issue is usually because of giving way too much importance to the weights. These issues can be addressed through the implementation of the following techniques:

- Identity initialization: Initializing weights to resemble an identity mapping for stable training.
- Truncated back-propagation: Limiting back-propagation to address gradient issues in recurrent neural networks.
- Gradient clipping: Scaling down gradients to prevent large values and stabilize training.

Another issue is the vanishing gradient. Opposed to the exploding gradient problem, the values of a gradient fall below the safe limits, and the model stops learning or takes way too much time to process information. These issues can be addressed through the implementation of the following techniques:

- Weight initialization
- Selection of appropriate activation function

- Utilizing LSTM (Long Short-Term Memory) networks to mitigate the vanishing gradient problem.

2.2.4.2 Long Short-Term Memory (LSTM)

LSTM (Long Short-Term Memory) [34] is a widely adopted technique in addressing the limitations of traditional RNNs. While RNNs are effective in capturing relationships and dependencies within input data, they often struggle with long-term dependencies. LSTM is specifically designed to tackle this issue by introducing specialized components known as blocks.

Within each LSTM block, there is a cell that serves as a memory unit, capable of retaining information from all the input data. Additionally, a component called the "forget gate" is employed to selectively discard irrelevant information from the cell, ensuring that only important information is retained and utilized during processing. This enables LSTM to effectively preserve and leverage long-term dependencies in the data.

The flow of information is stored and identified with the existence of additional gates in the LSTM. These gates try to identify whether received information holds important features and should be stored for future use or should be removed completely. In an LSTM, there are three important types of gates: input, output and forget gate. The first two gates' work is similar to the previous models. The forget gate is dedicated to identifying unimportant data and eliminating it from the neurons. This increases the network performance. The structure of this gate contains two inputs, received from the earlier cells and the current cell. The bias and model weights are calculated and the resulting value is supplied to the sigmoid function. This result will tell us whether or not to keep the information (1) or remove it (0).

In its simplest form, the input gate is used to add information to neurons. It prepares the information, puts them in an array format, and uses activation functions, like the sigmoid function we mentioned earlier to act just like a filter and regulate the flow, to determine whether information should be fed to the network or not. The preparation phase uses an activation function named tan hyperbolic ($tanh$). This function generates results between -1 and 1.

Finally, the output gate in a recurrent neural network (RNN) is responsible for selecting relevant information from the current cell and presenting it as the output. It controls the flow of information from the hidden state to the final output, allowing the network to focus on important information for downstream tasks. It uses the $tanh$ activation function with the earlier outputs and current inputs to filter and regulate which outputs should be forwarded as an outcome of the network.

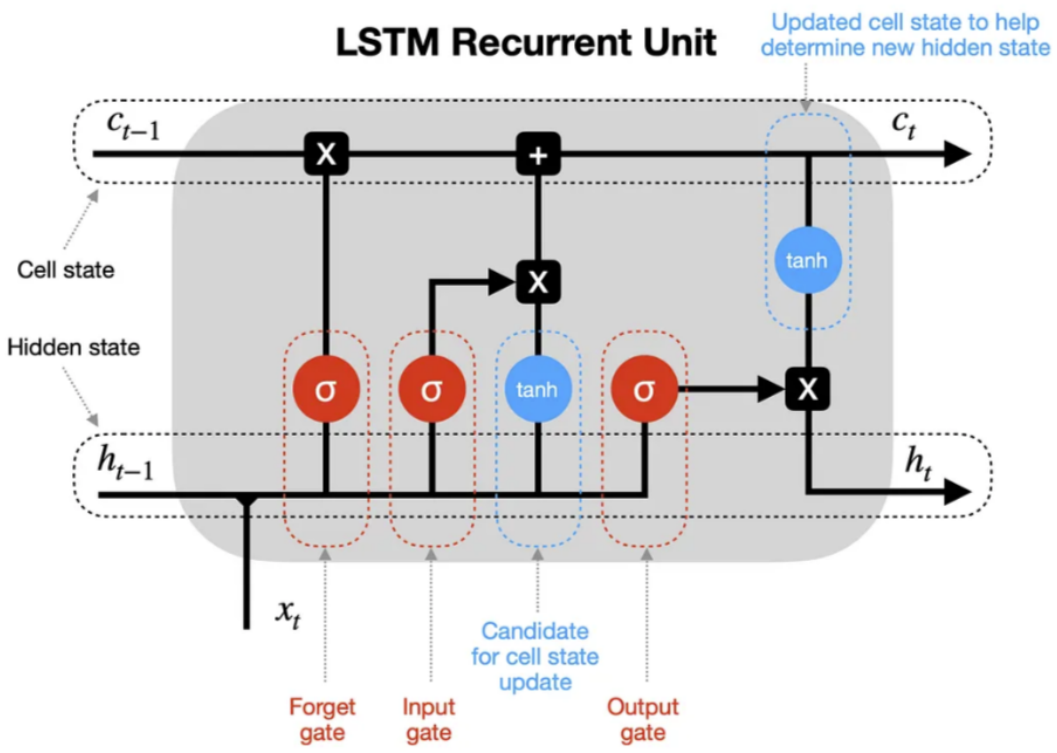


Figure 2.7: The structure of a long short-term memory (lstm) model. Image From: [27]

2.2.5 Ensemble Learning

Ensemble learning is a powerful approach in computational intelligence, where multiple models or classifiers are strategically generated and combined to address a specific problem. This thesis also employs ensemble learning by developing a separate model for each measured observation and relying on their combined decision. Researchers often turn to ensemble learning for several reasons:

- Improved Model Performance: Ensemble learning has been proven to enhance the overall performance of the models involved. By combining the predictions or decisions of multiple models, the ensemble can leverage the strengths of each model, leading to more accurate and robust results.
- Mitigation of Poor Model Selection: Ensemble learning helps mitigate the risk of selecting a single poor model that may struggle to identify or generalize well in real-life scenarios. By combining different models, the ensemble can compensate for individual model weaknesses and provide a more reliable solution.

Ensemble learning finds applications in various domains, including:

- Assigning Confidence to Model Decisions: Ensembles can estimate the confidence or reliability of their decisions, providing valuable insights into the uncertainty of predictions.
- Optimal Feature Set Selection: Ensembles can assist in selecting the most informative and relevant features from a large feature set, enhancing the efficiency and effectiveness of the learning process.
- Data Fusion: Ensembles excel in integrating information from diverse sources or modalities, resulting in a more comprehensive and accurate representation of the underlying data.
- Error Correction: Ensembles can identify and correct errors made by individual models, leading to improved overall performance and more reliable predictions.
- Ensemble learning encompasses many other applications and techniques, making it a versatile and widely adopted approach in machine learning.

2.2.6 Image Encoding Techniques For Time Series

Note: This concept is referred to in detail in section 3 when presenting the recurrence plots.

Time-series [11] data show different characteristics than discrete and stable samples. It provides important features which describe the changes in the given sample space and the variations involved in the process. When we observe how quickly these changes occur, we obtain many insights about the evolution of the event. When the degree of change is too granular, it becomes extremely hard for the researchers to develop models that can represent such sample spaces. Additionally, in an ideal scenario, we expect the observations of a time series to be taken at equally separated intervals but this is often not the case in real life. The successive observations are not measured in equal intervals and further processing has to be applied to normalize the space. This problem is also seen in our dataset which we have used in this thesis work.

On the other hand, measurements presented in time series format are usually in very long real-valued vector forms. When the measurements are taken from multiple sources, the dataset to be processed constitutes several long vectors, with possibly annotation labels, to be processed and classified accurately. Considering the processing of these long vectors as inputs to deep learning models, accuracy drops rapidly due to a very large number of network parameters to optimize and due to the lack of representing consecutive data dependencies when a whole time series is considered as one single block of signal values.

One potential solution to address this problem is to encode time series data as images, enabling machines to visually recognize, discriminate, and understand intricate structures and patterns at a more granular level. This transformation leverages the power of image processing techniques to extract meaningful features from the time series data, facilitating improved analysis and decision-making by machine learning models. Transformation of features of such series to visual representations and building network architectures for inspection and reformulation of distance measures are gaining more and more popularity in machine learning-related applications [79].

Deep learning approaches for multi-variable classification problems involving a sample space consisting of time series information can be classified into two main types: generative models and discriminative models. Generative models aim to capture the underlying probability distribution of the data and generate new samples, and discriminative models focus on learning the decision boundaries between different classes to enable accurate classification.

Generative models, also referred to as model-based classifiers, try to find a good formulation of the given input space to train an unsupervised model for classification purposes. Oppose to that discriminative models employ feature engineering and try to find a link between provided inputs and related labels via tuning. In the following section 3, we will delve into the explanation of a discriminative deep learning model that utilizes an image-based representation of time series data [81].

In this research, the following time series to image transformation technique will be used and its appropriateness within the context of this study will be explained in detail.

2.2.6.1 Recurrence Plots

Recurrence plots (RP) are a two-dimensional representation of a time series in phase space, providing insights into periodicity, chaos, and non-stationarity. It reveals the internal structure of the data and offers prior knowledge about similarity, information, and predictability. RP is particularly useful for time series data since it aims to represent consecutive variations between data samples. The encoding process of RPs involves the following steps:

Time-delay embedding: The time series

$$X = [X_1, X_2, X_3, ..., X_N] \tag{2.11}$$

is reconstructed in a two-dimensional phase space using the time-delay embedding method. The state of the phase space with a time delay of 1 is expressed as

$$S_1 = (X_1, X_2), S_2 = (X_2, X_3), ..., S_{N-1} = (X_{N-1}, X_N) \tag{2.12}$$

.

Phase space state representation: If the embedding dimension is denoted as m and the time delay as d, the state of the phase space is represented as

$$S_i^m = (X_i, X_{i+d}, X_{i+2d}, ..., X_{i+(m-1)d)}) \tag{2.13}$$

Recurrence plot generation: The recurrence plot

$$R_{i,j} = \Theta(\epsilon - |s_i - s_j|) \tag{2.14}$$

is constructed based on the distances between pairs of embedded vectors.

Rij is calculated using the Heaviside function, comparing the distance between s_i and s_j to a threshold value of ϵ. The resulting recurrence plot is a rectangular array of pixels, where the color of each pixel corresponds to the magnitude of data values.

Thresholding: The critical radius ϵ is used to determine whether a point is plotted as a darkened pixel in the recurrence plot. Only distances below or equal to epsilon are represented as darkened pixels.

To retain more image details through color transformation, the Heaviside function is usually not used. Different norms, such as the infinite norm, can be employed to calculate the distances. Additionally, an un-thresholded recurrence plot (UTRP) can be obtained by using the maximum value of the distance instead of thresholding.

$$R_{i,j} = \epsilon - max_{1 \leq k \leq m} \left| X_{i+(k-1)d)} - X_{j+(k-1)d)} \right| \tag{2.15}$$

This corresponds to the unthresholded RP(UTRP).

To further analyze the data, a thresholded recurrence plot (TRP) matrix B is generated by applying a threshold corridor

$$[\delta_l; \delta_h] \tag{2.16}$$

. This is done by generating a thresholded recurrence (TRP) matrix B:

$$B_{i,j} = 1 \text{ if} \delta_l \leq D(y_i, y_j) \leq \delta_h \tag{2.17}$$

$$= 0, \text{ otherwise} \tag{2.18}$$

. The TRP is obtained by darkening the pixels

$$(i, j)$$

corresponding to nonzero entries in matrix B. The choice of the threshold corridor is crucial, as a corridor that is too large may result in saturation of the TRP, while a narrow corridor may not provide sufficient points for subsequent analyses.

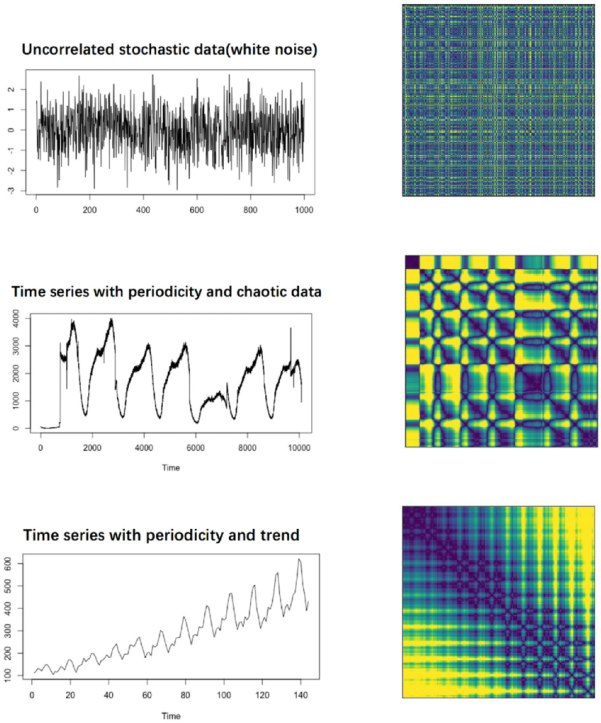


Figure 2.8: Examples of different recurrence plots. Image From: [2]

2.2.7 Model Metrics

In this section, different metrics used to help and boost the performance of machine learning models will be introduced.

> Note: This concept is referred to in section 3 where we discuss how we constructed and optimized our models and their parameters.

2.2.7.1 Optimizers

An optimizer can be defined in terms of a function of parameters, such as connection weights, and method-dependent parameters, such as the learning rate, that is used to reduce training loss and increase overall accuracy. The optimizers can be fine-tuned with different settings or Hyperparameters such as learning rate (lr), momentum, decay rate, etc. The optimizers can work pretty efficiently alone however, when tuned through their method-specific parameters, like lr, they boost the performance metrics of the model under training. Below are some of the most popular optimizers:

- Stochastic Gradient Descent (SGD) [59] is a popular optimization algorithm used in machine learning. It is based on the concept of gradient descent, which involves computing the derivative of the loss metric concerning the model parameters and updating them in the direction of the negative gradient. This process allows the model to adapt its parameters during training. While gradient descent is straightforward to implement, it can be slow to converge when the loss function has many local minima. To mitigate this, stochastic gradient descent introduces stochasticity by randomly selecting a subset of training examples (called a mini-batch) to compute the gradients at each iteration. This randomness helps in escaping local minima and leads to faster convergence. The SGD algorithm involves the following steps:
  - Obtain the number of training examples.
  - Iterate over a fixed number of iterations.
  - At each iteration:
    * Make predictions using the current model parameters.
    * Calculate gradients using the provided model and the selected mini-batch of training examples.
    * Update the model parameters by taking a step in the direction of the negative gradient, scaled by a learning rate.

  By repeating this process for multiple iterations and gradually adjusting the model parameters, SGD aims to minimize the loss function and improve the model's performance on the training data. SGD is a subcategory of gradient descent algorithms and hence the parameters inside the model are not adjusted in every epoch but with every individual batch. This way the model achieves a considerably quicker convergence however, it also introduces a level of instability to the optimization. This optimization method is very popular among problems involving large datasets. The mechanism with which SGD updates model parameters can be described with the following equation.

$$\Theta = \Theta - \alpha \Delta J(\Theta; x(i); y(i)), \text{where x(i) ,y(i) are the training examples.} \tag{2.19}$$

- Adaptive Moment Estimation (Adam): Adam [59] is another very popular optimizer. It tries to utilize the benefits coming with SGD. The learning rate (lr) is adjusted based on the first two moments of the gradients. Adam optimizer is particularly suitable for domains that involve a large number of parameters, as it has relatively lower memory requirements compared to other optimizers. The primary objective of the Adam optimizer is not to converge quickly, but rather to find a local minimum and gradually decrease the search speed, allowing for a more refined optimization process. One key aspect of Adam is that it maintains two averaged values from previously calculated gradients: $M(t)$, which represents the first moment (mean) of the gradients, and $V(t)$, which represents the second moment (uncentered variance) of the gradients. These values help in estimating the momentum and scaling of the optimization process. By storing and utilizing the averaged values of gradients, the Adam optimizer provides a more effective and efficient search for the optimal solution. It strikes a balance between exploration and exploitation, allowing for

a smoother optimization process and enabling the model to converge towards a local minimum effectively. The way through which the Adam optimizer adjusts the model parameters is given in the following equations.

$$m_t^- = \frac{m_t}{(1-\beta_1^t))} \tag{2.20}$$

$$v_t^- = \frac{v_t}{(1-\beta_2^t))} \tag{2.21}$$

$$\theta_{t+1}^- = \theta_t - \frac{\eta}{(sqrt(v_t^-)+\varepsilon)} m_t^- \tag{2.22}$$

- Adagrad [61]: Adagrad is an optimization algorithm that adjusts the learning rate on a per-feature basis. This means that each weight in the model can have a different learning rate, allowing the algorithm to adapt and prioritize updates based on the importance of each feature. By individually adapting the learning rates, Adagrad can effectively handle sparse features and focus more on infrequent or informative ones during the training process. Adagrad is an excellent selection when the dataset in hand contains a lot of lost and incomplete entries. The biggest disadvantage of Adagrad is originated from its biggest advantage. Over time, the learning rate is decremented so much that the training progresses so slowly. One way of overcoming this issue is to store the sum of squared gradients during optimization (g). To adjust the model parameters, Adagrad follows the equations listed below.

$$g_0 = 0 \tag{2.23}$$

$$g_{t+1} \leftarrow g_t + \Delta_\Theta \iota \Theta^2 \tag{2.24}$$

$$\Theta_j \leftarrow \Theta_j - \varepsilon \frac{\Delta_\Theta \iota}{\sqrt{g_{t+1}} + \epsilon^{-5}} \tag{2.25}$$

  The paper [61] explains the following equation in more depth: To understand the rationale behind AdaGrad, let's consider a two-dimensional space with a loss function. In this scenario, one direction may have a very small gradient, while the other direction exhibits a significantly higher gradient. When we sum up the gradients along the axis with small gradients, the squared sum of these gradients becomes even smaller. By dividing the current gradient by a small sum of squared gradients "g" during the update step, the resulting division yields a larger value for the axis with small gradients and a smaller value for the axis with high gradients. Consequently, the algorithm ensures that updates are made proportionately in all directions, accelerating the update process along the axis with small gradients by increasing the gradient and slightly slowing down updates along the axis with large gradients.

  However, there is a drawback to this optimization algorithm. Over time, as training progresses, the sum of squared gradients can become larger. Dividing the current gradient by this larger value leads to very small update steps for the weights. This situation resembles using an extremely low learning rate that diminishes further as training continues. In the worst-case scenario, AdaGrad could cause the training process to become stuck indefinitely.
- RMSprop: [61] RMSprop is a special version of Adagrad developed by Professor Geoffrey Hinton. To tackle with some of the drawbacks of Adagrad, RMSprop tries to store only the gradients within a specified frame instead of summing up all the momentums.

2.2.7.2 Regularization

Regularizers [69] penalize the model and reflects these penalties on the loss function to prevent the model to overfit and achieve generalization. Regularizers play a crucial role in ensuring that a model generalizes well without overfitting. These regularization techniques are applied on a per-layer basis, allowing for better control over the model's complexity and preventing it from memorizing the training data excessively. By applying the regularizers appropriately, the model can strike a balance between fitting the training data and generalizing well to unseen examples, ultimately improving its overall performance.

When a large value is assigned to the regularizer, the optimization process tends to choose a hyperplane with a smaller margin if it leads to better classification of the training points. This encourages a more conservative decision boundary, reducing the risk of overfitting. Conversely, if a lower value is

selected for regularization, the optimizer will prioritize exploring larger distances in search of a hyperplane, even if it results in misclassifications. This can lead to a wider separation between classes in the hyperplane. Regularizers can be applied in different forms, such as:

- Kernel regularizer: This penalty is applied to the layer's kernel, encouraging smoother and more regular patterns in the learned representations.
- Bias regularizer: This penalty is applied to the layer's bias, discouraging overly large biases that could lead to overfitting.
- Activity regularizer: This penalty is applied to the layer's output, promoting sparsity and preventing excessive activation of the neurons.

Below are some of the most popular regularizers:

- The L1 regularization [74] is calculated with the following equation:

$$loss = l1 * reduce_sum(abs(x)) \tag{2.26}$$

  Problems concerning a lot of features may benefit more from L1 regularization since it offers a relatively reduced number of solutions. This also provides an advantage over computational power since the features having zero coefficients do have a critical effect on the model. The regression model that uses the L1 regularization technique is called Lasso Regression.
- The L2 regularization [74] can be formalized as :

$$loss = l2 * reduce_sum(square(x)) \tag{2.27}$$

  This regularizer punishes the model considering the squared values of model parameters. As a result, the model is forced to use the same values with a reduction in size, making it simpler and more general. This regularization is also referred to as Ridge Regression. To compare L1 and L2 regularizers, as we mention earlier L1 tries to decrease the coefficients of the parameters close to zero whereas the L2 regularizer tries to decrease and separate the parameters evenly. As the paper [74] mentions : The L1 regularization technique is beneficial for feature selection as it allows us to identify and discard variables associated with coefficients that become zero. On the other hand, L2 regularization is particularly useful when dealing with collinear or codependent features, as it helps mitigate the effects of multicollinearity.
- Dropout [48]: Dropout is a regularization technique that randomly drops or ignores some output units of a layer during training. This effectively changes the architecture of the layer, as it appears to have a different number of nodes and connections to the preceding layer. By introducing this randomness, dropout prevents the network layers from relying too heavily on each other and encourages the model to learn more robust and generalized representations. It can be applied to individual layers in the network, including convolutional and recurrent layers. Dropout can be used on hidden layers as well as the visible/input layer.
- Batch normalization [65] is another regularization technique that normalizes the activations within a layer. It computes the average of the activations in a batch and divides each activation by the standard deviation of the corresponding batch. This normalization process helps in stabilizing the learning process by reducing internal covariate shifts, allowing for smoother and more efficient training. Batch normalization is commonly used in combination with standardization as a preprocessing step for pixel values in various computer vision tasks.

2.2.7.3 Activation Functions

An activation function can be used to determine the degree of signal strength for a neuron to be activated or not. Through a functional description, an activation computes the output of a neuron as a function of the accumulated strength of its inputs. Hence, activation functions have a huge impact on the neural network's performance. Let's keep in mind that, we would like the model to do more than just learning and providing a linear function that represents the problem. We would like it to perform more complex tasks than modeling. Some of the well-known activation functions which we also included in this research include the following:

- ReLu [63]: In this activation function, only a selection of neurons are activated at a given time. This implies that a neuron will be deactivated or set to zero if and only if the output of the linear transformation is zero.

$$f(x) = max(0, x) \tag{2.28}$$

- Sigmoid [63]: It provides outputs in the range [0,1]. It is mostly used in binary classification problems. The sigmoid function is not symmetric around the origin, it has an S-shape curve, meaning that the signs of all outputs are the same.

$$f(x) = 1/e^{-x} \tag{2.29}$$

- Linear [63]: This activation function uses the received input without a gradient adjustment. It is not very popular since it is not possible to improve the error due to the zero gradient problem.

$$F(x) = ax \tag{2.30}$$

- Tanh [63]: It is similar to the Sigmoid function, however, it is symmetric around the origin, meaning that the signs of outputs may differ. The values change in the range [-1,1]

$$f(x) = 2sigmoid(2x) - 1 \tag{2.31}$$

- Softmax [63]: The softmax function is composed of multiple sigmoid functions. While a sigmoid function produces values between 0 and 1, which can be interpreted as probabilities for binary classification, the softmax function is specifically designed for multiclass classification problems. It computes the probabilities for each data point across all individual classes, providing the probability distribution for the classes. It can be expressed as:

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

Figure 2.9: The formula of softmax function.

2.2.7.4 Learning Rate and Momentum

Learning rate and momentum [40] are two important metrics that impact model performance drastically. The function of the learning rate is to define a lower bound for the step size for the loss calculation when considering the gradient. On the other hand, the function of momentum is to consider adjustments in the weight while the model updates itself. Although these metric values do not follow the standard backpropagation theory, the usage of both variables in collaboration with each other is a common practice for the optimization of the model training process. While the model is under training, for each node in the network an error for the corresponding weight is estimated by the backpropagation and this value is used to scale the weights up or down, via the learning rate, instead of the full amount. In more simple words, a learning rate of 0.1 would correspond to a scaling on the network weights by a factor of 0.1*(estimated weight error) or 10% of the estimated weight error each time the weights are updated. In this sense, the learning rate [14] hyperparameter controls the rate or speed at which the model learns. This is because the learning rate is responsible for the shared error that the model weights have with each update at the batches of the training. When the learning rate is appropriately configured, the model can effectively approximate the desired function using the available resources, such as the number of layers and the number of nodes per layer. If the learning rate selected is larger than optimal, the model will learn quicker, however, the final weights of the model will end up below the optimal selection. Selecting a minor learning rate will let the model achieve almost optimal or may even succeed to reach global optimal weight selection but it will increase the training time and may slow down the learning process. In some cases, the training performance may exhibit oscillations as a result of learning rate selection. If an extremely large lr is selected, the model performance will start to oscillate since the updates on the model will become equally as large. Oscillating performance in a model can be originated from divergent weights, where the weights keep moving away from the optimal values. When the learning rate is too small, the model may struggle to converge or become trapped in sub-optimal solutions.

Momentum [40] on the other hand, is an adjustment on SGD based on a weighted average of previously calculated updates. This forces future updates to proceed in the same direction as the earlier updates. Momentum may make the model training faster for the cases where the weights of the model include structures that may misguide the SGD algorithm due to their high dimensionality. Momentum can slow down the rate of updates, preventing oscillations that we explained earlier and smoothing the whole optimization process. Configuring the learning rate is not directly influenced by momentum. The step size and momentum are independent of each other. However, momentum can enhance the optimization process by working together with the step size to accelerate the search for better weight values. This can lead to the discovery of improved weights in a shorter number of training epochs.

2.3 Evaluation Measures

There are very well-known and trusted methods to evaluate a machine learning model. Since our problem is a classification problem, evaluation metrics related to the classification problems will be detailed in this section.

> Note: This concept is referred to in section 4 when results and discussion are presented.

There are four possible outcomes out of a classification model and almost all metrics are calculated based on these base variables. These outcomes include:

- True positives: True Positives are the outcomes where the model we develop predicts that a given sample belongs to class "A" and the sample, in reality, belongs to class "A".
- True negatives: True Negatives are the outcomes where the model we develop predicts that a given sample does not belong to the class "A" and the sample, in reality, belongs to some other class such as "B" or "C" but does not belong to the class "A".
- False positives: False Positives are the outcomes where the model we develop predicts that a given sample belongs to class "A" and the sample, in reality, does not belong to the class "A" but to some other class "B" or "C".
- False negatives: False Negatives are the outcomes where the model we develop predicts that a given sample does not belong to the class "A" and the sample, in reality, does belong to the class "A".

2.3.1 Confusion Matrix

A confusion matrix is a table consisting of the base metrics we mentioned: True Positives, False Positives, True Negatives, and False Negatives. It is widely used on classification problems to drive statistical information for the model under testing by comparing the predicted and real values. It is especially useful when the classification problem contains two or more classes.

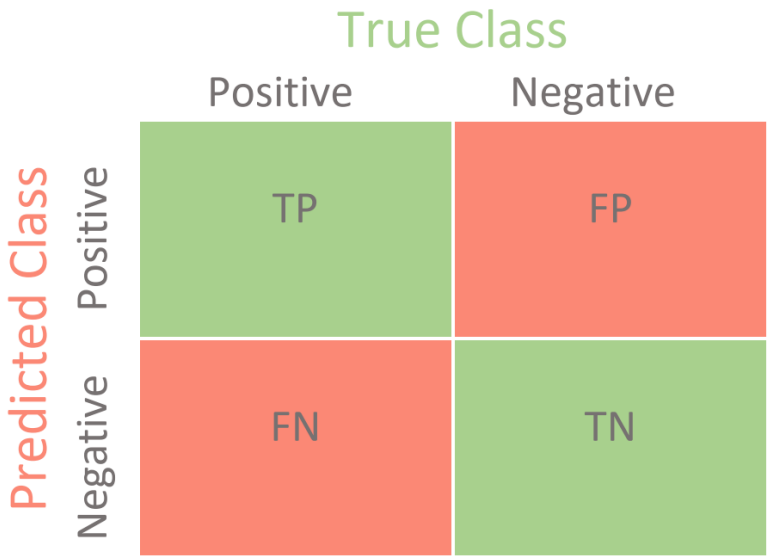


Figure 2.10: Figure showing the confusion matrix. Image from: [49]

2.3.2 Accuracy

An accuracy metric is employed to evaluate the algorithm's performance in a comprehensible manner [67]. Accuracy is expressed as a percentage and indicates the degree to which the model's predictions align with the actual labels. It quantifies the accuracy of the model's predictions relative to the true

data.

$$\frac{(\text{True Positives} + \text{True Negatives})}{(\text{True Positives } + \text{True Negatives} + \text{False Positives} + + \text{False Negatives})} \tag{2.32}$$

2.3.3 Precision

Precision, as defined in [24], assesses the precision or accuracy of a model when predicting positive labels. It measures the model's ability to correctly identify true positives, reflecting the precision of its positive predictions. It tries to identify from the number of positive predictions, how much of it was positive. It gives a percentage of how relevant the predictions of your model are. Prediction is especially a good option to consider calculating when the price of false positives is on the high edge and the damage caused by false negatives is on the low edge. In situations where the cost of false negatives is significantly high, it is recommended to utilize recall as an evaluation metric. Recall measures the model's ability to correctly identify positive instances, emphasizing the model's sensitivity in capturing all relevant positive cases. The formula for precision is below:

$$\frac{\text{True Positives}}{(\text{True Positives } + \text{False Positive})} \tag{2.33}$$

2.3.4 Recall (Sensitivity)

Recall, as defined by [24], quantifies the proportion of true positives correctly identified by a model. It measures the model's effectiveness in capturing actual positive instances. Recall becomes particularly crucial when the consequences of a false negative are significant, making it essential to prioritize the identification of all relevant positive cases. The formula for the recall is below:

$$\frac{\text{True Positives}}{(\text{True Positives } + \text{False Negative})} \tag{2.34}$$

Opposed to the other metrics having a high recall measurement does not always let us conclude that we have a good model. For example, the model we build our case may predict that out of 100 patients all will have neurological disorders within two years. If we were to calculate the recall for this, we would end up with almost a flawless measurement. However in reality we would end up with a lot of false positives which would mean that a lot of people will be identified as unhealthy when they are perfectly healthy.

2.3.5 F1 Score

The F1 score is calculated by feeding the precision and recall through a harmonic mean function. We usually aim to get a high F1 score, which would mean that we will get a high precision and a high recall. If we take a look at the formula of the F1 function, we can see that there is a multiplication of the two metrics. This means that any change in any of those two metrics will significantly affect the F1 score. A model achieves a high F1 score when it accurately identifies positive instances (precision) and avoids misclassifying positive cases as negative (recall). It strikes a balance between these two measures, ensuring that the model doesn't miss positive instances while also minimizing false positives. [54]. The mathematical definition of the F1 score is as follows:

$$F_\beta = \frac{(2 * precision * recall)}{(precision + recall)} \tag{2.35}$$

Another thing to consider here is that the formula gives the same weight to both metrics. In reality, we may need to provide more importance on one of those metrics than the other. This may cause the F1 score to be less effective in calculations. Therefore either a weighted F1 score or seeing the PR or ROC curve can help.

2.3.6 F Score (Weighted F1 Score)

To provide a single evaluation metric for machine learning models, it is common to combine precision and recall into a single score. This combined metric is called the F-score. The F-score captures both precision and recall and provides a comprehensive assessment of the model's performance. By considering

both precision and recall, the F-score offers a balanced measure that reflects the model's effectiveness in handling positive instances while minimizing false positives.[41]. The definition of an F score is given below:

$$F_\beta = (1 + \beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall} \tag{2.36}$$

The β parameter allows us to control the trade-off between precision and recall.

$$\beta < 1 \text{ focuses more on precision} \tag{2.37}$$

while

$$\beta > 1 \text{ focuses more on recall.} \tag{2.38}$$

2.3.7 Loss Functions

A loss function [67] is another building block for the optimization process of the models under training. It is a calculation which shows how badly the model miscalculated the predictions it failed to hit. It is calculated as a summation of error values both for train sets and validation sets. It provides a penalty score for those missed estimates.

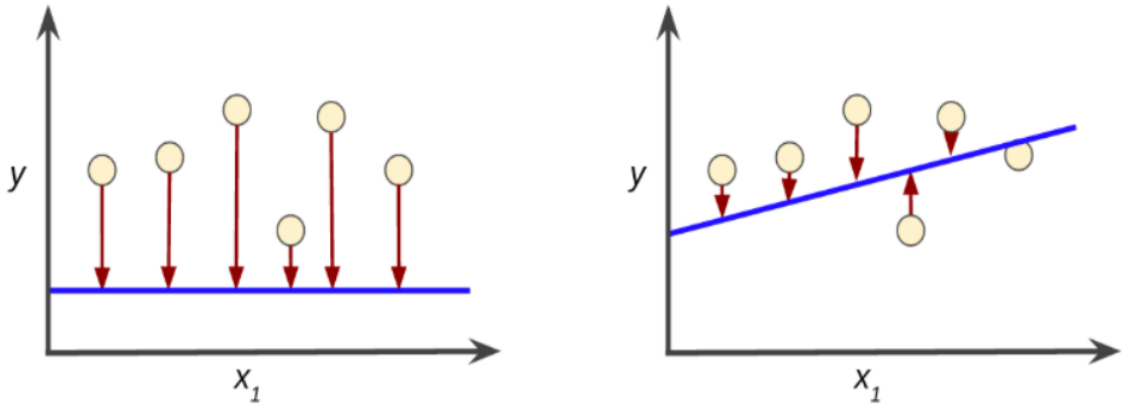


Figure 2.11: Figure showing the loss of predictions made for 2 models. The blue line represents the predictions and the arrows are representing the losses. The first graph is showing a bigger loss since the predictions made have larger errors and the second graph is showing a smaller loss since the predictions made have smaller errors, even when the number of predictions made is fixed and their truth values are the same. Image From: [52]

There are different loss function calculations based on the needs of the developer. For classification problems, cross-entropy is one of the most popular methods.

2.3.7.1 Binary Classification

It is used when there are only two class labels in the sample space. This provides only one output value between -1 and 1 [3].

$$\text{Loss} = -\frac{1}{\substack{\text{output} \\ \text{size}}} \sum_{i=1}^{\substack{\text{output} \\ \text{size}}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

Figure 2.12: The binary cross entropy function.

For binary classification tasks with a single output, it is common to use the sigmoid function as the activation function. The sigmoid function produces a vector of values in the range of (0, 1) and is applied element-wise to each output element. The sigmoid function is also referred to as the logistic function [52].

$$f(s_i) = \frac{1}{1 + e^{-s_i}}$$

Figure 2.13: The formula of the sigmoid function.

2.3.7.2 Multi-Class Single-Label Classification

When dealing with classification problems where each sample can belong to one of the multiple classes, we have two options to calculate the loss: Categorical Cross-Entropy (CCE) and Sparse Categorical Cross-Entropy (SCCE) [3]. The main difference between these two metrics can be explained as follows:

- Categorical Cross-Entropy (CCE) produces a one-hot array that represents the predicted probabilities for each category. This means that the output of the model will be a vector with the same length as the number of classes, and each element of the vector represents the probability of the corresponding class.
- Sparse Categorical Cross-Entropy (SCCE) produces a single category index that represents the class with the highest predicted probability from a list of categories. Instead of returning a one-hot array, it directly provides the index of the most probable class.

Both CCE and SCCE are commonly used loss functions in multiclass classification tasks, and the choice between them depends on the specific requirements of the problem and the desired output format.

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output} \\ \text{size}}} y_i \cdot \log \hat{y}_i$$

Figure 2.14: The categorical cross entropy function.

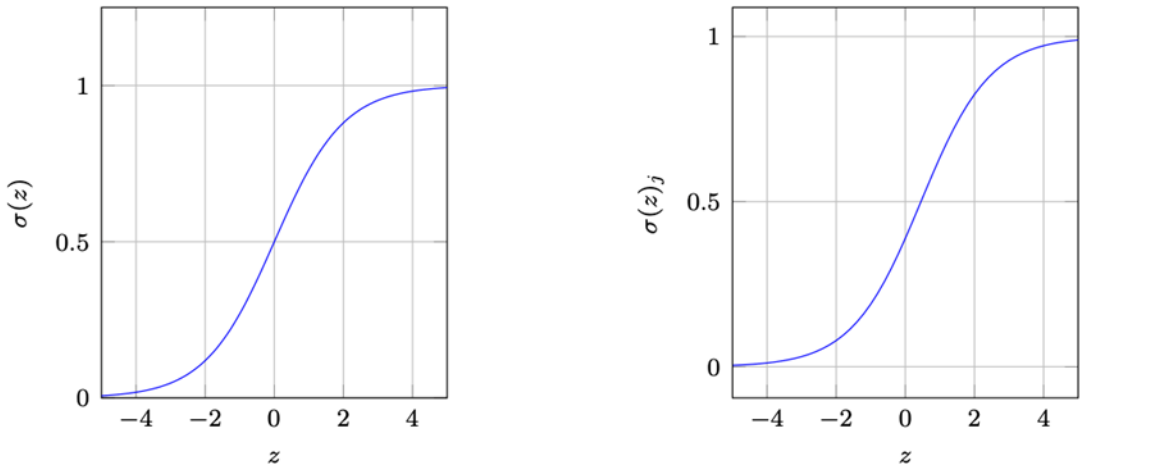
There are cases where you may prefer to use sparse categorical cross entropy function, including:

- when your classes are mutually exclusive, and you do not want to know how close your prediction is to other probable classes.
- the case where the number of categories becomes large and the prediction output becomes overwhelming.

Recalling the definition of the Softmax activation function presented earlier, we can also define a softmax loss. It is a combination of softmax activation function and cross-entropy loss. It produces an outcome between [0,1]. The summation of vector values equals 1. Each of the vector values corresponds to a class probability. Since the softmax function depends on all elements of the vector [52], applying it alone to individual elements is not possible. For a given class, the Softmax function can be computed as:

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

Figure 2.15: The formula of softmax function.



(a) Sigmoid activation function. (b) Softmax activation function.

Figure 1: Sigmoid and Softmax activation functions

Figure 2.16: The difference of the behavior of sigmoid and softmax functions. Image From: [53]

2.3.8 Receiver Operating Characteristic Curve

The Receiver Operating Characteristic Curve (ROC) Curve and the ROC AUC (Area Under Curve) scores are mainly adopted for binary classification problems however, they can be extended for multi-class classification problems. They reveal fruitful information about the evaluated model. They indicate how well our model separates the class by taking into account all possible threshold values. This gives information on the classification abilities of our model [72].

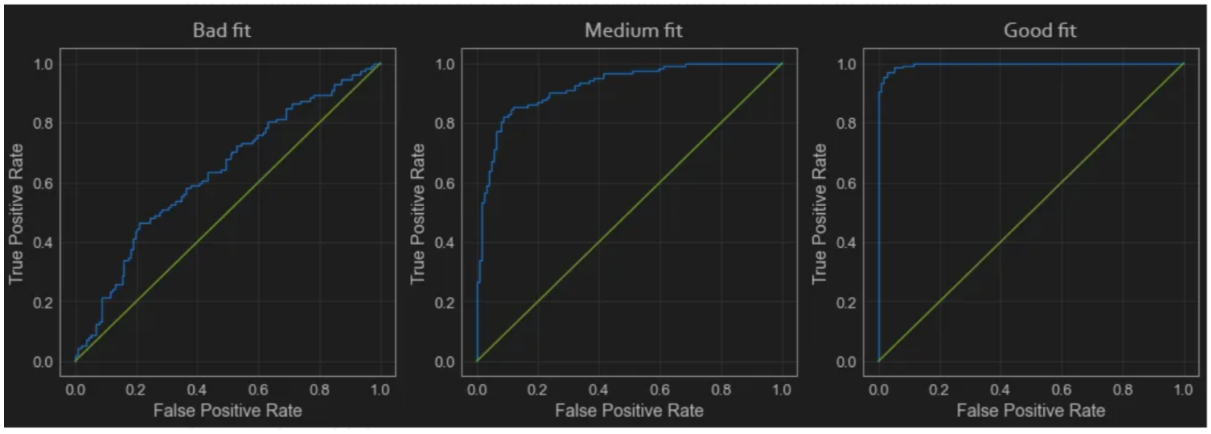


Figure 2.17: Different ROC curves. Image From: [70]

On the plots given above in Figure 3.2, the green line indicates the threshold value where the true positive rate is equal to the false positive rate. The blue line indicates the computed ROC curve of our model. If the green line and the blue line are positioned on top of each other, meaning that our model has the same TPR and FPR, it has a 50% chance to correctly classify a class. It is almost equivalent to flipping a coin. The first figure represents a similar scenario to what we just explained, which is not a

preferred view. The last figure represents an opposite scenario where the blue line is close to the (0,1) coordinate forming an elbow shape. This is an excellent classifier which is not realistic in many cases. The middle figure is somewhere in between the two other classifiers, which is considered to be good and realistic.

Mathematically, we can formulate this comparison of curves by The AUROC Curve (Area Under the ROC Curve). If we were to consider the green line, the area under that curve will give us a score of 0.5. If our model was to achieve 100% accuracy the area under the curve would give us a value of 1. In a realistic scenario, we would try to achieve a value as close to 1.

As we mentioned earlier, the ROC curve is a binary classification metric. To utilize this in our multi class multi-label problem, we have to choose one of the possible solutions [73] given below:

- OvR : *OvR* is an abbreviation for One vs Rest. In this approach, you would have separately compared each class with the remaining classes. To give an example: Let's assume you have three classes; A, B, and C. You would first select Class A and label it as the positive class. All other remaining classes, B and C, would be labeled as the negative class. This way we easily convert our multi-class classification problem to a multiple binary classification problem. This should be repeated for all remaining classes. So for a three-class multi-class problem, you would end up with three different "One vs Rest" scores. You can take finally take the average of all scores to come up with a single score.
- OvO : *OvO* is an abbreviation for One vs One. In this approach, you would compute all possible pair combinations of classes and compute the AUC score for each of them separately. To give the same example: With a three-class multi-classification problem, you would first choose Class A and Class B and discard the other classes. You would label Class A as positive and Class B as negative. You need to pay attention that A vs B is not the same as B vs A since the labeling is different. That is why both cases should be considered when evaluating. This is why, if we were to have a three-class classification problem we would get six class OvO scores:
  - A vs B
  - A vs C
  - B vs C
  - B vs A
  - C vs A
  - C vs B

  As in OvR, we can average all the OvO scores to get a final OvO model score.

Chapter 3

Research

3.1 Pre-Processing of the Datasets

In this section, specific details of preprocessing phases applied within this thesis study will be introduced. There are two inputs received in this phase that will be further investigated in later sections. These inputs are namely:

- Patient Observations Dataset
- MRI Score Dataset

3.1.1 Pre-Processing of Patient Observations

The patient observations included in the dataset are Heart Rate Frequency, Oxygen Saturation, Blood Pressure, Critical Temperature, Five-Minute Apgar Score, Ten-Minute Apgar Score, Gestational Age, Cooled Temperature, and HI Lactated Ring Score. These observations are received as multiple CSV files. First, they are loaded into the development environment and converted to data frames using Python's pandas library functions. Each data frame is later converted to a dictionary using the patient IDs as key attribute and the remaining columns as value attribute. The main goal of this conversion is to provide a better link and to relate individual patients and their observations. This way we converted the table structure to a linked list structure using dictionaries. This also allowed us to view the time series characteristics of individual observations. For ease of calculations, the patient IDs are cast as string data type, and time attribute and Observation attributes are cast as float data types. From each data frame, values equivalent to 0 are removed. The main reason behind this is to overcome challenges that may result from sensor faults, patient or doctor intervention on the sensor value, and any other thing that may introduce a level of bias or cause our system performance to degrade. After removing the 0 values, the data frames are sorted based on the time attribute so that a time series is formed from the moment the patients are first admitted to the hospital to the moment their treatment is considered to be over. Any "NaN" value, meaning that the observation at that point in time is missing, is dropped from individual observations. The measurements are not taken with equally spaced intervals. In an ideal situation, we would expect to see all measurements to be equally distant from each other, like beats per minute. However, in real-life cases, this is not the case which brings the necessity of imputation. The files are structured as time series. This means the information is not a point in time but a curvature in time. Therefore the NaN values are imputed with spline interpolation, which estimates values that minimize overall curvature, thus obtaining a smooth surface passing through the input points. The remaining NaN values, usually the very first and very last rows, are imputed with forward and backward linear interpolation. Duplicated rows are removed since it does not make any sense to have multiple values at a given point in time. Only the last observation taken at that time is added to the evaluation.

The observations are given in different units. For example, heart rate frequency is measured in beats per minute while oxygen saturation is given in percentages. Those differences will have a considerable impact on our model's performance, therefore the data frames are normalized using the Min-Max Scaler.

A bias may be introduced to the model fitting phase because parameters stored in various scales involved in the training provide an unbalanced and unequal contribution.

Min-Max Scaler is a possible solution to this problem since it introduces a feature-based normalization scheme before the training phase. This type of normalization is beneficial when we observe a non-Gaussian distribution or when the data is restricted to upper and lower bounds. To give a simple

example, in images, the pixel value usually has a lower bound of 0 and an upper bound of 255. By applying Min-Max Scaler to these values, we make sure that the portion of which they contribute to overall model fitting is balanced. Min-Max Scaler transforms a data sample x into a scaled sample x' as follows:

$$x^{'} = a + \frac{(x - min(x))(b - a)}{max(x) - min(x)} \tag{3.1}$$

Since we are working with young patients, the severity of the outcome will be affected by gestational age. The younger the patient, the longer they stay in the hospital. Keeping this in mind, we removed patients with gestational age lower than 36 weeks from our dataset to preserve a sample space far from bias. Out of the included features, an optimal feature set that best represents the outcome must be selected. To do that, Multiple meetings were conducted with the experts in UMC Utrecht. Also, a simple CNN model is constructed to be able to see whether or not in practice this selection will optimally reflect our actual results. Our goal with this step was to see how these metrics work with each other and individually. The following table illustrates the accuracies and loss values achieved by a simple CNN (input+ Three Hidden Layer+Flattened Output Layer) on individual time series signals.

Apgar Score 5 min(6) + Apgar Score 10 min(7) :	loss: 0.5460 - accuracy: 0.6977 - val_loss: 0.6331 - val_accuracy: 0.7027
Heart Rate Frequency (5) + Oxygen Saturation(3):	loss: 0.3015 - accuracy: 0.8372 - val_loss: 0.7025 - val_accuracy: 0.6757
Apgar Score 5 min + Heart Rate Frequency:	loss: 0.5895 - accuracy: 0.7093 - val_loss: 0.6315 - val_accuracy: 0.7297
Apgar Score 10 min + Heart Rate Frequency:	loss: 0.5838 - accuracy: 0.7209 - val_loss: 0.5900 - val_accuracy: 0.7297
Apgar Score 5 min + Oxygen Saturation:	loss: 0.6015 - accuracy: 0.6977 - val_loss: 0.5713 - val_accuracy: 0.7027
Apgar Score 10 min + Oxygen Saturation:	loss: 0.5644 - accuracy: 0.6977 - val_loss: 0.5907 - val_accuracy: 0.7027
Apgar Score 5 min + Local Temperature (4)	loss: 0.5711 - accuracy: 0.6977 - val_loss: 0.5722 - val_accuracy: 0.7027
Apgar Score 10 min + Local Temperature:	loss: 0.4997 - accuracy: 0.7674 - val_loss: 0.6995 - val_accuracy: 0.6216
Apgar Score 5 min + Blood Pressure	loss: 0.4344 - accuracy: 0.8023 - val_loss: 0.7043 - val_accuracy: 0.7027
Apgar Score 10 min +Blood Pressure:	loss: 0.4324 - accuracy: 0.8140 - val_loss: 0.5989 - val_accuracy: 0.7297
Apgar Score 5 min + Heart Rate Frequency + Oxygen Saturation:	loss: 0.5772 - accuracy: 0.7093 - val_loss: 0.5613 - val_accuracy: 0.7027
Apgar Score 5 min + Heart Rate Frequency + Local Temperature:	loss: 0.5883 - accuracy: 0.6977 - val_loss: 0.5853 - val_accuracy: 0.7027
Apgar Score 5 min + Heart Rate Frequency + Blood Pressure::	loss: 0.4785 - accuracy: 0.7791 - val_loss: 0.5711 - val_accuracy: 0.7027
Apgar Score 5 min + Oxygen Saturation + Local Temperature:	loss: 0.5824 - accuracy: 0.7442 - val_loss: 0.6079 - val_accuracy: 0.7297
Apgar Score 5 min + Oxygen Saturation + Blood Pressure:	loss: 0.4648 - accuracy: 0.7674 - val_loss: 0.5714 - val_accuracy: 0.7568
Apgar Score 5 min + Blood Pressure + Local Temperature:	loss: 0.4729 - accuracy: 0.8023 - val_loss: 0.5164 - val_accuracy: 0.7568
Apgar Score 10 min + Heart Rate Frequency + Oxygen Saturation:	loss: 0.5654 - accuracy: 0.7093 - val_loss: 0.6256 - val_accuracy: 0.7027
Apgar Score 10 min + Heart Rate Frequency + Local Temperature:	loss: 0.5910 - accuracy: 0.7209 - val_loss: 0.5796 - val_accuracy: 0.7027
Apgar Score 10 min + Heart Rate Frequency + Blood Pressure::	loss: 0.5027 - accuracy: 0.8023 - val_loss: 0.5085 - val_accuracy: 0.8108
Apgar Score 10 min + Oxygen Saturation + Local Temperature:	loss: 0.5704 - accuracy: 0.7093 - val_loss: 0.5573 - val_accuracy: 0.7297
Apgar Score 10 min + Oxygen Saturation + Blood Pressure:	loss: 0.3918 - accuracy: 0.8488 - val_loss: 0.7234 - val_accuracy: 0.6486
Apgar Score 10 min + Blood Pressure + Local Temperature:	loss: 0.4591 - accuracy: 0.7791 - val_loss: 0.6081 - val_accuracy: 0.7297
Heart Rate Frequency + Oxygen Saturation + Local Temperature:	loss: 0.5688 - accuracy: 0.6977 - val_loss: 0.6074 - val_accuracy: 0.7027
Heart Rate Frequency + Oxygen Saturation + Blood Pressure:	loss: 0.4515 - accuracy: 0.7907 - val_loss: 0.5161 - val_accuracy: 0.7568
Oxygen Saturation + Local Temperature + Blood Pressure:	loss: 0.4982 - accuracy: 0.7791 - val_loss: 0.5033 - val_accuracy: 0.7568
Apgar Score 5 min + Heart Rate Frequency + Oxygen Saturation + Blood Pressure:	loss: 0.4824 - accuracy: 0.7791 - val_loss: 0.5477 - val_accuracy: 0.7838
Apgar Score 5 min + Heart Rate Frequency + Oxygen Saturation + Local Temperature:	loss: 0.5896 - accuracy: 0.7209 - val_loss: 0.5643 - val_accuracy: 0.7027
Apgar Score 5 min + Heart Rate Frequency + Oxygen Saturation + Apgar Score 10 min :	loss: 0.5812 - accuracy: 0.7093 - val_loss: 0.5438 - val_accuracy: 0.7027
Apgar Score 5 min + Oxygen Saturation + Blood Pressure + Local Temperature:	loss: 0.5081 - accuracy: 0.7442 - val_loss: 0.5311 - val_accuracy: 0.7297
Apgar Score 5 min + Oxygen Saturation + Blood Pressure + Apgar Score 10 min:	loss: 0.4767 - accuracy: 0.7791 - val_loss: 0.5371 - val_accuracy: 0.7027
Apgar Score 5 min + Blood Pressure + Local Temperature + Heart Rate Frequency:	loss: 0.5172 - accuracy: 0.7907 - val_loss: 0.4993 - val_accuracy: 0.7838
Apgar Score 5 min + Blood Pressure + Local Temperature + Apgar Score 10 min:	loss: 0.4514 - accuracy: 0.7907 - val_loss: 0.7492 - val_accuracy: 0.7027
Apgar Score 5 min + Local Temperature + Heart Rate Frequency + Blood Pressure:	loss: 0.4940 - accuracy: 0.7558 - val_loss: 0.5737 - val_accuracy: 0.7838
Apgar Score 5 min + Local Temperature + Heart Rate Frequency + Apgar Score 10 min:	loss: 0.4741 - accuracy: 0.7674 - val_loss: 0.6056 - val_accuracy: 0.7297
Blood Pressure + Local Temperature + Heart Rate Frequency + Oxygen Saturation:	loss: 0.5555 - accuracy: 0.7093 - val_loss: 0.7113 - val_accuracy: 0.6486
Blood Pressure + Local Temperature + Heart Rate Frequency +Apgar Score 10 min:	loss: 0.4634 - accuracy: 0.8023 - val_loss: 0.5645 - val_accuracy: 0.7568
Blood Pressure + Heart Rate Frequency + Oxygen Saturation + Apgar Score 10 min:	loss: 0.3902 - accuracy: 0.7791 - val_loss: 0.9879 - val_accuracy: 0.7297
Heart Rate Frequency + Oxygen Saturation + Local Temperature + Apgar Score 10 min:	loss: 0.5298 - accuracy: 0.7326 - val_loss: 0.5542 - val_accuracy: 0.7027
Heart Rate Frequency + Blood Pressure + Apgar Score 5 min + Apgar Score 10 min:	loss: 0.4823 - accuracy: 0.8140 - val_loss: 0.6808 - val_accuracy: 0.6757

Table 3.1: The accuracies and loss values of training and testing samples for the basic CNN model with three convolutional layers and the effect of selecting different feature sets.

From the results illustrated above, we decided on using five metrics: Heart Rate Saturation, Blood Pressure, Local Temperature, Apgar Score and later we also added Respiration Frequency based on experimentation. We also need to mention that the values shown here are the "RAW" values. These values are the ones used before the employed CNN Model is placed inside the experimental framework we prepared. The details of this framework are provided in the coming chapters. After these initial preprocessing steps, preprocessing phases specific phases are followed. Each of these phases are introduced in this section and the output of this phase is used as an input to the remaining phases.

3.1.1.1 Preprocessing Based on Generative Methods

In this research, we have followed two different approaches. Each of those approaches requires specific computational needs.

Each of the data structures we mentioned in Section 3.1.1 is divided into features. This means we would have five different feature sets: Heart Rate Frequency, Oxygen Saturation, Respiration Frequency, Blood Pressure, and Temperature. Each feature represents an observation. Each feature is then further divided into three classes. These classes are extracted from the MRI Labels, which are explained in Section 3.1.2. The possible classes these samples may end up in include Class 0, meaning the lowest severity level, Class 1, meaning the moderate severity level, and Class 2, meaning the highest severity level. Each Class is later divided into a subset of days (first day, first two days, first three days, first four days, first five days, and finally first six days).

3.1.1.2 Preprocessing Based on Discriminative Methods

For the second approach, We have applied the following additional processing steps. Each of the data structures we mentioned in section 3.1.1 is divided into features. Each feature is then divided into three classes as described above. Each feature is divided into different patient arrays (containing 1440 observations, observations taken every two hours for six days) considering the class information. Each patient array is also segmented into 120 (two days) consecutive overlapping blocks. ([0,120],[1,121],[2,122]...). From each block, a unique array is generated and exported in .npz format.

3.1.1.3 Our Goal With Additional PreProcessing

The idea behind these divisions is :

- To be able to further augment the data.
- To be able to reduce future space complexity.
- To be able to preserve the critical information between the transitions.
- To be able to see which days are the most critical days for the prediction.
- To be able to build a committee of models to vote for the final decision.

3.1.2 Pre-Processing of Magnetic Resonance Imaging (MRI) Scores

The MRI Dataset consists of grey matter and white matter-related measurements like cerebellum score, stroke, BGT result, DAG result, etc. The dataset is delivered as a separate CSV file and loaded to our development environment using the pandas library of Python. The loaded dataset is converted to a data frame and the observations are cast as float data type for the ease and accuracy of our calculations. The "NA" values are removed which may cause any bias on our model. The MRI Scores are designed to be used for our labeling. Each row in the given structure falls under a category. Therefore the paper [38] suggested by the clinicians is used to calculate and divide the measured MRI scores into three categories.

Table I. MRI scoring form

	Items	Sequence used to assess injury	Degree		
Grey matter			0	1	2
1	Thalamus abnormal SI or diffusion restriction	T1/T2 DWI	No	Focal (<50%)	Extensive (≥50%)
	Specify location			Unilateral	Bilateral
2	Basal ganglia abnormal SI or diffusion restriction	T1/T2 DWI	No	Focal (<50%)	Extensive (≥50%)
	Specify location			Unilateral	Bilateral
3	PLIC myelination or diffusion restriction	T1/T2 DWI	Normal or no diffusion restriction	Equivocal/partially myelinated or partial (<50%) diffusion restriction	Absent myelination or extensive (≥50%) diffusion restriction
	Specify location			Unilateral	Bilateral
4	Brainstem (peduncles) abnormal SI or diffusion restriction	T1/T2 DWI	No	Focal (<50%)	Extensive (≥50%)
	Specify location			Unilateral	Bilateral
5	Perirolandic cortex diffusion restriction	DWI	No	Mild	Clear
	Specify location			Unilateral	Bilateral
6	Hippocampus diffusion restriction	DWI	No	Yes	
	Specify location			Unilateral	Bilateral
Grey matter subscore					
	Basal ganglia NAA	^{1}H-MRS	Normal	Reduced	
	Basal ganglia lactate	^{1}H-MRS	Absent	Increased	
Grey matter subscore (including ^{1}H-MRS)					
	White matter/cortex		0	1	2
1	Cortex abnormal SI or diffusion restriction not being perirolandic cortex	T1/T2 DWI	No	Focal (1 lobe)	Extensive (>1 lobe)
	Specify location			Unilateral	Bilateral
2	White matter increased SI or diffusion restriction not being PWML	T1/T2 DWI	No	Focal (1 lobe)	Extensive (>1 lobe)
	Specify location			Unilateral	Bilateral
3	PWML	T1/T2, DWI, SWI	No	<6	≥6
	Specify location			Unilateral	Bilateral
4	Hemorrhage not being PWML	T1/T2, SWI	No	Single hemorrhage <1.5 cm	≥1.5 cm or multiple hemorrhages
	Specify location			Unilateral	Bilateral
5	Optic radiation diffusion restriction	DWI	No	Mild	Clear
	Specify location			Unilateral	Bilateral
6	Corpus callosum diffusion restriction	DWI	No	Yes	
White matter subscore					
	Cerebellum		0	1	2
1	Cerebellum abnormal SI or diffusion restriction	T1/T2 DWI	No	Focal (<0.5 cm)	Extensive (≥0.5 cm or multiple lesions)
	Specify location			Unilateral	Bilateral
2	Cerebellar hemorrhage	T1/T2, SWI	No	Single hemorrhage <0.5 cm	≥0.5 cm or multiple hemorrhages
	Specify location			Unilateral	Bilateral
Cerebellum subscore					
	Additional		0	1	2
1	IVH	T1/T2, SWI	No	Yes	
2	SDH	T1/T2	No	Yes	
3	CSVT	T1/T2, MRV	No	Yes	
Additional subscore					
Total score (grey matter + white matter + cerebellum + additional score)					

Figure 3.1: Figure showing the MRI Scoring From from the paper [38]. As you can see from the above figure, we can classify the patients into three different severity levels based on the provided thresholds. Each of the rows provides a different measurement related to the brain region affected by the disease. Based on the threshold values, the generated recurrence plots are labeled as either class 0 (lowest severity level), class 1(moderate severity level), and class 2 (highest severity level). To give a simple example, if we were to take the PLIC Myelination score and if the patient showed a normal or no diffusion it would be labeled as 0, partial myelination as 1, and absent/extensive myelination as 2. Depending on the final condition of the patient on the day the MRI score is measured, the recurrence plots are generated labeled reflecting the final day outcome on earlier days.

The paper [38] shows that there is a great relation between the outcome and grey matter subscore. The degree column in the MRI Scoring Form reveals information on how the patients are divided into severity levels.

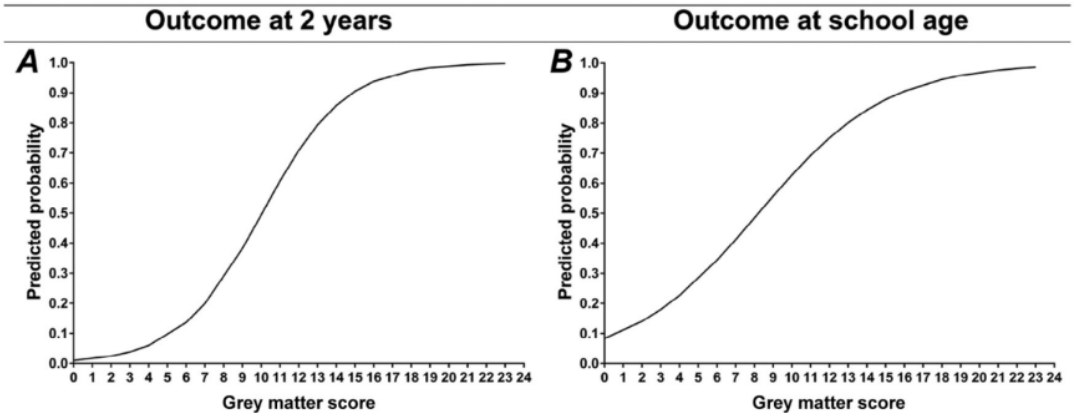


Figure 3.2: Figure showing the predicted probability of death or impairment about grey matter score from the paper [38]. This proves that the given MRI scores can be used as labels from a medical point of view.

3.2 Reasons Why a Regression Model Would Fail to Solve This Problem

Regression for time series data can be useful for modeling and predicting future values based on historical patterns. However, it is important to note that traditional regression techniques assume that the data points are independent and identically distributed (i.i.d.), which may not hold for time series data.

Time series data often exhibit temporal dependencies, such as trends, seasonality, and autocorrelation, which violate the assumptions of traditional regression models. Therefore, specialized time series analysis techniques are typically employed to capture these dependencies and make accurate predictions. While regression techniques for time series data are designed to handle dependencies and patterns, they may not explicitly assume that the data follows a fixed distribution. The distributional assumptions can vary depending on the specific modeling technique used (e.g., Gaussian distribution in ARIMA), but the primary focus is on capturing the temporal dependencies rather than assuming a fixed distribution for the data. To practically show the failure of the issue, We have contracted a regression model as the following:

We have constructed a regression model with four dense layers with 512,256,128 and 64 weights respectively. We added an initial dropout of 0.5 and batch normalization in between layers to ensure the model does not overfit. We used the softmax function as the activation function and categorical cross-entropy as the loss function. We also used the GridSearch hyperparameter optimization technique to ensure the most optimal parameters are selected for the model training. We have inputted generated enhanced recurrence plots to the regression model for each observation separately. On average the results achieved were suboptimal due to the above-explained issue:

loss: 0.6895 - accuracy: 0.7590 - val_loss: 0.7256 - val_accuracy: 0.7489

This proves that we need a more advanced structure like convolutional layers for this problem. Convolutional layers in deep learning models are specifically designed to learn spatial and consecutive features from input data. Convolutional neural networks (CNNs) leverage the convolution operation to extract relevant features from images, time series data, and other forms of structured data. In the context of images, convolutional layers are capable of capturing spatial features by applying filters (kernels) across the input image. These filters slide over the image, performing element-wise multiplications and summations to generate feature maps. Each filter learns to detect specific patterns or features such as edges, corners, or textures, which are spatially encoded in the resulting feature maps. For time series or sequential data, convolutional layers can learn consecutive features by applying 1D convolutions. In this case, the filters slide along the temporal dimension of the input, capturing patterns and dependencies across neighboring data points. This allows the network to recognize local patterns and extract relevant information from the sequential input. By stacking multiple convolutional layers, CNNs can learn hierarchical representations of features, starting from low-level features (e.g., edges) and gradually progressing to higher-level features (e.g., shapes or objects). The subsequent layers in a CNN can then further process these learned

features to make predictions or perform specific tasks. The ability of convolutional layers to learn spatial and consecutive features makes them particularly effective for tasks such as image classification, object detection, semantic segmentation, and time series analysis. These layers can automatically learn and extract meaningful features from the data, reducing the need for manual feature engineering and enhancing the model's performance.

3.3 Research Based on Generative Methods

3.3.1 Generative Adversarial Network

We have trained a generator and a discriminator for each class separately (three generators and three discriminators in total for Mild, Severe, and Healthy classes). Each model is an expert in his/her class. Therefore the test data will be given to all three models and in return the models will tell us if that test data belongs to their class or not. The generator will generate nearly real fake data. The discriminator will get our training set and the data generated by the generator and try to differentiate between real data and fake data. Over time, the generator will become good at generating fake data and the discriminator will be good at discriminating data. Overall we have a training dataset collected from 123 patients. We were hoping that using a GAN will allow us to augment this number, by taking into account the class changes that may occur over time in a time series data.

In the below code, you can view how the discriminators and generators are created and interact with each other.

```
def train(dataset, epochs, dim)
def train_step(real_asph, dim):
```

We have two main functions acting in the training phase. The train function is associated with the flow of training, where the inputs and outputs of discriminators and generators are fed into each other. The train step function on the other hand controls what happens in each step of training procedures. The details of each method are like the following:

The function train_step(real_asph, dim) receives real asphyxiated samples and a dim variable concerning the dimensions.

A noise function is computed taking the dimension of the original sample into account. In our case, we selected to go with a normal distribution.

```
noise = tf.random.normal(dim)
```

The generator model is trained with the noise vector. Initially, the generator will make a lot of errors, because the noise function is not yet updated and the element values are not representing the real samples.

```
genResult = generator(noise, training=True)
```

The discriminator model is trained with real asphyxiated samples.

```
discResultReal = discriminator(real_asph, training=True)
```

To test the ability of the discriminator to discriminate the real and fraud samples, we also train it with the generated samples.

```
discResultFake= discriminator(genResult, training=True)

gradientsGen = gen_tape.gradient(genLoss, generator.trainable_variables)

generator_optimizer.apply_gradients(zip(gradientsGen, generator.trainable_variables))
```

Later we use the optimizers to update and optimize the gradient values for the next step of the training phase. To Summarize: Once our models are generated, the generator is initialized with random noise. The discriminator is trained with the first batch of real samples. Later, the generator input is combined with the rest of the batches and provided to the discriminator. Losses and accuracies are calculated. Both model states are updated and the whole process is repeated until they converge.

3.3.2 Discriminator Characteristics

The Discriminator [83] is a binary classifier that takes inputs x or G(z) and outputs discrimination results (Real or Fake). Its purpose is to accurately discriminate between x and G(z). The Discriminator is trained with the Back propagation algorithm (BP algorithm) to calculate the objective function and update the network weights.

Upon receiving input x, the Discriminator aims to classify it as a "Genuine" instance (labeled as 1). Conversely, when presented with G(z) as input, the Discriminator is anticipated to classify it as a "Fraud" instance (labeled as 0). However, the Generator aims to "cheat" the Discriminator by generating outputs that can fool it. This creates a competitive and adversarial relationship between the Generator and Discriminator.

The Discriminator's output is a value D that indicates how close the generated image is to a real image. The goal is for the Discriminator to effectively discriminate between authentic images and fraudulently generated images by the Generator. The cross-entropy function is commonly used to measure the loss $p.log(q)$ in this process. This function evaluates the performance of a classification model, working well for this task because the loss increases as the predicted probability deviates further from the true label.

Below, you can view the code for the Discriminator. Since we used a CNN GAN, the network characteristics are similar to a CNN architecture we have explained in detail in section 2

```
model = tf.keras.Sequential()
```

We created a Sequential Model. The Sequential model allows you to create a neural network by simply adding layers one by one. Each layer in the model performs a specific operation on the input data and passes the output to the next layer in the sequence.

```
model.add(layers.Input(shape=(5,1919)))
model.add(layers.Permute((2, 1)))
model.add(layers.Conv1D(filters=32, kernel_size=16, strides=1, padding='same'))
```

We received 5 observations with size 1919 for the first day measurements. The received observations contained Heart Rate frequency, Oxygen Saturation, Blood Pressure, Respiration Frequency, and Temperature as we described in the preprocessing phase of the generative approach. We provided the *permute* function to shuffle the training samples. During the training phase, shuffling the order of training samples within each epoch can help avoid any potential bias that may arise from the original ordering of the data. It ensures that the model sees a diverse range of samples during each training iteration, preventing it from becoming biased toward specific patterns in the data. Later we provided a one-dimensional convolutional neural network, which we tested separately for each observation. In total, we added 4 convolutional layers with filter sizes 128,64,32 and 16 respectively.

```
model.add(layers.LeakyReLU())
model.add(layers.Dropout(0.4))
model.add(layers.MaxPool1D(pool_size=2))
```

We introduced the above structures to improve the generalizability of the model, as we described in more detail in section 2.

```
model.add(layers.Flatten())
model.add(layers.Dense(1))
```

Finally, we flattened the model so that we reduced the model into a binary classification problem. This way the model expert in one aspect of our system will tell us whether or not, the given sample belongs to that model's domain or not.

3.3.3 Generator Characteristics

Within the conventional GAN framework [83], the Generator and Discriminator are structured as Multilayer Perceptrons (MLPs). The primary objective of the Generator is to acquire knowledge about the distribution of real samples, labeled as x, and subsequently produce synthesized samples, denoted as $G(z)$, that closely emulate the distribution of real samples.

To achieve this, the Generator takes a basic random noise input, typically represented as z, which adheres to a simplistic distribution, such as the Gaussian distribution. By utilizing random noise as input, the Generator aims to introduce diversity and encompass the wide range of variations present in the real sample distribution. The Generator's output, $G(z)$, is a generated sample possessing the same dimensions as the real samples, x. Through an adversarial learning process, wherein the Generator and Discriminator compete, the Generator gradually learns to generate samples that exhibit a remarkable resemblance to the real samples, effectively capturing the intricate nature of the underlying distribution.

In essence, the Generator component, operating within the GAN model, leverages an MLP architecture to generate synthetic samples, $G(z)$, leveraging a random noise input, z, to approximate the distribution characteristics observed in the real samples, x.

Below, you can view the code for the Generator. The details of the CNN and LSTM architectures can be found in details in section 2

```
model = tf.keras.Sequential()
```

The generator model is sequential just like the discriminator model.

```
model.add(layers.Input(shape=(88,9595,)))
model.add(layers.Bidirectional(layers.LSTM(64, return_sequences=True)))
```

It will accept the remaining samples and build an LSTM model on top of those samples. A bidirectional LSTM is a type of recurrent neural network architecture that processes input sequences in both forward and backward directions. It combines the information from past and future time steps, enabling the model to capture dependencies in both directions.

```
model.add(layers.Conv1D(filters=128, kernel_size=16, strides=1, padding='same'))
```

Similar to the discriminator we added 4 convolutional layers, with filters 128,64,32 and 16 respectively.

```
model.add(layers.LeakyReLU())
model.add(layers.Permute((2, 1)))
```

We also added activation functions and permutations for hyperparameter optimization purposes.

3.3.4 Problems With Generative Adversarial Network and Directives Which Lead us to Apply the Second Approach

In contrast to alternative generative models [83], GAN exhibits several distinctive advantages. Firstly, GAN's computational processing solely relies on the BP algorithm to calculate gradients. As a result, GAN exhibits faster computational speed and does not necessitate approximate reasoning during the learning process. Secondly, GANs lack a variational lower bound and can generate clear, unbiased images. Thirdly, GAN can generate data samples in parallel, significantly reducing the time required for sample generation. Lastly, GAN's Generators have no limitations on input data size and can effectively train various generative networks within a flexible framework.

On the other hand, Generative Adversarial Networks come with certain drawbacks. The first disadvantage lies in achieving convergence, which is particularly challenging outside the scope of convex functions. Although stochastic gradient descent (SGD) can be utilized to efficiently convex cases, simultaneous training of the Discriminator and Generator remains a daunting task. The second drawback is mode collapse, where the Generator's learning capability declines and the generated samples lack diversity when it becomes proficient at generating real-like samples under specific parameters. The third challenge is gradient disappearance, occurring when the Discriminator consistently distinguishes real samples from generated ones, leading to zero gradients and preventing the Generator from further learning the real sample distribution. The fourth limitation is the uncontrollable nature of GAN. In the standard GAN model, the Generator relies solely on random noise as input, and the model cannot be constrained by specific conditions for generating samples with desired characteristics. In our case, we experienced the following:

- Even though the discriminator works around 97% accuracy, the Granularity of differences in the data makes it hard for the generator to achieve optimal accuracy. After some time the generator does not recognize and generate the required differences because the differences are not very significant in magnitude.

- To verify our experimental conclusion, we tried to train GAN with sinusoidal signals, and I have observed that as the length signal gets larger, generator accuracy approaches to 0. That is generator becomes unable to represent small differences in signal amplitudes.
- The length of features makes it computationally complex when the values taken into account increase.

3.4 Research Based on Discriminative Methods

In this section, we will be explaining the final successful approach we took to solve the problem. This section tries to identify a solution that resolves the problems faced in earlier trials.

3.4.1 Recurrence Plots

The *.npy* files obtained in the preprocessing phase, are one by one fed into our recurrence plot generation function. The specific method details are explained in section 2. For each of the images generated with this logic, we also applied an additional elimination step where we eliminate images with a standard deviation below 0.01. This further processing was required to eliminate stable intervals where the standard deviation falls below 0.01. This indicates that there are no significant increments or decrements that may benefit our model. Our main goal is to successfully identify the sudden increments and decrements in the patient observations. That is why we are trying to identify the existence of these changes, not when they occur. Eliminating these stable intervals will not have an adverse effect on our evaluation. The images are also reshaped to 32x32. This value is selected based on experimentation based on two criteria: To reduce computational complexity and to preserve critical information.

Some of the generated plots are presented below to give the readers a better understanding of the outcome. In the below images, the highlighted portions are where the sudden increments are decrements that occur at a granular level:

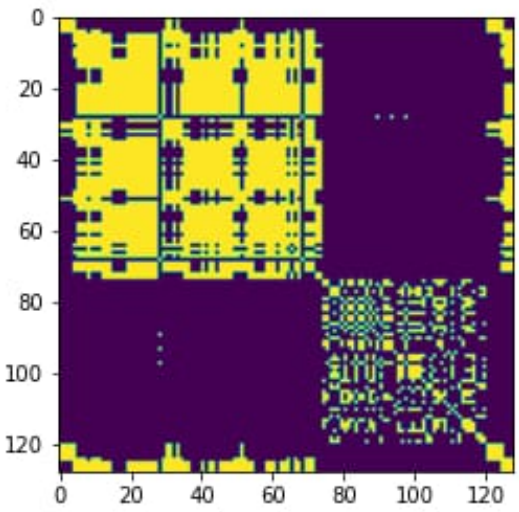


Figure 3.3: Figure showing the generated recurrence plot for patient 0 class 0 Heart Rate Block 1.

As mentioned earlier, the time series information is divided into consecutive overlapping periods of 2 hours. Each of these blocks are then processed using the mathematical information we provided in section 2. The generated recurrence plots give us information on where the descriptive information lies. The color changes in the recurrence plots give us information about the sudden increments and decrements. The changes in variance both in location over the image and color depth give how significant the change is and how frequently it occurred. If we were to look at the plot, we can see that for class 0, the changes are accumulated mainly in the right bottom and left top corners and the degree of change is more observable on the upper left corner.

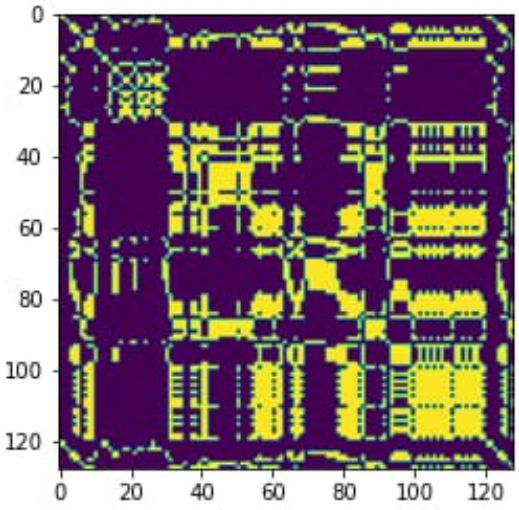


Figure 3.4: Figure showing the generated recurrence plot for patient 0 class 1 Heart Rate Block 6.

If we were to look at the plot, we can see that for class 1, the changes are accumulated differently than class 0. This is not possible to observe with bare eyes, because these differences are very small in magnitude in their raw form and won't make sense to the human eye or the trained model. That's why recurrence plots are needed to enhance those differences.

3.4.2 Advanced Network Architecture Building: AlexNet

AlexNet is a large network structure with 60 million parameters and 650,000 neurons. The network begins with a convolutional layer followed by a rectified linear unit (ReLU) activation function, which introduces non-linearity. Max pooling layers are then used to downsample the feature maps. This is followed by more convolutional layers, ReLU activations, and max pooling. The final layers are fully connected layers that map the learned features to the desired number of output classes. AlexNet utilizes dropout and local response normalization techniques to prevent overfitting. The table from the paper [44] provides a visual explanation of the generic architecture of Alexnet as the following :

1. Input: The input to the network is a 32x32 RGB image.
2. Conv1: The first layer is a convolutional layer with 96 filters of size 5x5x3, using a stride of 1 and padding of 0. ReLU activation is applied.
3. Pool1: Max pooling is applied with a filter size of 2x2, a stride of 2, and padding of 0.
4. Conv2: The second layer is a convolutional layer with 256 filters of size 5x5x96, using a stride of 1 and padding of 2. ReLU activation is applied.
5. Pool2: Max pooling is applied with a filter size of 2x2, a stride of 2, and padding of 0.
6. Conv3: The third layer is a convolutional layer with 384 filters of size 3x3x256, using a stride of 1 and padding of 1. ReLU activation is applied.
7. Conv4: The fourth layer is a convolutional layer with 384 filters of size 3x3x384, using a stride of 1 and padding of 1. ReLU activation is applied.
8. Conv5: The fifth layer is a convolutional layer with 256 filters of size 3x3x384, using a stride of 1 and padding of 1. ReLU activation is applied.
9. Pool5: Max pooling is applied with a filter size of 2x2, a stride of 2, and padding of 0.
10. FC6: A fully connected layer with 4096 neurons is added. ReLU activation is applied.
11. FC7: Another fully connected layer with 4096 neurons is added. ReLU activation is applied.
12. FC8: The final fully connected layer consists of 1000 neurons, corresponding to the 1000 classes in the dataset.
13. Softmax: A softmax activation function is applied to obtain class probabilities.
14. Output: The output layer represents the classification output, using cross-entropy loss with the

class information.

We used this described logic and constructed an AlexNet architecture separately for each patient observation. An example code for "Heart Rate" Observation is given below:

In our model, we started by constructing a sequential base by using the Sequential() function from keras. We used two-dimensional convolutional neural networks with an activation function "relu". Our images had the input shape (32,32,3). They were in 32x32 size and had 3 dimensions each since they are colored RGB images. In total, we introduced 3 convolutional blocks. Within each block, we employed batch normalization and max-pooling with pool size (2,2) to enhance generalizability. The convolutional block filters were 128, 64, and 128 respectively. In the last dense layers, we introduced dropouts and used the *tanh* activation function, which is different than the explained table. We modified the generic architecture so that it fits our specific problem. Later we used the softmax function to provide the final predictions. Since we are dealing with a multi-class classification problem, we used a categorical cross-entropy function for the loss calculation. We also used Adam optimizer with learning 1e6. We selected these by experimenting with the dataset.

3.4.3 K-Fold Cross-Validation

Ensuring the reliability and robustness of validation procedures is a crucial step in model development, particularly when it comes to predicting known outcomes and classifying response variables [47]. Validation, distinct from calibration which assesses how well a model fits a specific dataset, involves testing the model against an independent dataset that was not used during the initial construction and parameterization of the model. Various validation methods exist, including bootstrapping, jackknifing, and cross-validation, each serving to assess the model's performance in novel situations. Cross-validation, in particular, offers insights into the model's accuracy and classification success when confronted with new or unfamiliar scenarios. It also helps identify overfitting, a situation where the model performs well on the calibration data but poorly on new data.

A widely employed form of model validation is k-fold cross-validation. In this approach, the dataset, consisting of n cases with covariate and response variable values, is divided into k equal segments. The model is trained and tested iteratively on different combinations of these segments, measuring classification error rates against known outcomes. The value of k can vary from 2 to n-1, with k=2 representing a simple splitting of the dataset in half, and k=n-1 known as the "leave one out" (LOO) approach, where the model is trained on n-1 cases and evaluated on each case. However, LOO can be computationally expensive and may not provide additional validation benefits compared to lower values of k, potentially leading to high variance and overfitting.

Validation tests for each k subset involve calculating classification accuracy, bias, and variance in error rates. As k increases from 2 to n-1, bias decreases while variance in error rates increases, accompanied by longer computation times. Additionally, bias and model classification error tend to be inversely related. It's important to note that when k=1, the validation results pertain to model calibration rather than validation since there are no subsets of the dataset.

In the case of our training using the AlexNet architecture, we utilized k-fold cross-validation with a value of k=15, determined through experimentation. After completing the folds, we selected the best-performing model with high accuracy and low loss, representing our problem domain effectively.

3.5 Model Committee Building and Majority Voting

After constructing individual models for each observation separately, We saved those models and designed the full flow of the application. The idea is to create a script that will accept an individual CSV file containing patient observations for a single patient that is taken in between one to six days. This gives the doctors the flexibility to view the patient's condition at any point in time before the six days.

The received *.csv* file first passes through the explained preprocessing phases, the same phases which we applied before we trained our model so that the image blocks for each observation are created and the differences in the consecutive image blocks are enhanced. These image blocks for different patient observations are fed into 5 different models (for each patient observation separately):

Model Loading Phase:

```
heartmodel = keras.models.load_model("HeartRate/HeartRateModel")
bloodmodel = keras.models.load_model(BloodPress/BloodPressureModel")
oxygenmodel = keras.models.load_model("OxygenSat/OxygenSaturationModel")
```

```
respfreqmodel = keras.models.load_model("RespFreq/RespirationFrequencyModel")
tempmodel = keras.models.load_model(Temp/TemperatureModel")
```

Example Model Prediction For Respiration Frequency:

```
if observation_name=="RespFreq":
        image_folder=f"C:/Users/acans/Desktop/RecurrencePlotsCleaned/{observation_name}"

        # Get a list of all image files in the folder
        image_files = [f for f in os.listdir(image_folder) if os.path.isfile(os.path.join(image_

        # Load each image using Pillow
        images = []
        for image_file in image_files:
            image_path = os.path.join(image_folder, image_file)
            image = Image.open(image_path)
            image_array = np.array(image)
            image_tensor = tf.convert_to_tensor(image_array)
            images.append(image_tensor)

        images=np.array(images)
        # Reshape the whole image data
        images = images.reshape(len(images),3,32,32)
        #print("Shape after reshape and before transpose:", images.shape)
        # Transpose the whole data
        images = images.transpose(0,2,3,1)

        res=respfreqmodel.predict(images)

        print("Labels For Respiration Frequency Corresponding to Images:")
        print(np.argmax(res,axis=1))
        lst=list(np.argmax(res,axis=1))
```

After the predictions are made for all five models, the committee is gathered to evaluate the final results. The final predictions are presented in terms of a curve. This also allows the clinicians to view at which point in time, the patient started to move from a lower severity class to a higher severity class. This will provide them a better chance to identify what they referred to as the "surprise" cases, where the patient looks healthy but unexpectedly shows really bad MRI scores. Below you can see an example of how the prediction looks for oxygen saturation, respiration frequency, and blood pressure observations:

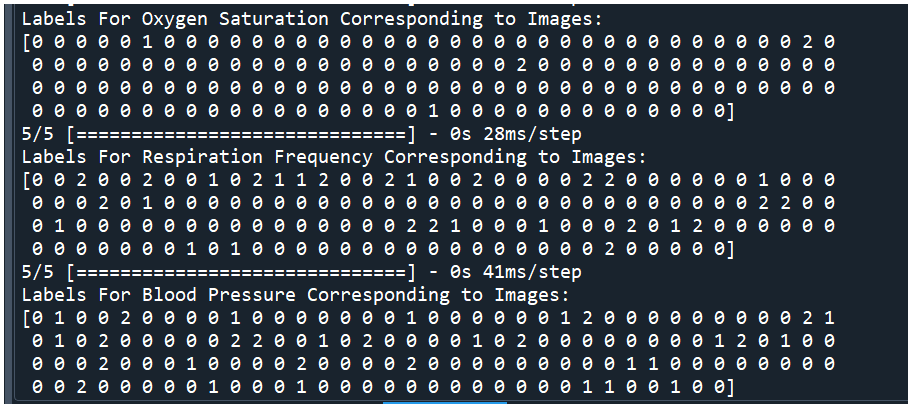


Figure 3.5: Figure showing the predicted class values for each observation separately. You can also view at which interval the patient started to change class information. This also explains why the "surprise" cases are so unexpected.

Later, the expert models will show what they provide as an output, and a final curve showing the patient's final situation is provided:

Below you can see an example of how the prediction looks for combined observations:

```
[2, 2, 2, 0, 2, 2, 0, 0, 1, 1, 2, 1, 1, 2, 0, 0, 2, 1, 0, 0, 2, 0, 0, 0, 1, 2,
2, 0, 0, 0, 0, 0, 0, 2, 1, 2, 1, 0, 1, 0, 2, 0, 1, 0, 0, 0, 2, 2, 0, 0, 1, 0, 2,
0, 2, 0, 0, 1, 2, 2, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 0, 0, 0, 2, 0, 2, 0, 0,
0, 1, 0, 0, 0, 0, 2, 0, 1, 0, 0, 2, 2, 1, 0, 0, 2, 1, 1, 0, 0, 2, 1, 1, 2, 2, 0,
0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 0, 1, 2, 0, 0, 1, 0, 0]
```

Figure 3.6: Figure showing the predicted class values for combined patient observations using weighted average method taking into account the model accuracy and severity levels for a given time interval of all observations.

3.6 Final Prognosis

Time final prognosis for the patient under consideration, is provided via a time-weighted averaging operation. The list representation could be too technical for the doctors and a single prognosis would be a better option. Therefore, taking into account the accuracy results of the models, we divided the weights of each model accordingly to the list representation. The final weighted list which is formed from the final lists provided by each model separately, contains a list of classes assigned to each time slot (every two hours for the number of days the doctor decides to take into account), we have provided a time-weighted average, which gives higher weight to the most recent observation.

```
FINAL RESULT :
WEIGHTED AVERAGE BASED ON ACCURACY SCORES:
[1.0, 2.0, 1.0, 0.0, 1.0, 2.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0,
0.0, 1.0, 1.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 1.0, 2.0, 1.0, 0.0, 0.0, 0.0,
0.0, 0.0, 0.0, 2.0, 1.0, 2.0, 1.0, 0.0, 1.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0,
0.0, 2.0, 2.0, 0.0, 0.0, 1.0, 0.0, 2.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 3.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 2.0, 2.0, 2.0, 0.0, 0.0, 0.0,
2.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0, 0.0,
0.0, 2.0, 1.0, 1.0, 0.0, 0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 2.0, 1.0, 1.0, 1.0,
1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0,
1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 1.0, 2.0, 0.0, 0.0, 1.0, 0.0, 0.0]
Time weighted prognosis:
1.0
```

Figure 3.7: The final prognosis of a patient.

To make everything look less technical and visually more understandable for the doctors, we convert the list of information to a graph like the following:

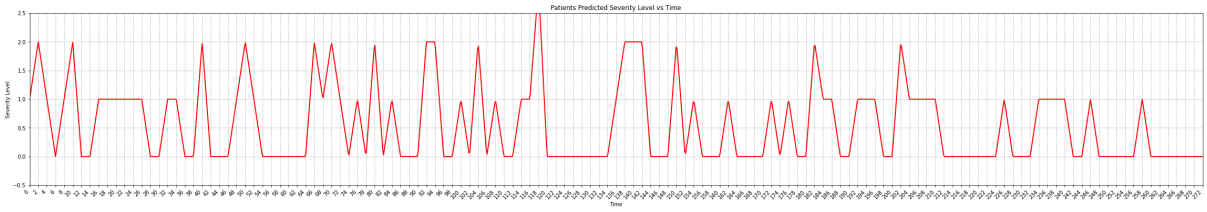


Figure 3.8: The final prognosis of a patient.

Chapter 4

Results

In this chapter, we present the results of our experiment(s) based on evaluation metrics. To better understand what the numbers represent in these results, we will be providing a patient inclusion flowchart:

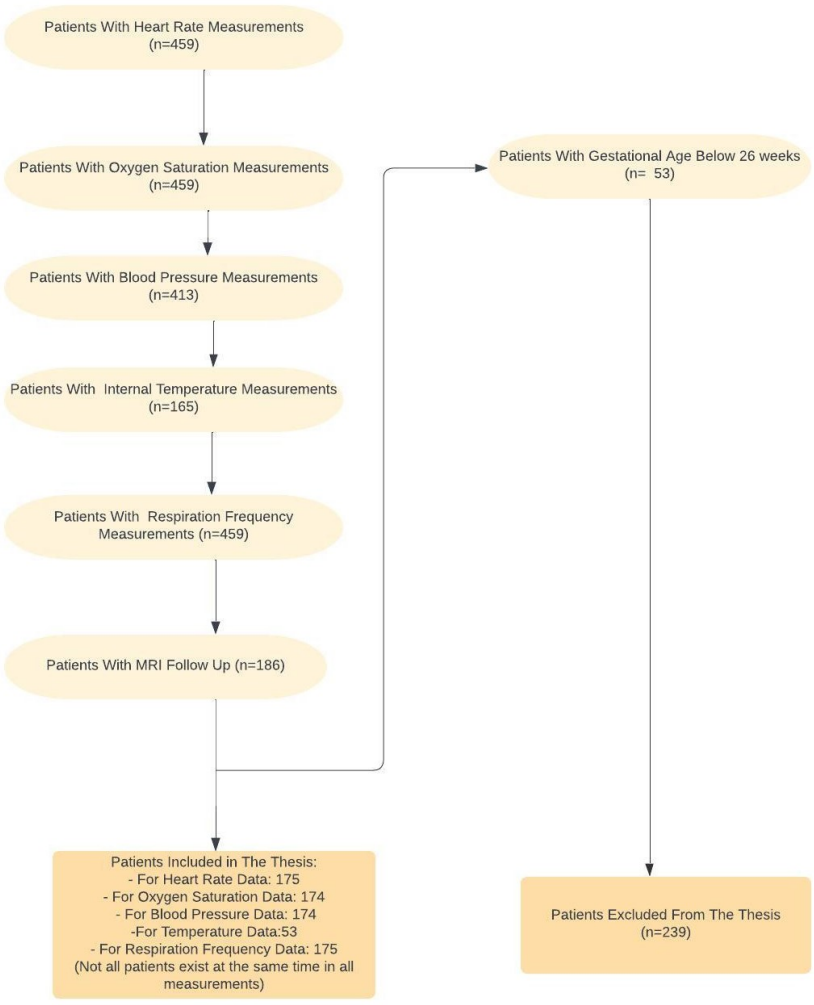


Figure 4.1: The patient inclusion flowchart displays the number of included patients in this study.

4.1 Training Results

In this section, I will be presenting a detailed analysis of the model performance on the training data. I used 90% of the original dataset for training purposes. After the successful completion of the training phase, I tested my model using multiple metrics and plots with the remaining dataset. I have provided both the before and after standard deviation calculation and cleaning of recurrence plots accordingly, with accurate results. This is because I would like to emphasize the importance of this step in the model performance. Overall in both of the training issues, I achieved above-average performance when compared with other related approaches. I also need to mention that there is no previous study that tries to tackle this exact problem. By the term related approaches, I try to refer to the approaches which try to find a solution to a similar problem. The most affected observation by this operation is oxygen saturation and I think this is because doctors continuously provide supplements to the patients to preserve vital signals which have a considerable impact on the model.

Table 4.1: Accuracy Results for AlexNet

Metric	Before Standard Deviation	After Standard Deviation
Heart Rate	85%	97%
Oxygen Saturation	40%	90%
Blood Pressure	85%	96%
Temperature	84%	90%
Respiration Frequency	81%	93%

4.2 Testing Results

In this section, we will be presenting a detailed analysis of the model performance on the testing data. We reserved 10% of the original dataset for testing purposes. After the successful completion of the training phase, We have tested my model using multiple metrics and plots, which you can find more information on the calculation information in Section 2:

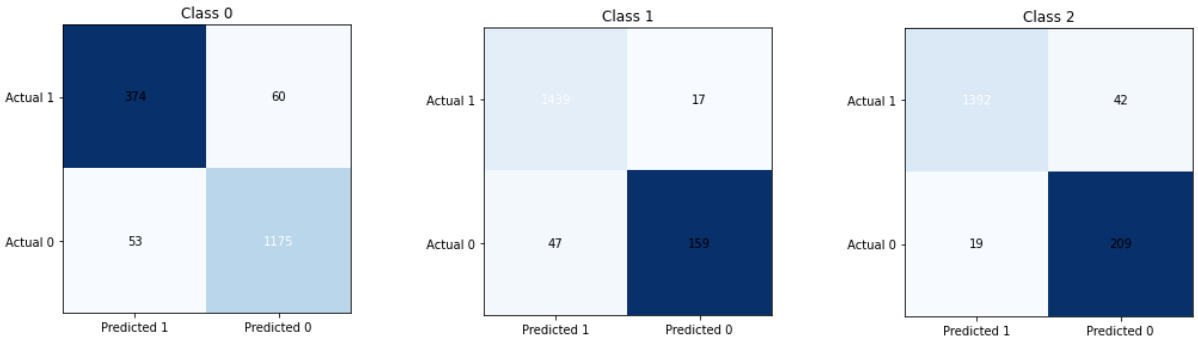
4.2.1 Blood Pressure Results

Below are the calculated testing metric results for the Blood Pressure model :

Table 4.2: Testing Metrics for the Blood Pressure Model

Metric	Value
Accuracy	0.9283995186522263
Categorical cross-entropy loss	0.3194803767184765
Precision	0.93
Recall	0.93
F1 Score	0.93

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure 4.2: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Blood Pressure Observation. The numbers you see here represent the number of matching blocks generated following the logic I have explained for recurrence plots in Section 3.

ROC Curve:

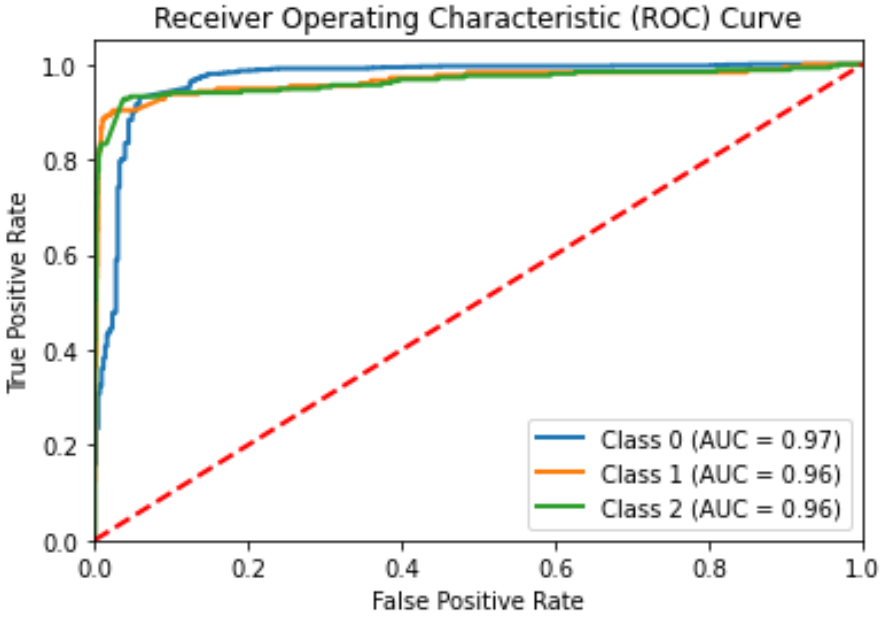


Figure 4.3

The area under the curve score below represents the probability that a random positive sample is positioned to the right of a random negative sample. We chose to use the 1 vs rest approach to get a more in-depth idea of how each class separately performed vs the others. As you can see the AUC scores range between 0 and 1, and the higher the score the better the result is. A more detailed explanation can be found in section 2 . The above results are pretty close to 1, meaning that we achieved almost ideal results for Blood Pressure measurements.

4.2.1.1 Evaluation of the Results

The results related to the Blood Pressure model were very satisfactory. Because other related approaches achieved an average of 85% accuracy, the model showed an amazing performance. We also need to mention that there is no previous study that tries to tackle this exact problem. By the term related approaches, We try to refer to the approaches which try to find a solution to a similar problem. The loss value is also at a good level. This means that our model does not make huge mistakes, in other words, the misclassified patients are somewhere near to their true classes. To give an example, if the patient's true class is 2 and if the model were to misclassify this patient, it would probably select 1 instead of 0,

which will make the margin of error a lot smaller. Precision, Recall, and F1 score values are all above 90% which is also higher than the average 87.5% of the analyzed previous approaches. The confusion matrix gives us a very clear view of the statistical measurements of the true positives, true negatives, false positives, and false negatives. These values indicate that there are significantly fewer false predictions than true predictions. If we were to analyze the values in the confusion matrix: For Class 0, out of 1662 samples only 60 false positives and 53 false negatives exist. For Class 1, out of 1662 samples, 17 false positives and 47 false negatives exist. For Class 2, out of 1662 samples, only 42 false positives and 19 false negatives exist. When we compare the ROC Curve of our model with the mentioned optimal ROC Curve in Section 2, you will be able to see that our model achieved an almost optimal area under the curve result.

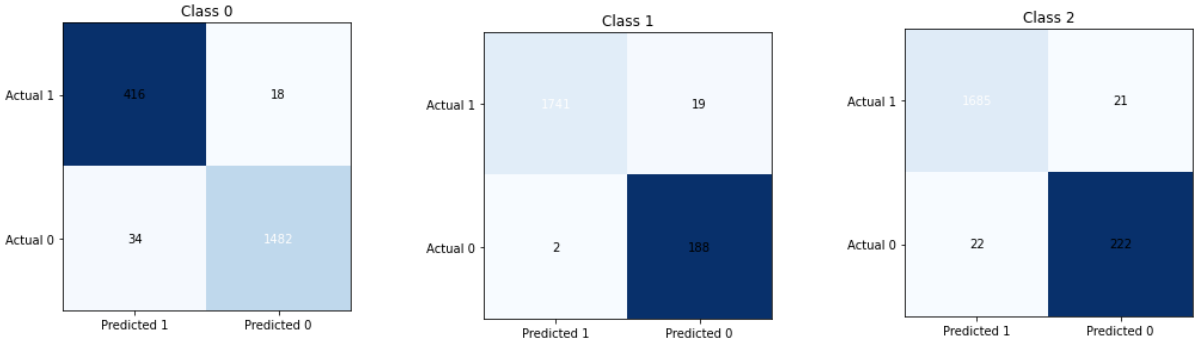
4.2.2 Heart Rate

Below are the calculated testing metric results for the Heart Rate model :

Table 4.3: Testing Metrics For Heart Rate

Metric	Value
Accuracy	0.97
Categorical cross-entropy loss	0.19664617908228593
Precision	0.94
Recall	0.96
F1 Score	0.95

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure 4.4: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Heart Rate Observation. The numbers you see here represent the number of matching blocks generated following the logic I have explained for recurrence plots in Section 3.

ROC Curve:

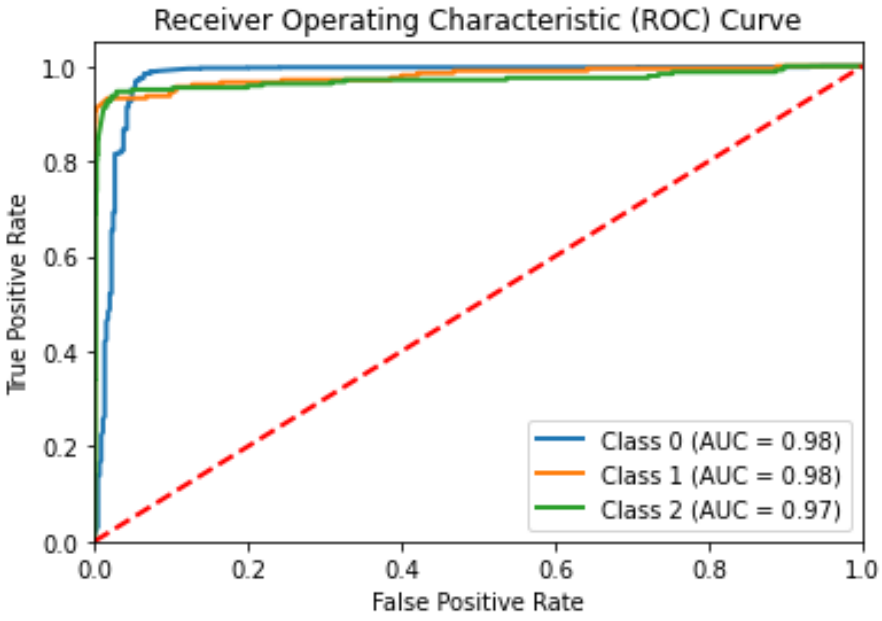


Figure 4.5

A more detailed explanation of how to evaluate the AUC scores can be found in section 2 . The above results are pretty close to 1 similar to the Blood Pressure scores, meaning that we achieved almost ideal results for Heart Rate measurements as well. These indicate that Heart Rate and Blood Pressure Measurements alone are good identifiers of the disease outcome.

4.2.2.1 Evaluation of the Results

The results related to the Heart Rate model were even better than the blood pressure model. Considering the fact that other related approaches achieved an average of 85% accuracy, the model showed an outstanding performance. The loss value is at a very satisfactory level, especially when compared with the blood pressure model. We can conclude that the margin of error for the heart rate model is considerably small. Precision, Recall, and F1 score values are all almost above 95% which is also higher than the average 87.5% of the analyzed previous approaches. The confusion matrix values are very nice. If we were to analyze the values in the confusion matrix: For Class 0, out of 1950 samples only 18 false positives and 34 false negatives exist. For Class 1, out of 1950 samples, 19 false positives and 2 false negatives exist. For Class 2, out of 1950 samples, only 21 false positives and 22 false negatives exist. When we compare the ROC Curve of our model with the mentioned optimal ROC Curve in Section 2, you will be able to see that our model achieved an almost optimal area under the curve result.

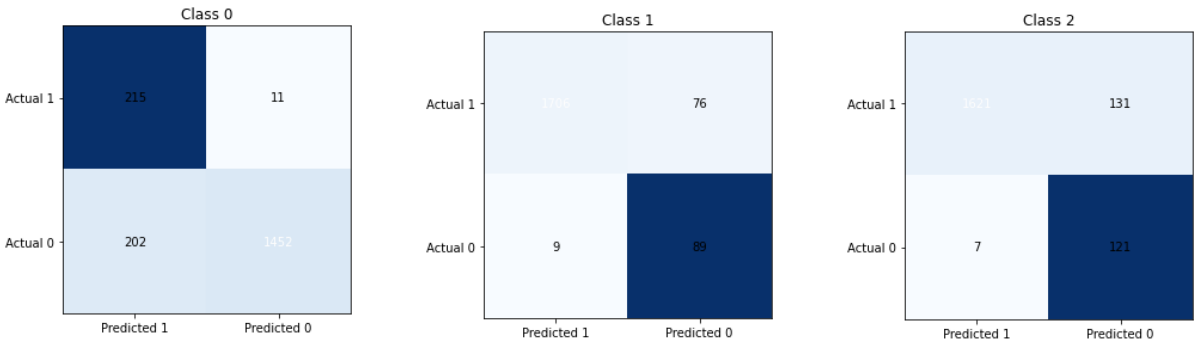
4.2.3 Oxygen Saturation

Below are the calculated testing metric results for the Oxygen Saturation model :

Table 4.4: Testing Metrics for Oxygen Saturation

Metric	Value
Precision	0.90
Categorical cross-entropy loss	0.39308049931580896
Precision	0.93
Recall	0.88
F1 Score	0.90

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure 4.6: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Oxygen Saturation Observation. The numbers you see here represent the number of matching blocks generated following the logic I have explained for recurrence plots in Section 3.

ROC Curve:

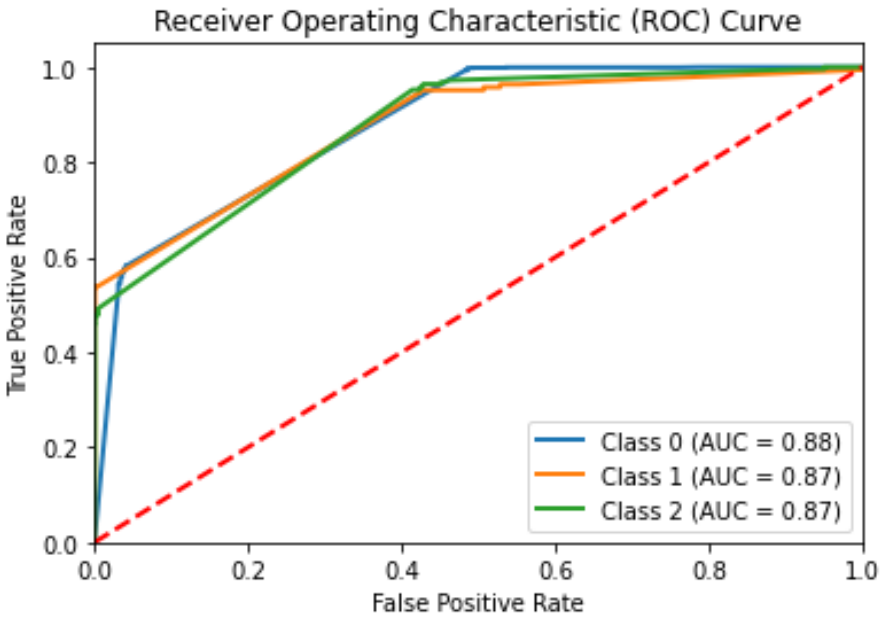


Figure 4.7

The above results are slightly lower than what we have achieved with the Heart Rate Scores and Blood Pressure scores. The main reason for this difference can be found in the below evaluation of the results section. These indicate that the Oxygen Saturation identifier should be supported with other identifiers to provide a more accurate outcome of the disease.

4.2.3.1 Evaluation of the Results

The results related to the Oxygen Saturation model were at an average level as the previous approaches. The other models achieved a better performance than the oxygen saturation level. The main reason for this difference is that oxygen saturation is the first thing affected by this disease. Hence, the doctors usually give oxygen treatments to the patients as a supplement, which may introduce some sort of bias to the measurements and may affect the model performance. Considering the fact that other related approaches achieved an average of 85% accuracy, the model still showed a nice performance. The loss value is considerably increased, especially when compared with the blood pressure and heart rate model. This is also due to the explained issue. Precision, Recall, and F1 score values are all above 90% which

is also higher than the average 87.5% of the analyzed previous approaches. The confusion matrix values are very nice. If we were to analyze the values in the confusion matrix: For Class 0, out of 1880 samples only 11 false positives and 202 false negatives exist. For Class 1, out of 1880 samples, 26 false positives and 9 false negatives exist. For Class 2, out of 1880 samples, only 131 false positives and 7 false negatives exist. As you can see the error number also increased. When we compare the ROC Curve of our model with the mentioned optimal ROC Curve in Section 2, you will be able to see that our model degraded a little bit however it still is close to that "elbow" structure. In Conclusion, although this model achieved a lower performance than the other models, I still believe that it performs better than the other approaches and hence should be included in the committee.

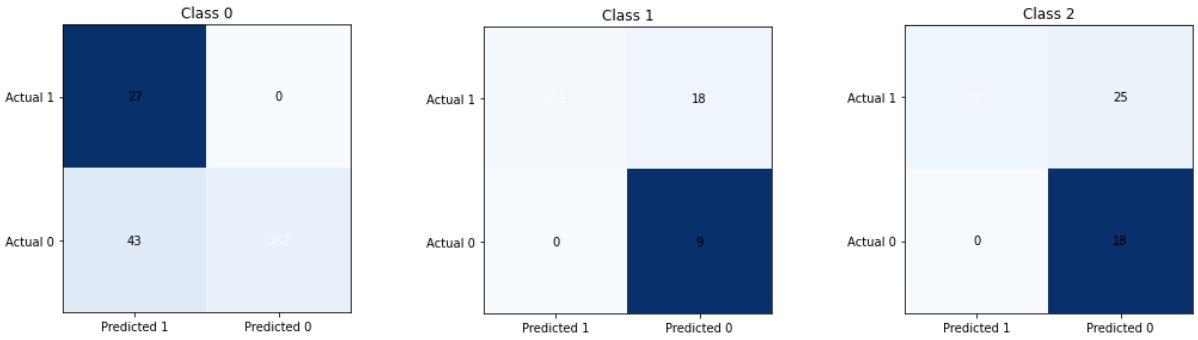
4.2.4 Temperature

Below are the calculated testing metric results for the Temperature model :

Table 4.5: Testing Metrics

Metric	Value
Precision	0.90
Categorical cross-entropy loss	0.5009483694195901
Precision	0.96
Recall	0.90
F1 Score	0.92

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure 4.8: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Temperature Observation. The numbers you see here represent the number of matching blocks generated following the logic I have explained for recurrence plots in Section 3.

ROC Curve:

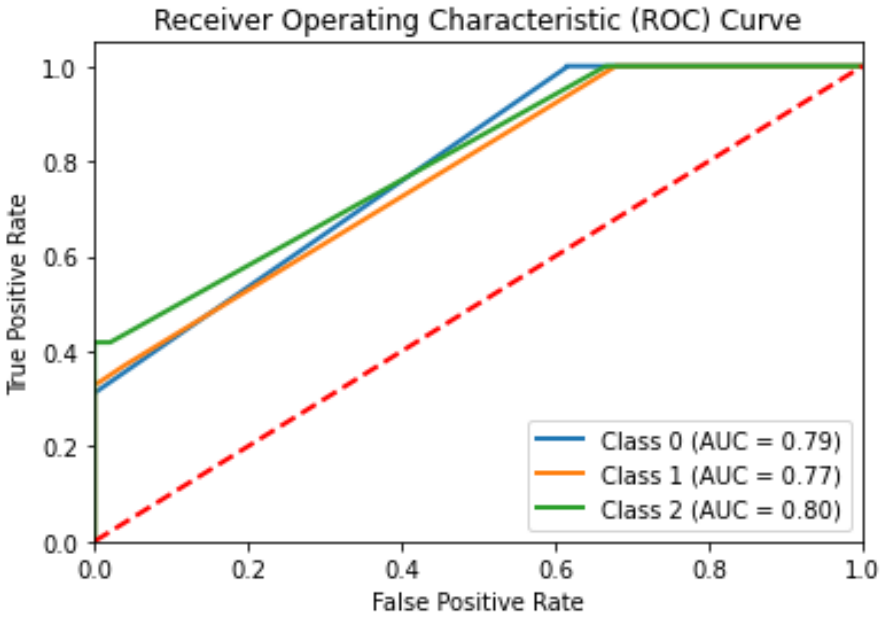


Figure 4.9

The above results are very similar to the Oxygen Saturation scores, because of the same reasons. The main reason for this difference can be found in the below evaluation of the results section. These indicate that Oxygen Saturation and Temperature, although providing approximately 80% accuracy and can be considered good identifiers, these identifiers should be supported with other identifiers to provide a more accurate outcome of the disease.

4.2.4.1 Evaluation of the Results

The results related to the Temperature model show a similar scenario to the Oxygen Saturation model. The other models achieved a better performance than these models. The main reason for this difference is that just like oxygen saturation, temperature is affected by this disease at a very early stage. Hence, the doctors usually try to lower the patient's temperature via external interventions, which may introduce some sort of bias to the measurements and may affect the model performance. Considering the fact that other related approaches achieved an average of 85% accuracy, the model still showed a nice performance. The loss value is considerably increased, especially when compared with the blood pressure and heart rate model. This is also due to the explained issue. Precision, Recall, and F1 score values are all above 90% which is also higher than the average 87.5% of the analyzed previous approaches. The confusion matrix values are very nice. If we were to analyze the values in the confusion matrix: For Class 0, out of 432 samples 0 false positives and 43 false negatives exist. For Class 1, out of 432 samples, 18 false positives and 0 false negatives exist. For Class 2, out of 432 samples, only 25 false positives and 0 false negatives exist. These values are interesting because as you can see, for some classifications the model provided 0 false positives, meaning that it never misclassified these groups. When we compare the ROC Curve of our model with the mentioned optimal ROC Curve in Section 2, you will be able to see that our model degraded a little bit however it still is close to that "elbow" structure. In Conclusion, although this model achieved a lower performance than the other models, I still believe that it performs better than the other approaches and hence should be included in the committee.

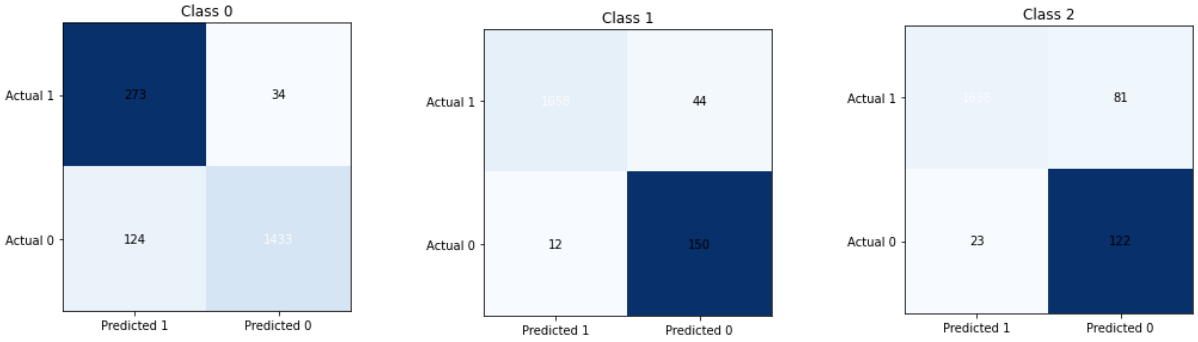
4.2.5 Respiration Frequency

Below are the calculated testing metric results for the Respiration Frequency model :

Confusion Matrix:

Table 4.6: Testing Metrics

Metric	Value
Accuracy	0.93
Categorical cross-entropy loss	0.3122747810775202
Precision	0.93
Recall	0.91
F1 Score	0.92



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure 4.10: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Respiration Frequency Observation. The numbers you see here represent the number of matching blocks generated following the logic I have explained for recurrence plots in Section 3.

ROC Curve:

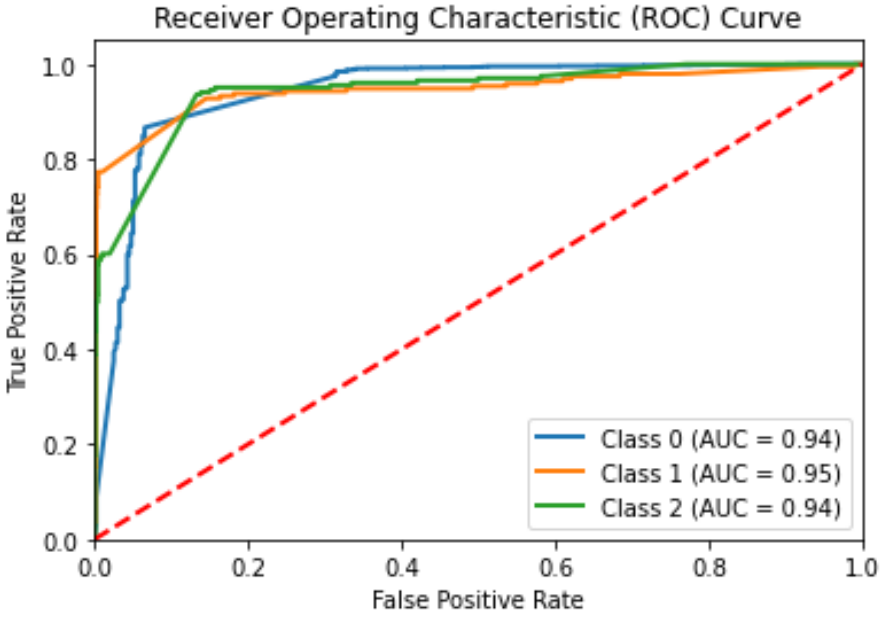


Figure 4.11

The above results are very similar to what we have achieved with the Heart Rate Scores and Blood Pressure scores. These indicate that the Respiration Frequency identifier can also be used alone to provide accurate results.

4.2.5.1 Evaluation of the Results

The results related to the Respiration Frequency model were outstanding. Considering the fact that other related approaches achieved an average of 85% accuracy, the model showed an amazing performance. This model is very similar to the heart model when the metrics are compared. The loss value is also at a good level. This means that our model does not make huge mistakes, in other words, the misclassified patients are somewhere near to their true classes. Precision, Recall, and F1 score values are all above 90% which is also higher than the average 87.5% of the analyzed previous approaches. The confusion matrix gives us a very clear view of the statistical measurements of the true positives, true negatives, false positives, and false negatives. These values indicate that there are significantly fewer false predictions than true predictions. If we were to analyze the values in the confusion matrix: For Class 0, out of 1834 samples only 34 false positives and 124 false negatives exist. For Class 1, out of 1834 samples, 46 false positives and 12 false negatives exist. For Class 2, out of 1834 samples, only 81 false positives and 23 false negatives exist. When we compare the ROC Curve of our model with the mentioned optimal ROC Curve in Section 2, you will be able to see that our model achieved an almost optimal area under the curve result.

We analyzed the progression of the metric values day by day starting from day 1 observations up to day 6 observations since our main goal is to identify whether or not it is possible to give a prognosis before the MRI Scan. My results indicate that the accuracy, precision, recall, f-1, ROC, and loss metrics show high performance on all days. This indicates that the results are trustable even in the earlier days of the treatment. I also observed that the metric performances increased as we introduce more data as the days pass. You can also check the progression of the metric values for each section in the Appendix 7.1

Chapter 5

Discussion

In this chapter, we will be highlighting the important outcomes of our analysis from the results section.

> **Finding 1:** The patient's observations can be enhanced using recurrence plots, which can not be done solely using generative approaches.

The findings from Finding 1 revealed the following to us:
Recurrence plots, as described in section 2, are valuable tools in analyzing and visualizing complex time series data. They provide insights into the dynamic behavior and patterns within the data by highlighting recurrent states and the temporal relationships between them. By constructing recurrence plots, we can identify and analyze the presence of recurrent patterns, periodicities, or irregularities in the observed data. Generative approaches, on the other hand, are methods that aim to model and simulate data based on learned patterns or statistical properties. While generative models can be useful in generating synthetic data that resembles the observed data, they may not capture the intricate and nonlinear dynamics present in the original time series. This limitation makes generative approaches less effective in capturing and enhancing the specific patterns and recurrence characteristics that recurrence plots can reveal. This issue led us to change our approach. By using recurrence plots, we can directly visualize the recurrence patterns within the time series data, enabling them to identify important features and characteristics that may not be captured by generative models alone. Recurrence plots provide a comprehensive and intuitive representation of the temporal dependencies, self-similarity, and complex behaviors present in the observed data.

> **Finding 2:** An advanced technique like AlexNet can be employed to perform accurate predictions on patient outcomes.

The findings from Finding 2 revealed the following to us:
AlexNet is a deep convolutional neural network (CNN) architecture that has been widely used in image classification tasks. By leveraging its deep architecture and a large number of learnable parameters, it can effectively extract meaningful features from medical images or other relevant data and make accurate predictions. This can be particularly valuable in healthcare scenarios where precise predictions are crucial for determining patient outcomes like ours. Accurate predictions of patient outcomes can have significant clinical implications. By leveraging advanced techniques like AlexNet, healthcare providers can make more informed decisions, personalize treatments, identify high-risk patients, and potentially improve overall patient care and outcomes.

> **Finding 3:** The outcomes do not only reveal severity information but also show at which time points those class changes occur.

The findings from Finding 3 revealed the following to us:
One of the unique aspects of healthcare data is its temporal nature. Patient data, such as vital signs, measurements, or symptoms, is often collected over time. With our AlexNet model, it is possible to analyze how the predicted classes change over different time points. This temporal analysis allows for a deeper understanding of the dynamics and progression of the condition being studied. This informa-

tion can provide insights into critical events, transitions between different stages or conditions, or the effectiveness of interventions. Identifying these time points can be valuable for clinicians, as it helps in understanding the temporal patterns and making informed decisions about patient management and treatment strategies.

5.1 Threats to validity

In this section, we will be explaining different types of threats to validity, sourcing from the paper [7] and their relation with our research:

5.1.1 Conclusion validity

The concept of "statistical conclusion validity," initially referred to as the degree to which conclusions drawn from collected data are reasonable, focuses on the reasonableness of conclusions regarding relationships between factors. The presence of researcher bias presents a potential threat to conclusion validity as it can significantly influence the reached conclusions. Additionally, weak statistical analysis results can be subject to various interpretations based on the researcher's bias, leading to incorrect conclusions. In this thesis, we have consulted the experts in the field, data scientists, and doctors who are working in neonatal care at University Medical Center Utrecht. Our conclusions are reevaluated multiple times, therefore I do not think there is any bias toward our conclusion. However, there is always a potential for the application to generate inaccurate or misleading conclusions due to factors like flawed data input. In this sense, we are narrowing our conclusions on the UMC Utrecht data sample specifically and putting extra stress on the fact that we are focusing solely on this dataset. Introducing a dataset outside this environment may introduce additional bias.

5.1.2 Internal validity

Internal validity, on the other hand, pertains to the examination of causal relationships. It seeks to determine whether an experimental treatment or condition has a genuine impact and whether there is sufficient evidence to support the claim being made. Assessing internal validity involves carefully controlling variables, employing appropriate research designs, and minimizing confounding factors to accurately attribute any observed effects to the intended cause. One potential threat is selection bias, where the sample of patients or data used in the application may not be representative of the broader population or may be biased in some way. This can introduce a confounding factor and affect the generalizability of the application's results. As we mentioned in Conclusion Validity, our scope only covers the samples in UMC Utrecht and any external sample may introduce a threat to internal validity. In terms of measurement bias, which can arise if the data collected or measurements taken by the software are inaccurate or unreliable, we can start by mentioning that the samples taken are from a trustworthy affiliation. Not only this, multiple preprocessing phases are performed on the data to ensure that this bias is minimized. However, during the time this thesis is written, there may be confounding variables or uncontrolled factors that influence the observed outcomes in the application. The causal relationships which we didn't take care of, which might be discovered later, may be introduced to the system to minimize this thread.

5.1.3 Construct validity

Construct validity pertains to the extent to which a test or experiment accurately measures what it claims to measure. It examines whether the researcher successfully captures the intended constructs or variables of interest in their measurements. For example, in a diagnostic software application like the one mentioned in our thesis, construct validity would involve evaluating whether the algorithms and assessments accurately identify and measure the specific medical conditions they are designed to detect. For this purpose, we employed several testing metrics to prove that we developed a reliable system. The specific details are introduced in section 4.

5.1.4 External validity

External validity focuses on the extent to which research findings can be generalized or applied to populations, settings, or contexts beyond the specific study. It addresses the question of whether the

results hold and have relevance beyond the sample or case under investigation. In quantitative research, the chosen sample size plays a critical role in assessing external validity. A larger and more diverse sample increases the likelihood of generalizability, as it enhances the representation of the target population. We employed multiple sampling techniques to increase generalizability, however, our samples all originated from the UMC Utrecht which may introduce some form of external validity.

Chapter 6

Related work

In the following sections, we present insights into how existing scientific studies approached the current problem. You will also find information on how our study will contribute and highlight the specified scientific gap and answer the mentioned research question.

6.1 Previous Solutions

In the following sections, we present 4 common approaches that are previously developed for this scientific research area. We have to mention that as an outcome of our literature study, we observed that there is no prior research that tries to identify the outcome of asphyxiated patients directly. In that sense, our research is a novelty from a medical perspective. Therefore we tried to analyze related works which try to employ machine learning techniques to tackle problems in the same medical field. The remaining papers are collected under categories of different approaches:

6.1.1 Approach 1

The following approaches try to diagnose asphyxiated patients, mostly before birth. Below are the related papers with the specified technique:

6.1.1.1 Unveiling Neonatal Encephalopathy Patterns from Electronic Health Records

The study utilized a dataset [29] derived from the Electronic Health Records (EHR) system at VUMC, encompassing a vast collection of over 2.5 million patient records spanning 29 years, as approved by Vanderbilt IRB. Focusing on predicting Neonatal Encephalopathy (NE) before delivery, the dataset consisted of EHRs about both mothers and their newborns. To construct NE prediction models, relevant variables were extracted from the mothers' EHRs, and a combination of regularized logistic regression and Long Short-Term Memory (LSTM) models were employed, yielding an impressive accuracy of 89% and sensitivity of 82%.

6.1.1.2 Combining Methods of Deep Learning and Supervised Machine Learning for Enhanced Predictive Fetal Monitoring"

The paper [33] addresses the problem of extracting features from the CTG which monitors the baby inside the pregnant woman for fetal heart rate and uterine activity. They used a software package called SISPORTO [15] which is a program for automated analysis of cardiotocograms to evaluate uterine contractions and classification of decelerations. The classification is done in a hybrid way using neural networks, support vector machine, and k-means clustering which successfully classified 14 out of 24 samples. The extracted features are later fed into the developed application to be used in the prediction of umbilical cord pH level and acidosis which are important bio-markers for asphyxia. In Python, using the Keras library, two distinct LSTM models were created for each CTG data sample. The mentioned algorithm works with 85% accuracy.

6.1.1.3 Utilizion of Locally Embedded-Reduced Mel Frequency Cepstrum Coefficient (MFCC) Features and Autoencoders for Infant Asphyxia Detection

In this study [82], they investigated the utilization of autoencoders, a type of deep learning paradigm, for diagnosing infant asphyxia. Autoencoders, trained through unsupervised learning using the back-propagation algorithm, have been widely employed for feature extraction and reduction. Unlike other neural networks with distinct inputs and targets, autoencoders train themselves to approximate the inputs by employing a constrained number of hidden units. This constraint forces the network to capture the essential structure and distinctive features of the inputs, resulting in a compressed representation akin to Principal Component Analysis (PCA). Extensive experimentation reveals that the most effective autoencoder network achieves a classification accuracy of 92.82%.

6.1.1.4 Discrimination of Asphyxia Fetuses: Machine Learning Models Utilizing Clinical Indices and Cardiotocographic Features

In the study by Ribeiro et al. [60], a univariable Binary Logistic Regression (BLR) analysis was conducted on all independent variables, yielding odds ratios (ORs), 95% confidence intervals (CIs), and p-values. Spearman correlation coefficient was used to identify and eliminate redundant variables, selecting the one with the lowest significance value among variables with a correlation coefficient greater than 0.6. For non-linear cardiotocographic (CTG) variables with p less than 0.2, both univariate BLR models and Naive-Bayes models were developed using R, focusing on binomial logistic regression and the Naive Bayes packages. A classification cut-off of 0.5 was employed in model construction. To ensure unbiased evaluation, a two-fold cross-validation approach was utilized, with a training set (55%) and a testing set (45%). Considering the imbalanced distribution of the asphyxia and no asphyxia groups, an oversampling technique was applied in the training dataset to achieve a 0.3 probability of asphyxia, implemented using the ROSE R package. The area under the receiver operating characteristic curve (AUC) and its confidence interval were computed for the respective test set, demonstrating a sensitivity of 87% (13 out of 15) and a specificity of 100%.

6.1.1.5 A Machine Learning Approach for the Classification of Foetal Distress and Hypoxia

This study [4] explores the application of machine-learning techniques for diagnosing and detecting intrapartum foetal hypoxia by analyzing foetal clinical data. The research utilizes an open-source database provided by Physionet, consisting of signal and clinical data collected between 2010 and 2012 at the university hospital in Brno, Czech Republic. The dataset includes neonatal outcomes and potential risk factors. Experimental results reveal that the Random Forest model performs as the most accurate classifier, achieving a high area under the curve quality measure (0.95). The Neural Network (NN) model ranks second, demonstrating a predictive ability of 0.91 for both accuracy (ACC) and area under the curve (AUC). The simulation results also highlight the high AUC (0.94) of the Gradient Boosting Machine (GBM) model, albeit with slightly lower accuracy (0.88). While both RF and GBM show the ability to detect true positive cases (pathological) with a sensitivity of 0.87, GBM exhibits lower specificity (0.89) in identifying true negative cases (normal). Additionally, the Support Vector Machine (SVM) and GLMNET models demonstrate reasonable prediction abilities, achieving AUC values of 0.80 and 0.83, respectively. However, their overall accuracies are slightly lower (0.84 for SVM and 0.79 for GLMNET) compared to other classifiers. Although the K-Nearest Neighbors (KNN) classifier performs well in detecting true positive cases with a sensitivity value of 0.93, it displays lower overall accuracy (0.77) and is the weakest classifier in identifying true negative cases (specificity of 0.67).

6.1.2 Approach 2

This approach tries to apply prediction after birth within the first energy failure (6-hour window) to see whether or not the patients will develop HIE and need further treatment such as hypothermic treatment. Below are the related papers with the specified technique:

6.1.2.1 Utilizing a Multimodal Deep Neural Network to Predict Neonatal Intensive Care Unit Intubation Requirement within 3 Hours

In this study[37], the objective was to predict the potential need for intubation in neonatal patients within the first 48 hours of life. The researchers utilized a multimodal neural network model, incorporating time-

series datasets comprising demographic, physiological, and laboratory information. The model employed a multilayer perceptron (MLP) architecture, where the flattened time variable x(t) and measurement x(n) were processed through MLP blocks, including a fully connected layer, batch normalization, rectified linear units, and dropout for regularization. The resulting vectors were concatenated and analyzed using the final MLP block, which provided information about the probability of intubation. The model demonstrated strong performance with an area under the curve of 0.917, sensitivity of 85.2%, and specificity of 89.2%.

6.1.2.2 Enhancing Care for Neurologically Impaired Newborns through Artificial Intelligence Assistance

In this research [55], the focus is on exploring the effectiveness of Artificial Intelligence-based prediction systems in automating seizure diagnosis using newborns' EEG waveforms. The study involves collecting EEG data along with patient characteristics, which are then utilized as input for various classification models. To enhance model performance, a Linear Series Decomposition Learner is employed as a preprocessing step. The achieved accuracy reaches a peak of 93.5%, while the secondary analysis reveals a maximum accuracy of 89.1% for predicting asphyxia.

6.1.2.3 Enhancing Neonatal Hypoxic-Ischemic Encephalopathy Prediction through the Integration of Umbilical Cord Metabolites and Clinical Markers

Utilizing Logistic Regression with LASSO feature selection, a subset of metabolites essential for an effective multivariable prediction model for hypoxic-ischemic encephalopathy (HIE) was determined. LASSO, an algorithm for continuous subset selection, minimizes the impact of irrelevant predictors by constraining the coefficients' magnitude. Important predictors are retained, while less significant ones shrink or are eliminated, resulting in an accurate model with minimal variables. To validate the logistic regression results, complementary classification models were developed using the Random Forest (RF) Algorithm, employing the same input variables. Robustness and generalizability were ensured through 10-fold cross-validation with 10 Monte Carlo repetitions. Receiver Operator Characteristic (ROC) curve analyses were performed to evaluate the optimal classifier models. Additionally, the described feature selection methods were compared and integrated with previously identified clinical markers for outcome prediction. The integrated approach achieved an overall accuracy of 86%, an Area Under the Curve (AUC) of 0.90 (95% CI, 0.84-0.94), a positive predictive value of 80%, a negative predictive value of 92%, a sensitivity of 92%, and a specificity of 80% [57].

6.1.2.4 Modeling the Risk of Hypoxic-Ischemic Encephalopathy after Perinatal Asphyxia Using Predictive Methods

Another approach is in paper [51] "Predictive modeling of hypoxic-ischaemic encephalopathy risk following perinatal asphyxia". The paper mentions that the 40% of patients who are not qualified for the treatment with the clinicians were actually eligible for the treatment and even when correctly classified around 20% of the patients end up with brain injury because of the strict 6-hour window. The Apgar score in 1, 5, and 10 minutes, postnatal blood gas values are collected and patients are periodically monitored by EEGs continuously within the first 24-hour period. Their change gives strong information on the prediction of the disease over the time period. Later the level of severity is determined by the Sarnat score and finalized by a qualified professional. The dataset used was missing some parameters which were filled by an R-package called "mice" by mean matching. All the information is then fed into a Random Forest algorithm continuously to achieve improved prediction within the 6-hour window. The study achieved a 97.3% accuracy.

6.1.3 Approach 3

This approach mainly tries to tackle the problem in a non-invasive manner such as evaluating child cries to understand whether or not babies need treatment. Below are the related papers with the specified technique:

6.1.3.1 GPU-Accelerated Deep Learning for Asphyxia Detection in Newborns

In addition to the previous 2 approaches the third approach is a non-intrusive one. The paper [50] suggests that the way babies cry gives various information on the status of the baby. If the acoustics

are investigated via the collected cry signals this will help the researchers to predict the occurrence of a disease. In this study, they trained a convolutional neural network with 3 different types of baby cries: typical cry signals, cry signals from patients with hearing difficulties, and infants with asphyxia. They achieved a 94% accuracy. The main advantage is that it doesn't require blood drawls or scans and works with high accuracy.

6.1.3.2 Utilizing Neural Transfer Learning for Diagnosis of Perinatal Asphyxia Based on Infant Cry Analysis

[58] The correlation between crying and respiration in infants has long been recognized, as both processes rely on the functionality of respiratory muscles and are coordinated by the same brain regions. Early studies, such as the work by Michelsson et al. in the 1970s and 1980s, revealed distinct characteristics in the cries of asphyxiated newborns, including shorter duration, lower amplitude, increased higher fundamental frequency, and a significant increase in the "rising" melody type. In this study, the Chillanto Infant Cry database was utilized to explore the application of statistical learning techniques in classifying various conditions, including deafness, asphyxia, and pain. The authors employed audio representations such as linear predictive coefficients (LPC) and Mel-frequency cepstral coefficients (MFCC), training a time delay neural network as the classifier. Leveraging the effectiveness of Residual Neural Networks (ResNets) in speech-related tasks, a consistent architecture with 6 residual blocks and additional convolutional layers was adopted to ensure fair comparisons across source tasks and enable transfer learning. By transforming the 2D MFCC representation of audio signals and extracting fixed dimension embeddings through average pooling, the model achieved an accuracy of 72.7% and a recall of 68% in predicting the classes of interest.

6.1.3.3 Leveraging GPU-Accelerated Deep Learning for Neonatal Asphyxia Detection

This project [62] employs the NVIDIA Deep Learning GPU Training System (DIGITS) to construct and train a Deep Neural Network (DNN) for image classification and real-time object detection. The Baby Chillanto Database, obtained from the National Institute of Astrophysics and Optical Electronics, CONACYT, Mexico, consists of 340 asphyxiated and 1049 normal cry tests, partitioned as 60:20:20 for preparation, cross-validation, and performance evaluation. The software was configured following the author's instructions, with the dataset split into 75% for training and 25% for validation. This approach achieved an impressive accuracy of 92%

6.1.4 Approach 4

Studies concerning finding the optimal feature sets within the 24-hour window.

6.1.4.1 Analyzing Signals and Classifying Clinically Significant Parameters in Neonatal Resuscitation

In this paper [75] they experimented on newborns who experienced birth asphyxia and tried to identify the outcome of the baby using the features obtained after therapeutic interventions and observed their response using machine learning frameworks. They evaluated the accuracy of their implementation comparing the outcome of the baby after 24 hours of birth. All possible outcomes of such a scenario include: normal, being placed in the neonatal care unit, and death. They used a nested cross-validation scheme for classification and SMOTE Boost and RUS Boost for balancing out the dataset to avoid bias. The experiments included the following: Utilizing all available features, this study focuses on identifying neonatal outcomes within 24 hours of birth. Additionally, for newborns initially in poor condition, the study investigates the identification of neonatal outcomes using all features and only initial and treatment parameters. The results demonstrated a precision of 96% and a recall of 89% for normal outcomes, while episodes ending in death showed a precision of 47% and a recall of 74%. These findings highlight the potential of the extracted features in describing the 24-hour neonatal outcome.

6.1.5 Comparison Of The Existing Approaches

In the below table you will be able to find information on what kind of content is covered by the above-mentioned four approaches (A1, A2, A3, and A4, respectively) and what should be our main concern based on the unanswered gap:

Contents Covered	A1	A2	A3	A4
Prediction Of Asphxia within 0-6 hour window	x	x	x	
Prediction Of Outcome after the 6 hour window				x
Use of Invasive BioMarkers Like Blood Tests	x	x		x
Use of Non-Invasive BioMarkers Like Sound of Cry			x	
Use of Discrete Data For Prediction	x		x	x
Use of Continous Data For Prediction		x		

Table 6.1: Comparison of The Contents For Previous Approaches

As you can see from the above table, based on my literature study, there is a research gap for the cases using continuous data after 6 hours of birth for the prediction of the neurodevelopmental outcome of asphyxiated and cooled-down patients.

Chapter 7

Conclusion

Asphyxia refers to a condition characterized by a sudden and severe lack of oxygen and reduced blood flow to the brain, resulting from birth abnormalities. This oxygen deficiency primarily affects the distribution of blood to vital organs. Consequently, newborn babies may face neurological disorders in their future, such as seizures. This issue is prevalent not only in the Netherlands but also worldwide, contributing to an infant mortality rate of up to 23%. The objective of our research was to determine whether it is possible to predict the outcome of asphyxiated babies by analyzing observations taken between 6 hours and 4 days after birth. At the time of this thesis publication, no previous studies directly addressed this specific problem. However, there are closely related research efforts that provide a more comprehensive understanding of the issue, which are elaborated upon in section 6. To address this research gap, we pursued various different approaches and compared their results to come up with an optimal solution. Although the first approaches were unsuccessful, it provided valuable insights into the data's characteristics, ultimately aiding us in succeeding with the final approach.In the selected approach, we developed a modified version of the advanced neural network structure known as AlexNet. The network was trained using generated recurrence plots. Our approach achieved an impressive average accuracy of 93.3%. Each observation was evaluated individually, and the detailed evaluation metrics are provided separately in section 4. More details about the highlighted core sections in this report is also provided in the Section 7.1 in Appendix.

Based on our study, we conclude that machine learning models can assist doctors in diagnosing the outcomes of asphyxiated patients without relying solely on MRI scores. Furthermore, we identified the time points at which the severity level of a patient begins to change. This information holds significant value for clinicians, as it facilitates an understanding of temporal patterns and aids in making informed decisions regarding patient management and treatment strategies.

7.1 Future work

As a result of our work, we can provide the following future contributions:

- Treatment Guidance: The method we have developed for predicting the outcomes of asphyxiated babies has the potential to significantly impact the treatment of this condition in the future. One avenue for future work is to explore how our predictions can be integrated into clinical practice to guide treatment decisions. By combining our machine learning models with existing medical protocols, clinicians may be able to personalize treatment plans for individual patients based on their predicted outcomes. This could involve adjusting the intensity and duration of therapeutic interventions, such as hypothermia therapy or oxygen supplementation, to optimize outcomes for asphyxiated infants. Future research should focus on conducting clinical trials to evaluate the effectiveness of incorporating our predictive models into the treatment decision-making process and assess their impact on long-term outcomes.
- Advanced Analysis Techniques: While our current study utilized state-of-the-art machine learning techniques, there is still room for further exploration and refinement of analysis methods. Future work should aim to incorporate more advanced techniques to gain a deeper understanding of the underlying patterns and mechanisms associated with asphyxia. For instance, exploring deep learning architectures beyond AlexNet, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), may reveal additional insights and potentially improve prediction

accuracy. Additionally, employing advanced feature selection and extraction techniques, such as transfer learning or autoencoders, could enhance the identification of relevant features and reduce noise in the dataset. Investigating alternative methods for handling imbalanced datasets, such as oversampling or undersampling techniques, may also be beneficial. By employing these advanced analysis techniques, we can potentially uncover more nuanced relationships between the observed variables and the outcomes of asphyxiated babies.

- Dataset Enhancement and Generalizability: In order to improve the generalizability of our predictive models, future studies should focus on enhancing the dataset used in this research. One possible avenue is to augment the existing dataset with data from other hospitals and healthcare centers across different countries. This would enable a more diverse representation of asphyxiated infants, accounting for potential variations in demographics, healthcare practices, and genetic factors. Moreover, collecting longitudinal data spanning a longer timeframe would provide a more comprehensive understanding of the temporal patterns and trajectories of asphyxia outcomes. It would be valuable to assess the long-term effects and developmental outcomes of these infants beyond the immediate post-birth period. Additionally, incorporating additional clinical variables, such as gestational age, birth weight, and maternal health indicators, would enrich the dataset and potentially improve the accuracy of predictions. Collaborative efforts and multi-center studies should be undertaken to facilitate the collection and sharing of such data, ensuring a larger and more representative sample for analysis.
- Prospective Validation and Clinical Integration: To validate the effectiveness and real-world applicability of our predictive models, future work should involve prospective validation studies conducted in clinical settings. Collaborating with healthcare institutions and clinicians, researchers can implement our models as decision support tools to aid in the management of asphyxiated infants. By prospectively collecting data from newborns and evaluating the predictions made by the models, we can assess their performance in real-time scenarios. This would provide valuable feedback and insights into the strengths and limitations of our models, helping to refine and optimize their performance. Moreover, conducting qualitative studies involving healthcare providers to understand their perspectives on using these predictive models can offer insights into the barriers and facilitators of implementation. Integration of the models into clinical workflows and assessment of their impact on decision-making, patient outcomes, and resource allocation should also be explored.
- Ethical Considerations and Interpretability: As with any implementation of machine learning in healthcare, it is crucial to address ethical considerations and ensure transparency and interpretability of the models. Future work should focus on developing explainable AI techniques to provide clinicians with insights into the decision-making process of the models. This would aid in building trust and understanding among healthcare providers, enabling them to confidently utilize the predictions in their clinical practice. Additionally, efforts should be made to ensure fairness and equity in the deployment of these models, considering potential biases and disparities in healthcare delivery. Ongoing monitoring and assessment of the models' performance in real-world settings should be conducted to detect and mitigate any unintended consequences or biases that may arise.

In conclusion, there are several promising avenues for future work to further advance the field of predicting outcomes for asphyxiated infants. By integrating our predictive models into clinical practice, exploring advanced analysis techniques, enhancing the dataset's generalizability, conducting prospective validation studies, and addressing ethical considerations, we can continue to improve the accuracy and utility of these models. These efforts have the potential to revolutionize the treatment and care provided to asphyxiated infants, ultimately improving their long-term outcomes and quality of life.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, dr. Adam Belloum, for accepting me as one of his students. He guided me throughout this thesis. He was always available and ready to provide me with the required expertise.

I would like to thank my external supervisor, dr. Richard Bartels, for giving me the opportunity to complete this thesis in collaboration with UMC Utrecht. He always provided me with constructive criticism and his in-depth experience about software applications in the medical field allowed me to look at the problem from different perspectives. I would like to thank, dr. Thomas Alderliesten, for answering my questions as quickly as possible even when he was extremely busy and providing feedback about medicine-related issues.

I want to thank the class of 2022-2023 UvA Masters Software Engineering for making it a beautiful and fun year. I have met really smart and fun people, and it was an amazing experience to spend this last year with like-minded people.

Finally, the biggest thanks go to my parents. I would not be able to complete this program without their continuous support and encouragement. They were with me through all the challenges I have faced. They were the secret heroes of my master's journey and I will be forever grateful for having them.

Bibliography

[1] URL: https://www.exxactcorp.com/blog/Deep-Learning/difference-between-ai-machine-learning-and-deep-learning.

[2] URL: https://www.researchgate.net/figure/Typical-examples-of-recurrence-plots-right-column-for-time-series-data-with-different_fig1_332493834.

[3] URL: https://gombru.github.io/2018/05/23/cross_entropy_loss/.

[4] Rounaq Abbas et al. "Classification of foetal distress and hypoxia using machine learning approaches". In: *Intelligent Computing Methodologies: 14th International Conference, ICIC 2018, Wuhan, China, August 15-18, 2018, Proceedings, Part III 14*. Springer. 2018, pp. 767–776.

[5] Caroline E Ahearne, Geraldine B Boylan, and Deirdre M Murray. "Short and long term prognosis in perinatal asphyxia: An update". In: *World journal of clinical pediatrics* 5.1 (2016), p. 67.

[6] Fatema Al Amrani et al. "Prediction of outcome in asphyxiated newborns treated with hypothermia: Is a MRI scoring system described before the cooling era still useful?" In: *European Journal of Paediatric Neurology* 22.3 (2018), pp. 387–395. ISSN: 1090-3798. DOI: https://doi.org/10.1016/j.ejpn.2018.01.017. URL: https://www.sciencedirect.com/science/article/pii/S1090379817317841.

[7] Apostolos Ampatzoglou et al. "Identifying, categorizing and mitigating threats to validity in software engineering secondary studies". In: *Information and Software Technology* 106 (2019), pp. 201–230.

[8] AndreaManeroBastin. *How to configure the number of layers and nodes in a neural network*. 2019. URL: https://www.datasciencecentral.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural/.

[9] Mehmet Arslan et al. "SMOTE and Gaussian Noise Based Sensor Data Augmentation". In: *2019 4th International Conference on Computer Science and Engineering (UBMK)*. 2019, pp. 1–5. DOI: 10.1109/UBMK.2019.8907003.

[10] Hala M Attia et al. *Diagnostic value of neuron specific enolase in serum for outcome of ...* 2016. URL: https://journals.ekb.eg/article_4672_74cbc74275cbaf330edf738ea444ccc6.pdf.

[11] Silvio Barra et al. "Deep learning and time series-to-image encoding for financial forecasting". In: *IEEE/CAA Journal of Automatica Sinica* 7.3 (2020), pp. 683–692.

[12] Jason Bell. "What is machine learning?" In: *Machine Learning and the City: Applications in Architecture and Urban Design* (2022), pp. 207–216.

[13] Dalip Kumar Bhagwani et al. "To Study the Correlation of Thompson Scoring in Predicting Early Neonatal Outcome in Post Asphyxiated Term Neonates." In: *Journal of clinical and diagnostic research : JCDR* 10 11 (2016), SC16–SC19.

[14] Jason Brownlee. *How to configure the learning rate when training deep learning neural networks*. 2019. URL: https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/.

[15] Diogo Ayres-de Campos et al. "Sisporto 2.0: A program for automated analysis of cardiotocograms". In: *The Journal of Maternal-Fetal Medicine* 9.5 (2000), pp. 311–318. DOI: https://doi.org/10.1002/1520-6661(200009/10)9:5<311::AID-MFM12>3.0.CO;2-9. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/1520-6661%28200009/10%299%3A5%3C311%3A%3AAID-MFM12%3E3.0.CO%3B2-9. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/1520-6661%28200009/10%299%3A5%3C311%3A%3AAID-MFM12%3E3.0.CO%3B2-9.

[16] Lina F Chalak. "Inflammatory biomarkers of birth asphyxia". In: *Clinics in perinatology* 43.3 (2016), pp. 501–510.

[17] Yu Chen, Rui Chang, and Jifeng Guo. "Effects of Data Augmentation Method Borderline-SMOTE on Emotion Recognition of EEG Signals Based on Convolutional Neural Network". In: *IEEE Access* 9 (2021), pp. 47491–47502. DOI: 10.1109/ACCESS.2021.3068316.

[18] V. Cherkassky and H. Lari-Najafi. "Data representation for diagnostic neural networks". In: *IEEE Expert* 7.5 (1992), pp. 43–53. DOI: 10.1109/64.163672.

[19] Phillip Chlap et al. "A review of medical image data augmentation techniques for deep learning applications". In: *Journal of Medical Imaging and Radiation Oncology* 65.5 (2021), pp. 545–563.

[20] Vijay Choubey. *Undestanding recurrent neural network (RNN) and long short term memory(lstm).* 2020. URL: https://medium.com/analytics-vidhya/undestanding-recurrent-neural-network-rnn-and-long-short-term-memory-lstm-30bc1221e80d.

[21] *Classic Machine Vision vs. Deep Learning.* URL: https://www.mvtec.com/technologies/deep-learning/classic-machine-vision-vs-deep-learning.

[22] ODSC Community. *Understanding the mechanism and types of recurrent neural networks.* 2021. URL: https://opendatascience.com/understanding-the-mechanism-and-types-of-recurring-neural-networks/.

[23] Antonia Creswell et al. "Generative Adversarial Networks: An Overview". In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 53–65. DOI: 10.1109/MSP.2017.2765202.

[24] Erika D. *Accuracy, Recall amp; Precision.* 2019. URL: https://medium.com/@erika.dauria/accuracy-recall-precision-80a5b6cbd28d.

[25] *Deep Learning for Computer Vision: The abridged guide.* URL: https://www.run.ai/guides/deep-learning-for-computer-vision#:~:text=Deep\%20learning\%20is\%20a\%20machine,through\%20multiple\%20layers\%20of\%20neurons..

[26] Li Deng and Yang Liu. *Deep learning in natural language processing.* Springer, 2018.

[27] Saul Dobilas. *LSTM recurrent neural networks-how to teach a network to remember the past.* 2022. URL: https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e.

[28] Hassan Ismail Fawaz et al. "Data augmentation using synthetic data for time series classification with deep residual networks". In: *arXiv preprint arXiv:1808.02455* (2018).

[29] Cheng Gao et al. "A Deep Learning Approach to Predict Neonatal Encephalopathy from Electronic Health Records". In: *2019 IEEE International Conference on Healthcare Informatics (ICHI)*. 2019, pp. 1–7. DOI: 10.1109/ICHI.2019.8904667.

[30] Matteo Giampietri and Pascal Biver. "Recovery of aEEG Patterns at 24 Hours of Hypothermia Predicts Good Neurodevelopmental Outcome". In: *Journal of Neonatal Biology* 5 (Jan. 2016). DOI: 10.4172/2167-0897.1000230.

[31] Ernest M Graham et al. "Blood biomarkers for evaluation of perinatal encephalopathy-State of the art". In: *Current opinion in pediatrics* 30.2 (2018), p. 199.

[32] Floris Groenendaal et al. "Introduction of hypothermia for neonates with perinatal asphyxia in the Netherlands and Flanders". In: *Neonatology* 104.1 (2013), pp. 15–21.

[33] Vinayaka Gude and Steven Corns. "Integrated Deep Learning and Supervised Machine Learning Model for Predictive Fetal Monitoring". In: *Diagnostics* 12.11 (2022), p. 2843. ISSN: 2075-4418. DOI: 10.3390/diagnostics12112843. URL: http://dx.doi.org/10.3390/diagnostics12112843.

[34] Xing Hao, Guigang Zhang, and Shang Ma. "Deep learning". In: *International Journal of Semantic Computing* 10.03 (2016), pp. 417–439.

[35] Sahar M. A. Hassanein et al. "Human Umbilical Cord Blood CD34-Positive Cells as Predictors of the Incidence and Short-Term Outcome of Neonatal Hypoxic-Ischemic Encephalopathy: A Pilot Study". In: *Journal of Clinical Neurology (Seoul, Korea)* 13 (2016), pp. 84 –90.

[36] G Houghes. "On the mean accuracy of statistical pattern recognition". In: *IEEE Trans. Inform. Theory* 14.1 (1968), pp. 55–63.

[37] Jueng-Eun Im et al. "Predicting the Need for Intubation Within 3 Hours in the Neonatal Intensive Care Unit Using a Multimodal Deep Neural Network". In: (2022).

[38] Ken Imai et al. "MRI changes in the thalamus and basal ganglia of full-term neonates with perinatal asphyxia". In: *Neonatology* 114 (2018), pp. 253–260.

[39] Brian Kenji Iwana and Seiichi Uchida. "An empirical survey of data augmentation for time series classification with neural networks". In: *Plos one* 16.7 (2021), e0254841.

[40] Stefan Jaeger. "The golden ratio of learning and momentum". In: *arXiv preprint arXiv:2006.04751* (2020).

[41] Jeremy Jordan. *Evaluating a machine learning model.* 2018. URL: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/.

[42] Shashmi Karanam. *Curse of dimensionality-A "curse" to Machine Learning.* 2021. URL: https://towardsdatascience.com/curse-of-dimensionality-a-curse-to-machine-learning-c122ee33bfeb.

[43] Emanuela Locci et al. "Exploring Perinatal Asphyxia by Metabolomics". In: *Metabolites* 10.4 (2020), p. 141. ISSN: 2218-1989. DOI: 10.3390/metabo10040141. URL: http://dx.doi.org/10.3390/metabo10040141.

[44] Siyuan Lu, Zhihai Lu, and Yu-Dong Zhang. "Pathological brain detection based on AlexNet and transfer learning". In: *Journal of computational science* 30 (2019), pp. 41–47.

[45] Kiran Maharana, Surajit Mondal, and Bhushankumar Nemade. "A review: Data pre-processing and data augmentation techniques". In: *Global Transitions Proceedings* (2022).

[46] Batta Mahesh. "Machine learning algorithms-a review". In: *International Journal of Science and Research (IJSR).[Internet]* 9 (2020), pp. 381–386.

[47] Bruce G Marcot and Anca M Hanea. "What is an optimal value of k in k-fold cross-validation in discrete Bayesian network analysis?" In: *Computational Statistics* 36.3 (2021), pp. 2009–2031.

[48] Parthiban Marimuthu. *Dropout regularization in Deep learning.* 2022. URL: https://www.analyticsvidhya.com/blog/2022/08/dropout-regularization-in-deep-learning/#:~:text=Dropout\%20is\%20a\%20regularization\%20method,connectedness\%20to\%20the\%20preceding\%20layer..

[49] Joydwip Mohajon. *Confusion matrix for your multi-class machine learning model.* 2021. URL: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826.

[50] Minal Moharir et al. "Identification of asphyxia in newborns using gpu for deep learning". In: *2017 2nd International Conference for Convergence in Technology (I2CT)*. 2017, pp. 236–239. DOI: 10.1109/I2CT.2017.8226127.

[51] Catherine Mooney et al. "Predictive modelling of hypoxic ischaemic encephalopathy risk following perinatal asphyxia". In: *Heliyon* 7.7 (2021), e07411. ISSN: 2405-8440. DOI: https://doi.org/10.1016/j.heliyon.2021.e07411. URL: https://www.sciencedirect.com/science/article/pii/S2405844021015140.

[52] Felipe A. Moreno. *Sparse categorical cross-entropy vs categorical cross-entropy.* 2021. URL: https://fmorenovr.medium.com/sparse-categorical-cross-entropy-vs-categorical-cross-entropy-ea01d0392d28.

[53] Naveen. *Difference between sigmoid and Softmax activation function?* 2023. URL: https://www.nomidl.com/deep-learning/what-is-the-difference-between-sigmoid-and-softmax-activation-function/.

[54] Kartik Nighania. *Various ways to evaluate a machine learning models performance.* 2019. URL: https://towardsdatascience.com/various-ways-to-evaluate-a-machine-learning-models-performance-230449055f15.

[55] Ejay Nsugbe. "Artificial Intelligence-assisted Care for Human Newborns with Neurological Impairments". In: *Digital Technologies Research and Applications* 1.2 (2022), pp. 35–47.

[56] Daragh S. O'Boyle et al. "Improvement in the Prediction of Neonatal Hypoxic-Ischemic Encephalopathy with the Integration of Umbilical Cord Metabolites and Current Clinical Makers". In: *The Journal of Pediatrics* 229 (2021), 175–181.e1. ISSN: 0022-3476. DOI: https://doi.org/10.1016/j.jpeds.2020.09.065. URL: https://www.sciencedirect.com/science/article/pii/S0022347620312579.

[57] Daragh S O'Boyle et al. "Improvement in the prediction of neonatal hypoxic-ischemic encephalopathy with the integration of umbilical cord metabolites and current clinical makers". In: *The Journal of Pediatrics* 229 (2021), pp. 175–181.

[58] Charles C Onu et al. "Neural transfer learning for cry-based diagnosis of perinatal asphyxia". In: *arXiv preprint arXiv:1906.10199* (2019).

[59] Shreya U. Patil. *Loss functions and optimizers in ML models*. 2023. URL: https://medium.com/geekculture/loss-functions-and-optimizers-in-ml-models-b125871ff0dc#:~:text=A\%20loss\%20function\%20measures\%20the,to\%20minimize\%20the\%20loss\%20function..

[60] Maria Ribeiro et al. "Machine learning models based on clinical indices and cardiotocographic features for discriminating asphyxia fetuses—Porto retrospective intrapartum study". In: *Frontiers in Public Health* 11 (2023).

[61] Ranit Roy. *Understanding all optimizers in deep learning*. 2022. URL: https://krishnaik.in/2022/03/28/understanding-all-optimizers-in-deep-learning/#:~:text=Adam\%20is\%20the\%20best\%20optimizer,descent\%20is\%20the\%20best\%20option..

[62] MU Sachin et al. "GPU based deep learning to detect asphyxia in neonates". In: *Indian J. Sci. Technol* 10.3 (2017).

[63] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. "Activation functions in neural networks". In: *Towards Data Sci* 6.12 (2017), pp. 310–316.

[64] Pramila P. Shinde and Seema Shah. "A Review of Machine Learning and Deep Learning Applications". In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. 2018, pp. 1–6. DOI: 10.1109/ICCUBEA.2018.8697857.

[65] Connor Shorten and Taghi M Khoshgoftaar. "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1 (2019), pp. 1–48.

[66] Aayush Kumar Singha et al. *Application of Machine Learning in Analysis of Infant Mortality and its Factors*. Feb. 2016. DOI: 10.13140/RG.2.1.3857.3687.

[67] Sonam, Shrutiparna, and Vinita. *How to interpret "loss" and "accuracy" for a machine learning model*. 2019. URL: https://intellipaat.com/community/368/how-to-interpret-loss-and-accuracy-for-a-machine-learning-model#:~:text=Loss\%20value\%20implies\%20how\%20poorly,the\%20form\%20of\%20a\%20percentage..

[68] Swapna. *Convolutional Neural Network: Deep learning*. 2022. URL: https://developersbreach.com/convolution-neural-network-deep-learning/.

[69] Keras Team. *Keras documentation: Layer weight regularizers*. URL: https://keras.io/api/layers/regularizers/.

[70] John Terra. *What is a ROC curve, and how do you use it in performance modeling?: Simplilearn*. 2022. URL: https://www.simplilearn.com/what-is-a-roc-curve-and-how-to-use-it-in-performance-modeling-article.

[71] Marianne Thoresen et al. "MRI combined with early clinical variables are excellent outcome predictors for newborn infants undergoing therapeutic hypothermia after perinatal asphyxia". In: *EClinicalMedicine* 36 (2021), p. 100885. ISSN: 2589-5370. DOI: https://doi.org/10.1016/j.eclinm.2021.100885. URL: https://www.sciencedirect.com/science/article/pii/S2589537021001656.

[72] Vinícius Trevisan. *Interpreting ROC curve and ROC AUC for Classification Evaluation*. 2022. URL: https://towardsdatascience.com/interpreting-roc-curve-and-roc-auc-for-classification-evaluation-28ec3983f077.

[73] Vinícius Trevisan. *Multiclass classification evaluation with ROC curves and Roc Auc*. 2022. URL: https://towardsdatascience.com/multiclass-classification-evaluation-with-roc-curves-and-roc-auc-294fd4617e3a.

[74] Neelam Tyagi. *L2 vs L1 regularization in machine learning: Ridge and Lasso regularization*. URL: https://www.analyticssteps.com/blogs/l2-and-l1-regularization-machine-learning.

[75] Jarle Urdal et al. "Signal processing and classification for identification of clinically important parameters during neonatal resuscitation". In: *2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*. 2017, pp. 547–552. DOI: 10.1109/ICSIPA.2017.8120672.

[76] Abhishek Verma. *Generative Adversarial Network*. URL: https://www.linkedin.com/pulse/generative-adversarial-network-abhishek-verma.

[77] Wojciech Walas et al. "Methods for assessing the severity of perinatal asphyxia and early prognostic tools in neonates with hypoxic-ischemic encephalopathy treated with therapeutic hypothermia". In: *Advances in Clinical and Experimental Medicine* 29.8 (2020).

[78] Kunfeng Wang et al. "Generative adversarial networks: introduction and outlook". In: *IEEE/CAA Journal of Automatica Sinica* 4.4 (2017), pp. 588–598. DOI: 10.1109/JAS.2017.7510583.

[79] Zhiguang Wang and Tim Oates. "Imaging time-series to improve classification and imputation". In: *arXiv preprint arXiv:1506.00327* (2015).

[80] Lauren C Weeke et al. "A novel magnetic resonance imaging score predicts neurodevelopmental outcome after perinatal asphyxia and therapeutic hypothermia". In: *The Journal of pediatrics* 192 (2018), pp. 33–40.

[81] Chao-Lung Yang, Zhi-Xuan Chen, and Chen-Yi Yang. "Sensor classification using convolutional neural network by encoding multivariate time series as two-dimensional colored images". In: *Sensors* 20.1 (2019), p. 168.

[82] IM Yassin et al. "Infant asphyxia detection using autoencoders trained on locally linear embedded-reduced Mel Frequency Cepstrum Coefficient (MFCC) features". In: *Journal of Fundamental and Applied Sciences* 9.3S (2017), pp. 716–729.

[83] Tao Zhou et al. "GAN review: Models and medical image fusion applications". In: *Information Fusion* 91 (2023), pp. 134–148.

[84] IA Zonnenberg et al. "Comparison of psychomotor outcome in patients with perinatal asphyxia with versus without therapeutic hypothermia at 4 years using the Ages and Stages Questionnaire screening tool". In: *european journal of paediatric neurology* 20.4 (2016), pp. 545–548.

Appendix A

Appendix

A.1 Progression of Results On Daily Basis Prior Day 6 (Before The MRI Scan)

The main goal of the project is to provide a prognosis before the MRI score on day 6. In this sense, it is also important to test the model performance given different days. Below you will be able to see the progression of evaluation metrics for different observations for measurements taken on (Day 1), (Day 1 and 2), (Days 1,2 and 3), (Day 1,2,3 and 4), (Days 1,2,3,4 and 5) and finally (Days 1,2,3,4,5 and 6). From the results we have obtained in this subsection, we can conclude that the results prior to the MRI Scan, are accurate and confident enough to provide a final prognosis.

A.1.1 Heart Rate Frequency

Below you will be able to find measurements concerning the heart rate frequency on separate days.

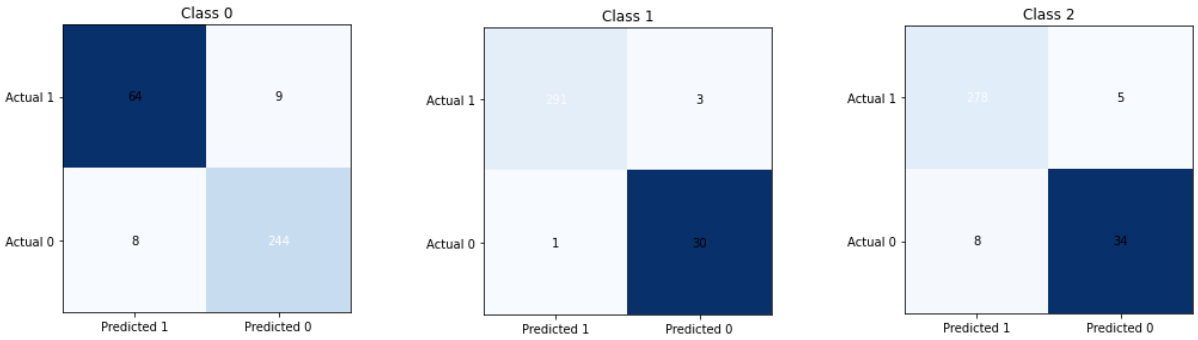
A.1.1.1 Evaluation Results on Day 1 Measurements

Below are the calculated testing metric results for the Heart Rate model for measurements taken on Day 1 :

Table A.1: Testing Metrics

Metric	Value
Accuracy	0.9476923076923077
Categorical cross-entropy loss	0.29401737019689556
Precision	0.92
Recall	0.92
F1 Score	0.91

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.1: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Heart Rate Frequency Observation based on Day 1 Measurements

ROC Curve:

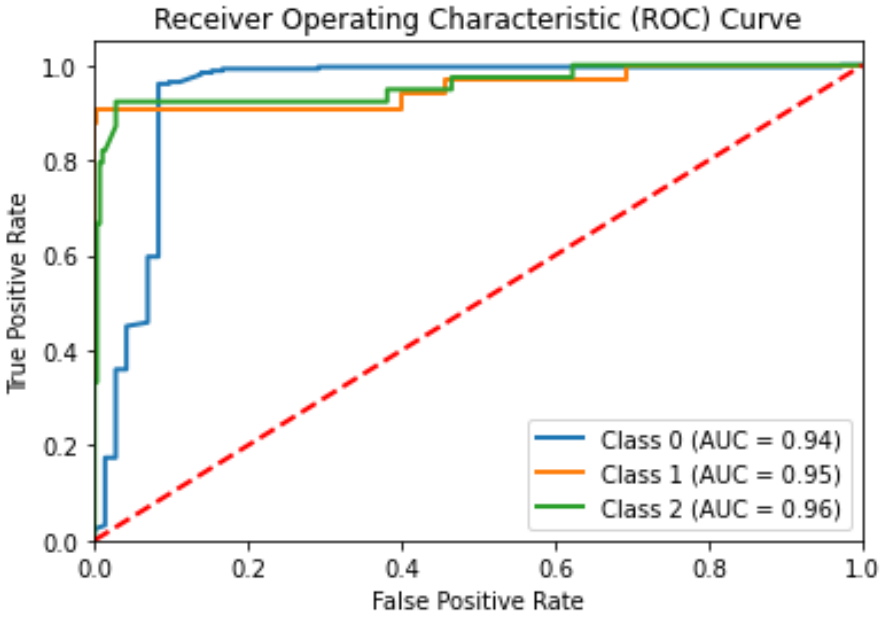


Figure A.2

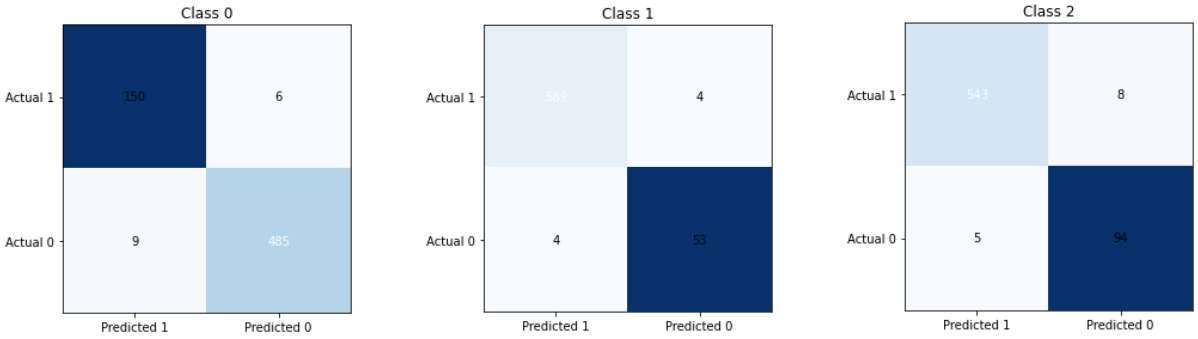
A.1.1.2 Evaluation Results on Day 1 and 2 Measurements

Below are the calculated testing metric results for the Heart Rate model for measurements taken on Day 1 and Day 2 :

Table A.2: Testing Metrics

Metric	Value
Accuracy	0.9723076923076923
Categorical cross-entropy loss	0.18239515519149227
Precision	0.95
Recall	0.95
F1 Score	0.95

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.3: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Heart Rate Frequency Observation based on Day 1 Measurements

ROC Curve:

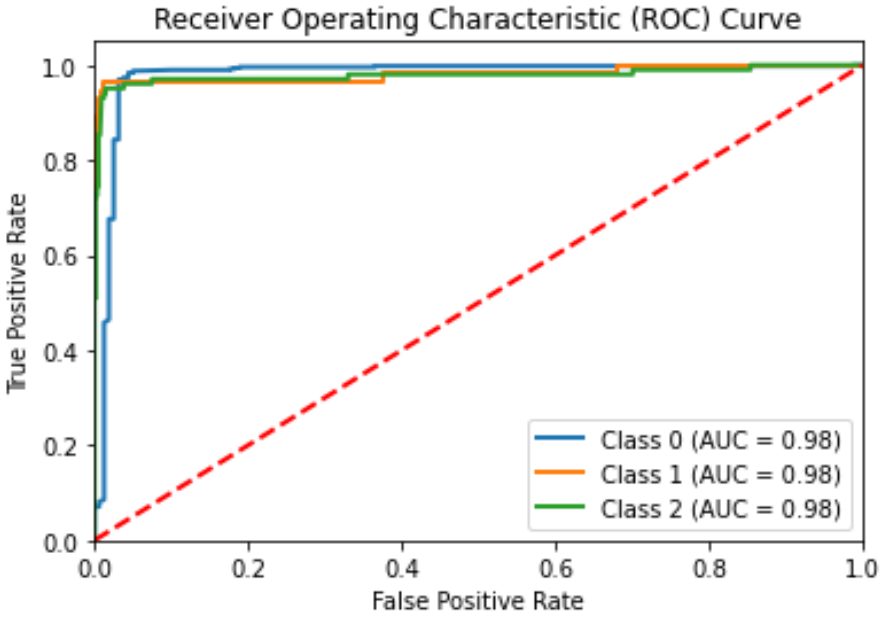


Figure A.4

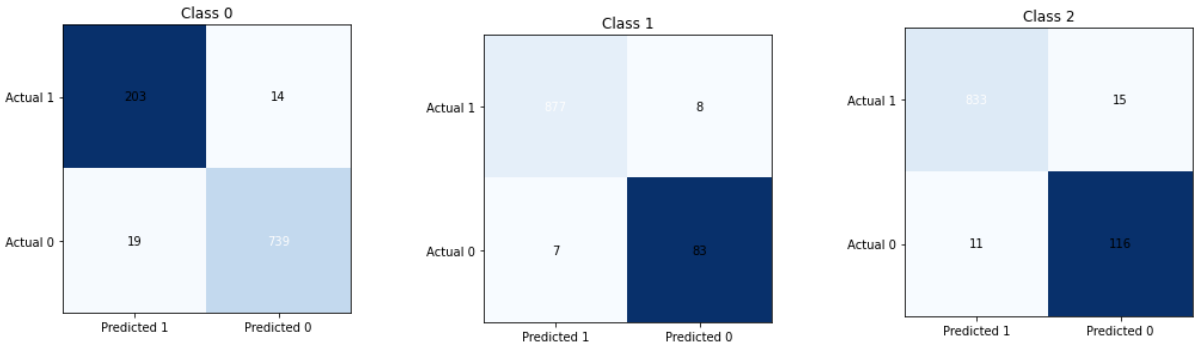
A.1.1.3 Evaluation Results on Day 1, 2, and 3 Measurements

Below are the calculated testing metric results for the Heart Rate model for measurements taken on Day 1, Day 2, and Day 3 :

Table A.3: Testing Metrics

Metric	Value
Accuracy	0.9641025641025641
Categorical cross-entropy loss	0.23993666138194616
Precision	0.97
Recall	0.96
F1 Score	0.96

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.5: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Heart Rate Frequency Observation based on Day 1 Measurements

ROC Curve:

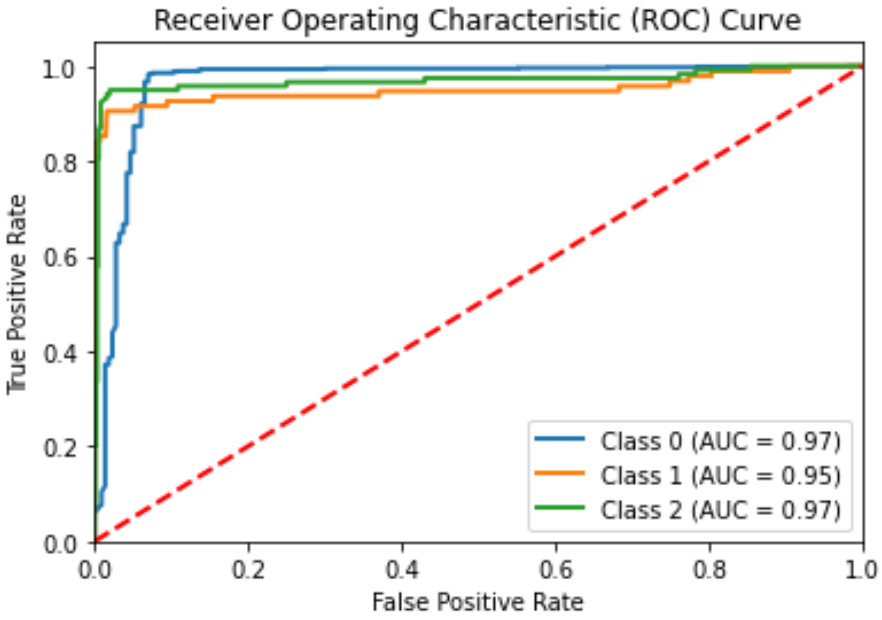


Figure A.6

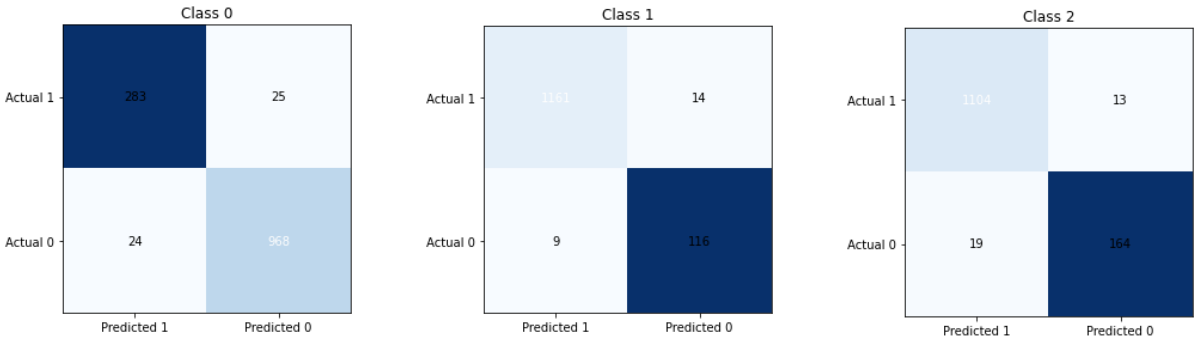
A.1.1.4 Evaluation Results on Day 1, 2, 3 and 4 Measurements

Below are the calculated testing metric results for the Heart Rate model for measurements taken on Day 1, Day 2, Day 3, and Day 4 :

Table A.4: Testing Metrics

Metric	Value
Accuracy	0.96
Categorical cross-entropy loss	0.21793777588973345
Precision	0.96
Recall	0.96
F1 Score	0.96

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.7: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Heart Rate Frequency Observation based on Day 1 Measurements

ROC Curve:

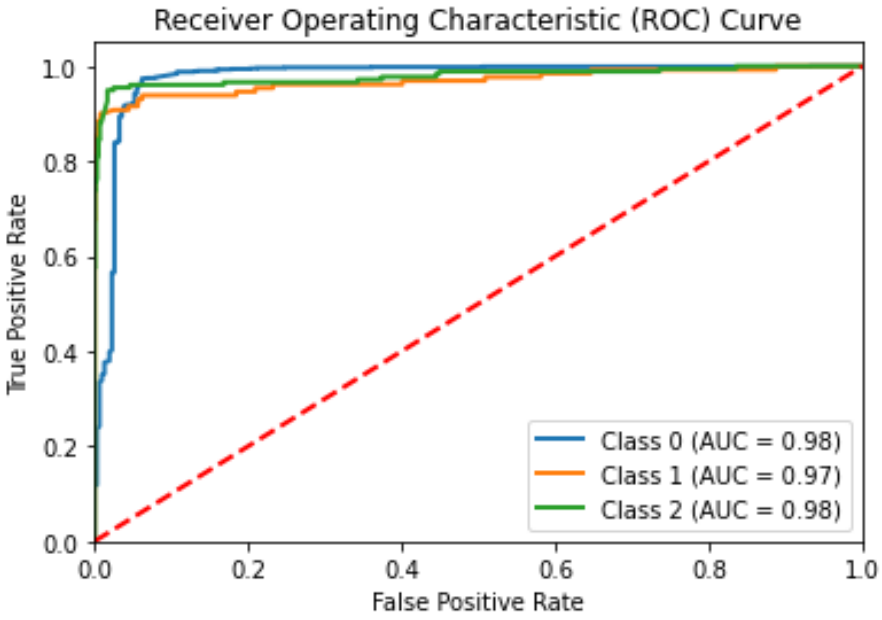


Figure A.8

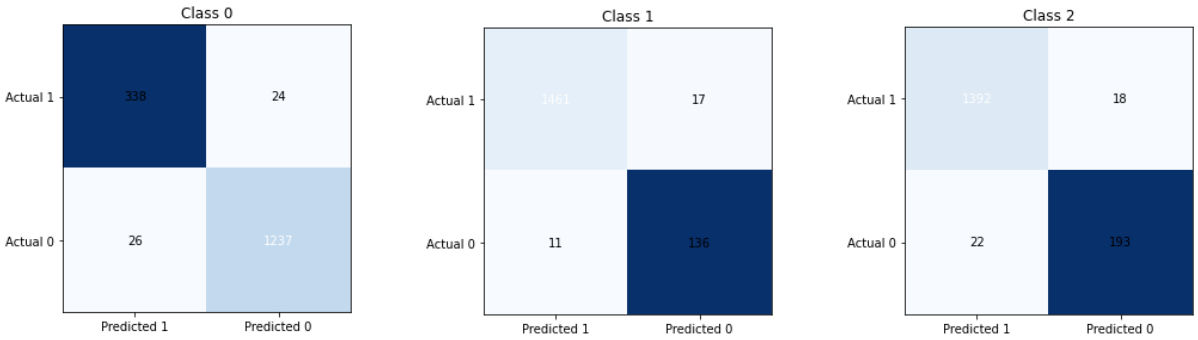
A.1.1.5 Evaluation Results on Day 1, 2, 3, 4 and 5 Measurements

Below are the calculated testing metric results for the Heart Rate model for measurements taken on Day 1, Day 2, Day 3, Day 4, and Day 5:

Table A.5: Testing Metrics

Metric	Value
Accuracy	0.9636923076923077
Categorical cross-entropy loss	0.2036896813926578
Precision	0.96
Recall	0.96
F1 Score	0.96

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.9: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Heart Rate Frequency Observation based on Day 1 Measurements

ROC Curve:

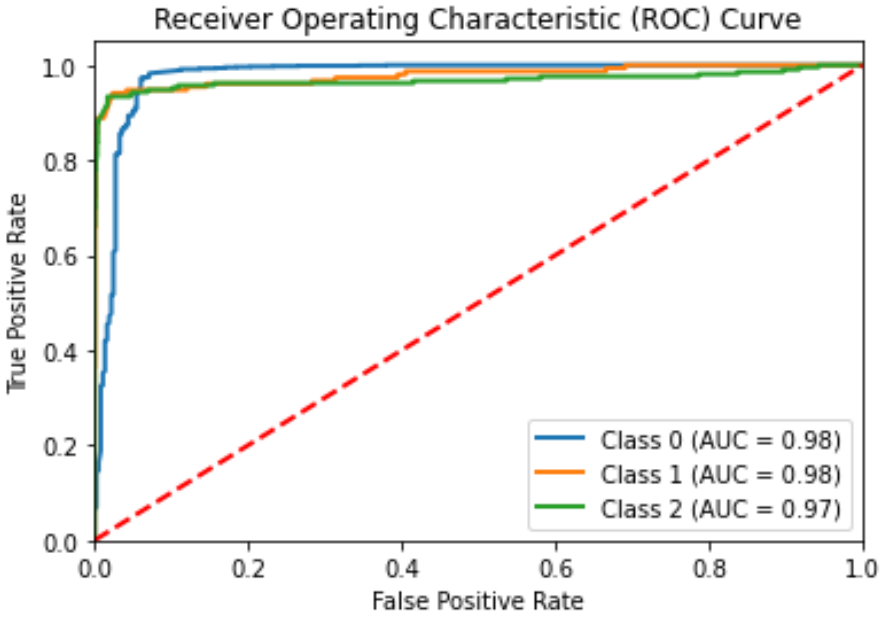


Figure A.10

A.1.1.6 Progression Of Heart Rate Metrics

In the below graph, you will be able to see how the metric values progress over 6 days according to various metrics:

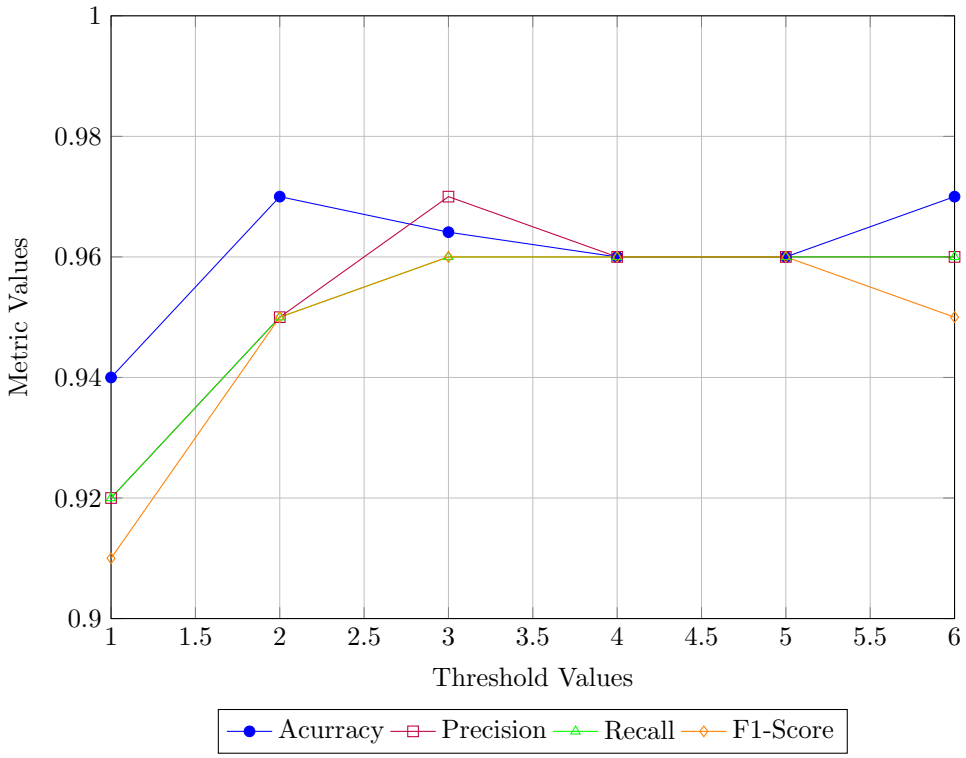


Figure A.11: Progression of Metrics over Different Threshold Values

As you can see from Figure A.63, the metrics performed best on Day 6. However, the metrics from Day 1 to the end are all above 90% and provide a good enough prognosis. In the below graph you will be able to see the progression of loss:

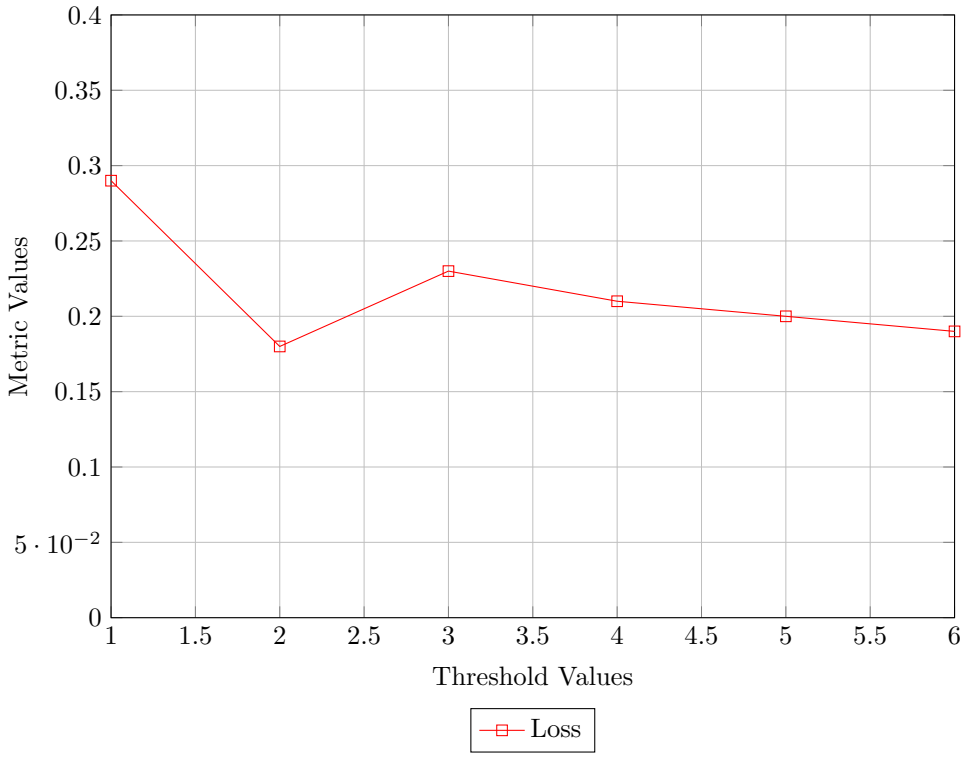


Figure A.12: Progression of Loss over Different Threshold Values

Loss values are ranging between 0.3 and 0.15. The best loss was achieved on Day 2. After a spike on

Day 3, the loss values continued to progressively become lower. In the below graph you will be able to see how the ROC curve for Class0, Class1, and Class2 progress over time:

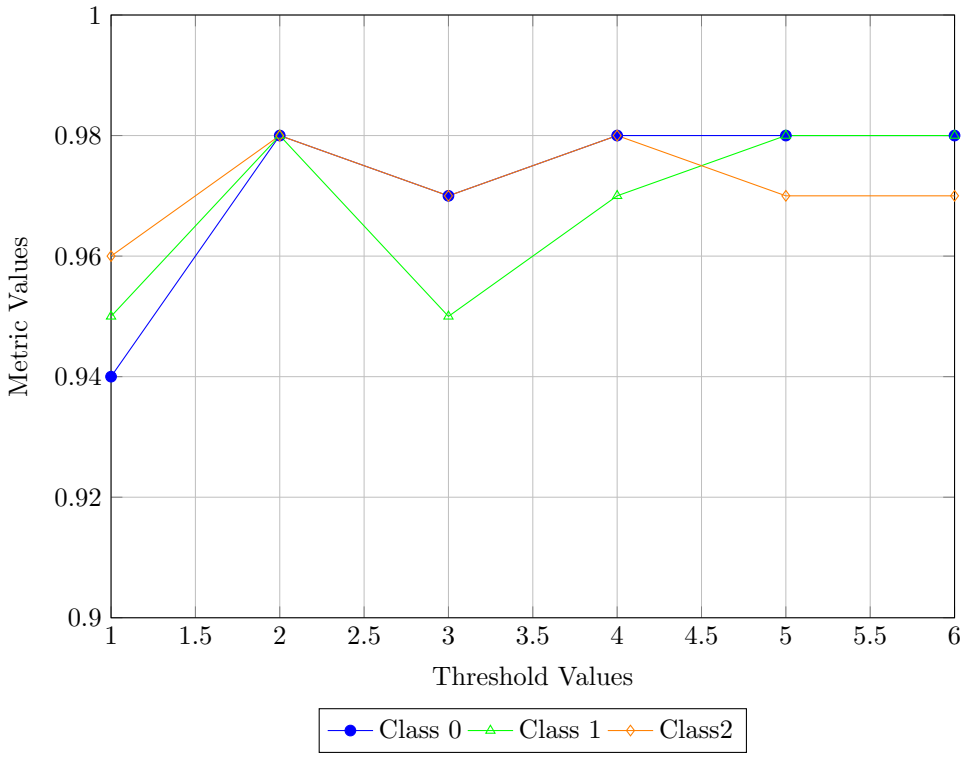


Figure A.13: Progression of AUC Score For Class0,Class1 and Class2

The AUC Scores are also more or less the same with a drop on Day 3 and stabilization again on Day 4.

A.1.2 Oxygen Saturation

Below you will be able to find measurements concerning the oxygen saturation on separate days.

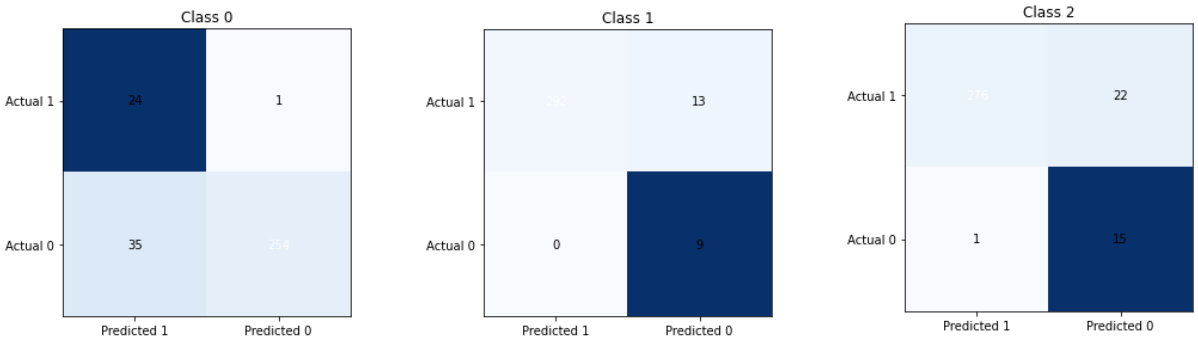
A.1.2.1 Evaluation Results on Day 1 Measurements

Below are the calculated testing metric results for the Oxygen Saturation model for measurements taken on Day 1 :

Table A.6: Testing Metrics

Metric	Value
Accuracy	0.8853503184713376
Categorical cross-entropy loss	0.3751741045391938
Precision	0.95
Recall	0.89
F1 Score	0.90

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.14: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Oxygen Saturation Observation based on Day 1 Measurements

ROC Curve:

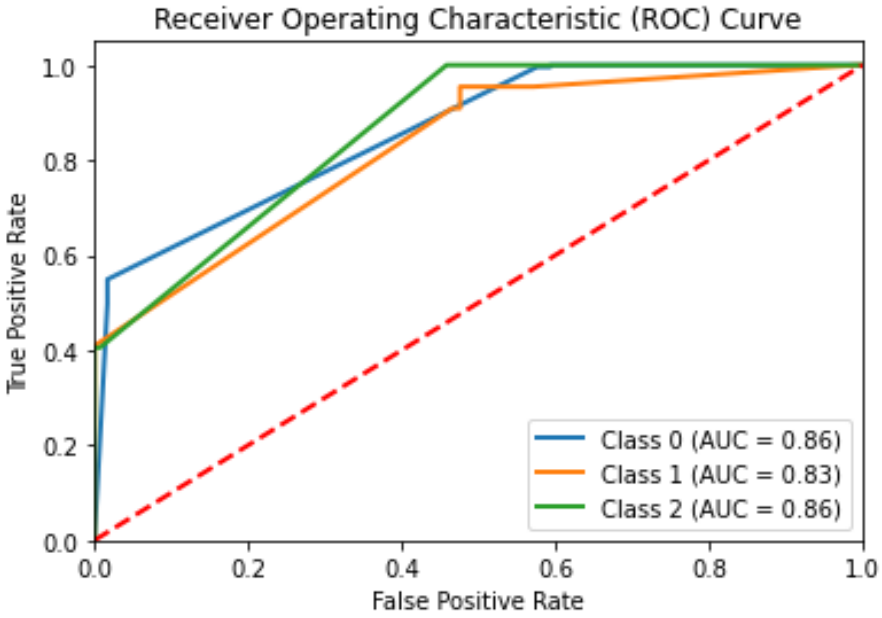


Figure A.15

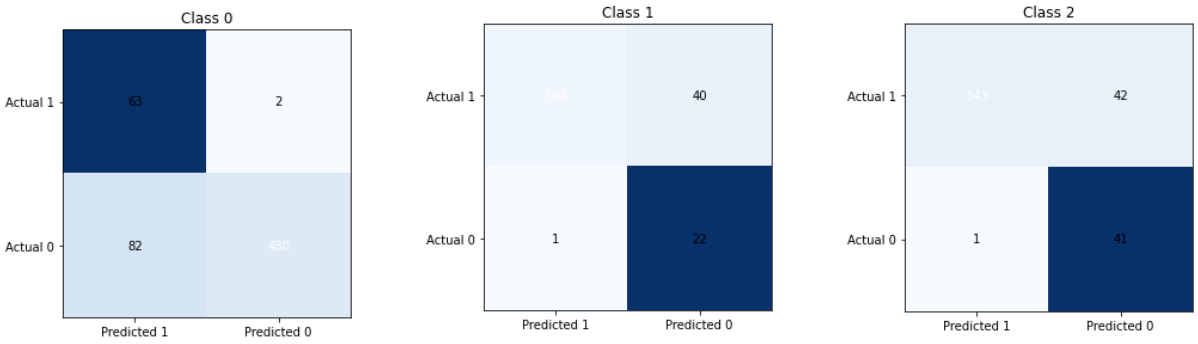
A.1.2.2 Evaluation Results on Day 1 and Day 2 Measurements

Below are the calculated testing metric results for the Oxygen Saturation model for measurements taken on Day 1 and Day 2 :

Table A.7: Testing Metrics

Metric	Value
Accuracy	0.8660287081339713
Categorical cross-entropy loss	0.442390907126446
Precision	0.94
Recall	0.87
F1 Score	0.89

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.16: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Oxygen Saturation Observation based on Day 1 and Day 2 Measurements

ROC Curve:

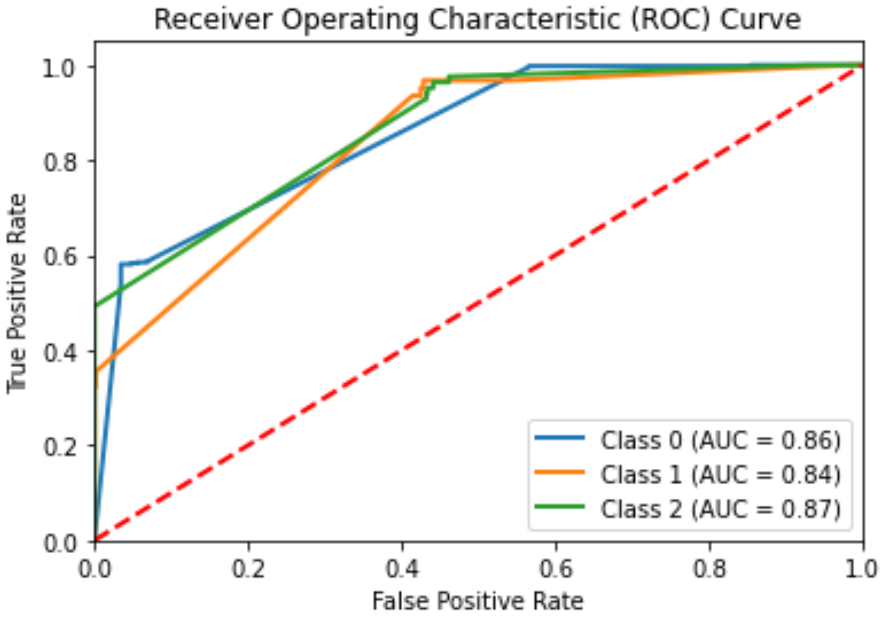


Figure A.17

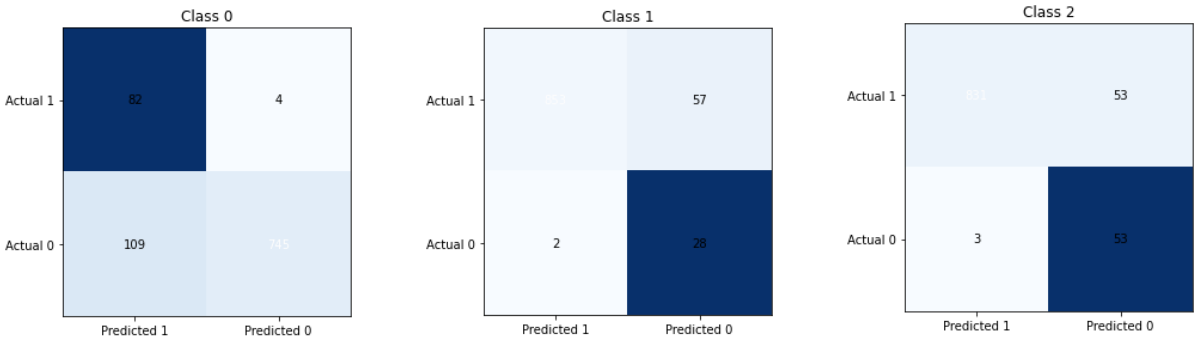
A.1.2.3 Evaluation Results on Day 1, Day 2, and Day 3 Measurements

Below are the calculated testing metric results for the Oxygen Saturation model for measurements taken on Day 1, Day 2and Day 3 :

Table A.8: Testing Metrics

Metric	Value
Accuracy	0.8765957446808511
Categorical cross-entropy loss	0.3988981335507593
Precision	0.94
Recall	0.88
F1 Score	0.89

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.18: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Oxygen Saturation Observation based on Day 1, Day 2, and Day 3 Measurements

ROC Curve:

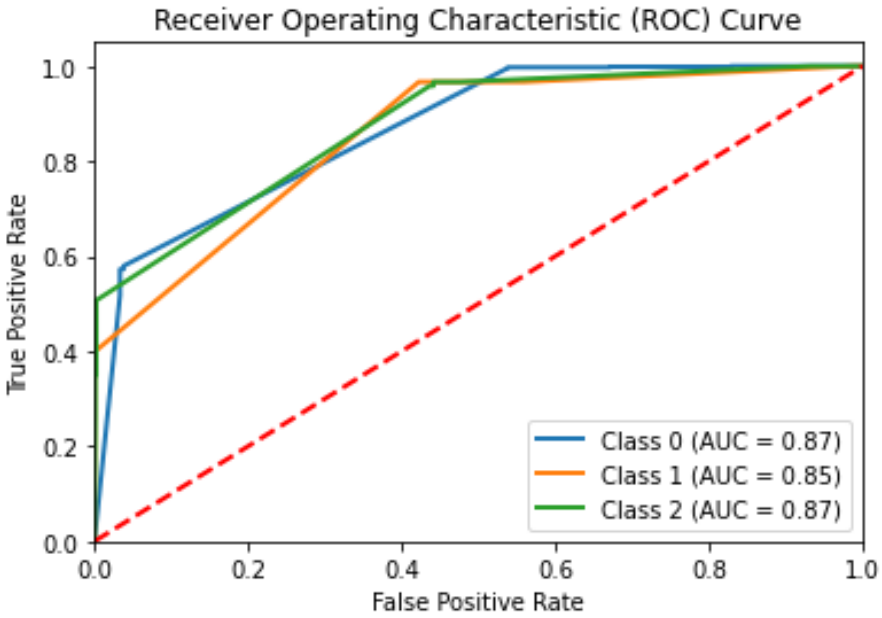


Figure A.19

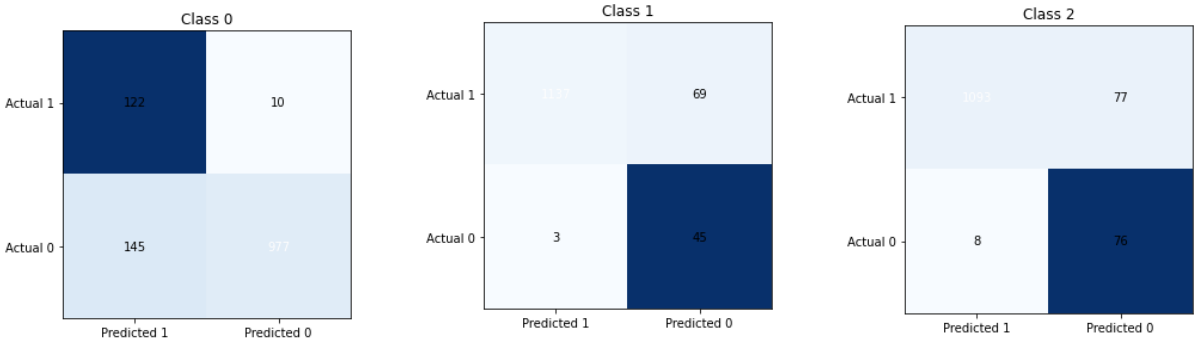
A.1.2.4 Evaluation Results on Day 1, Day 2, Day 3, and Day 4 Measurements

Below are the calculated testing metric results for the Oxygen Saturation model for measurements taken on Day 1, Day 2, Day 3, Day 4, and Day 5 :

Table A.9: Testing Metrics

Metric	Value
Accuracy	0.8787878787878788
Categorical cross-entropy loss	0.38436080157820124
Precision	0.94
Recall	0.88
F1 Score	0.89

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.20: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for OxygenSaturation Observation based on Day 1, Day 2, Day 3, and Day 4 Measurements

ROC Curve:

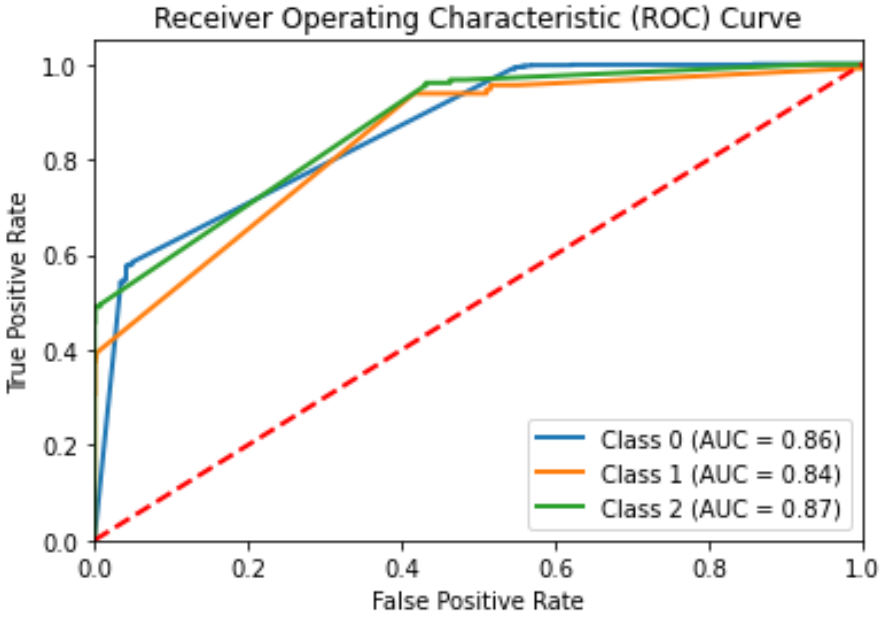


Figure A.21

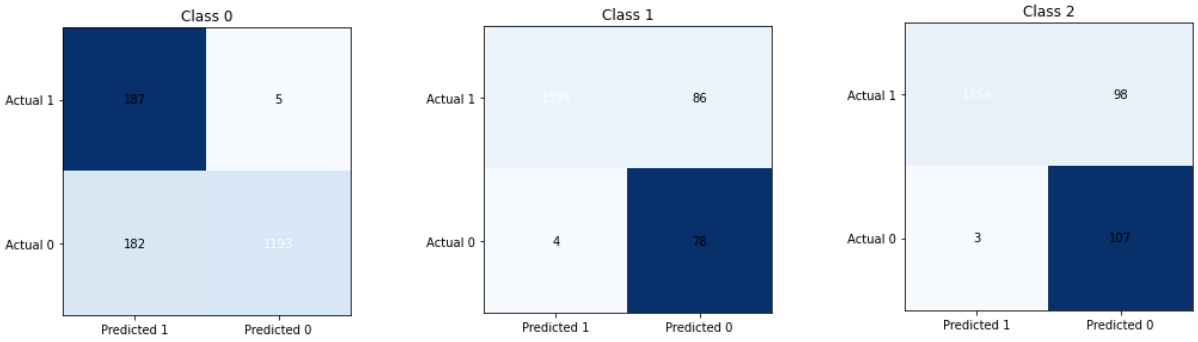
A.1.2.5 Evaluation Results on Day 1, Day 2, Day 3, Day 4, and Day 5 Measurements

Below are the calculated testing metric results for the Oxygen Saturation model for measurements taken on Day 1, Day 2, Day 3, Day 4, and Day 5 :

Table A.10: Testing Metrics

Metric	Value
Accuracy	0.8793873643905552
Categorical cross-entropy loss	0.3793660970814235
Precision	0.94
Recall	0.88
F1 Score	0.89

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.22: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Oxygen Saturation Observation based on Day 1, Day 2, Day 3, Day 4, and Day 5 Measurements

ROC Curve:

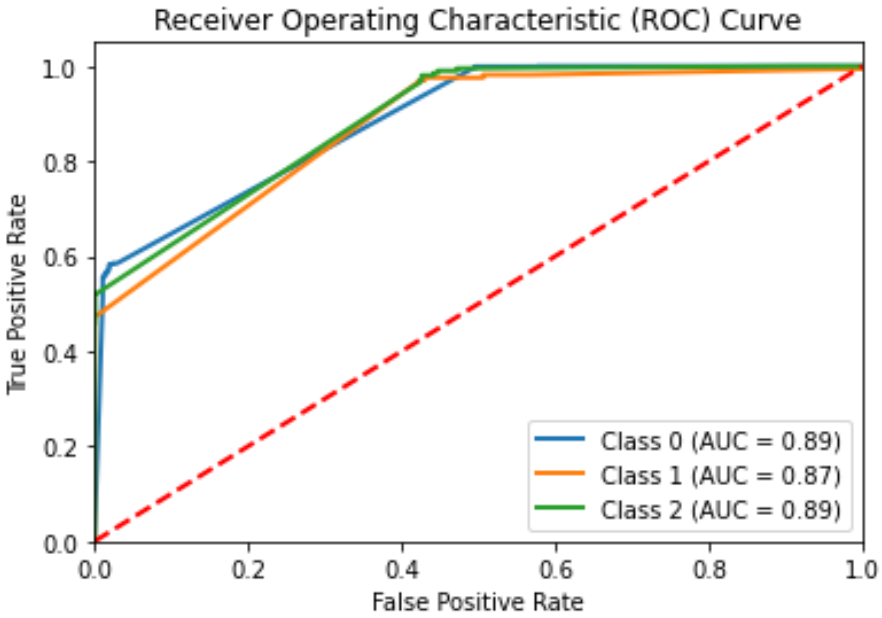


Figure A.23

A.1.2.6 Progression Of Metrics

In the below graph, you will be able to see how the metric values progress over 6 days according to various metrics:

The performance metrics for Oxygen saturation remained almost constant throughout the whole period. In the below graph, you will be able to see the progression of loss:

The same scenario we observed for accuracy, precision, recall, and f-1 score, applies to loss as well. Other than a small increase on Day 2, the loss values are pretty much stable. In the below graph you will be able to see how the ROC curve for Class0, Class1, and Class2 progress over time:

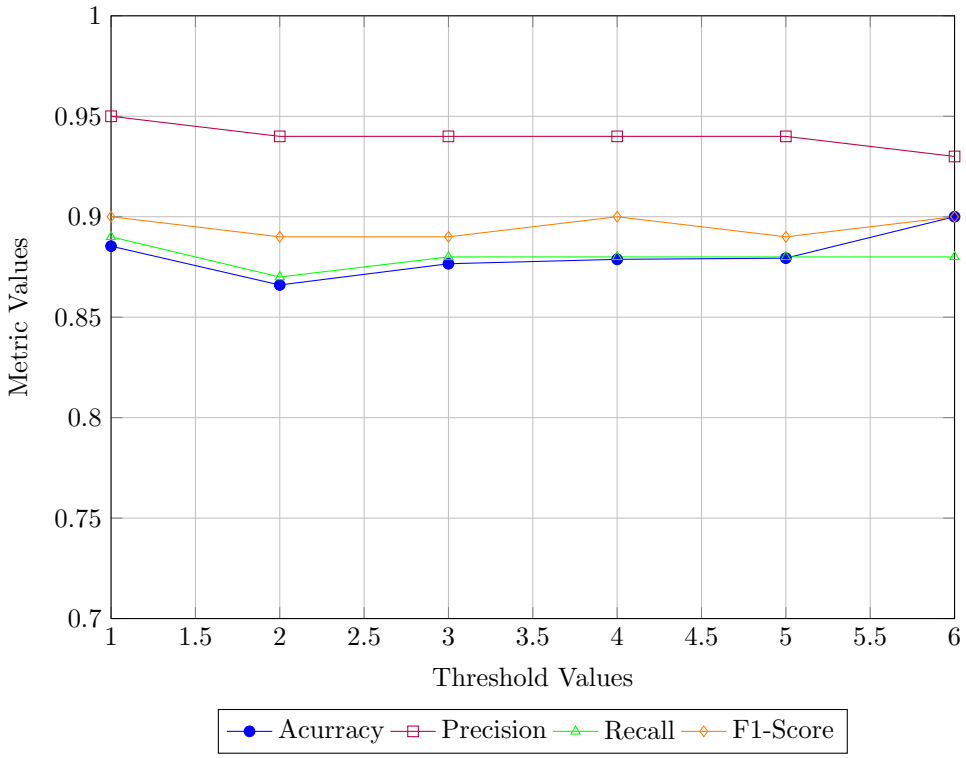


Figure A.24: Progression of Metrics over Different Threshold Values

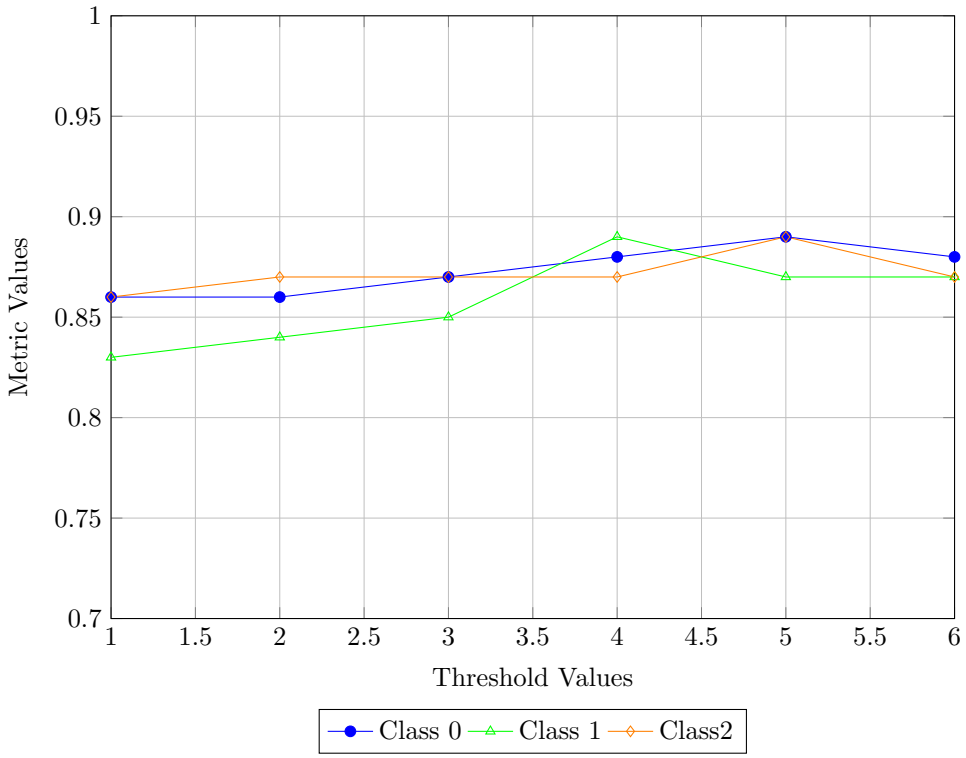


Figure A.26: Progression of AUC Score For Class0,Class1 and Class2

The AUC Scores started from approximately 0.80 and progressively reached 0.90 over 6 days. The rate of increase is constant except for Class 1, where a 0.5 jump is observed on Day 4, but again stabilized on Day 5.

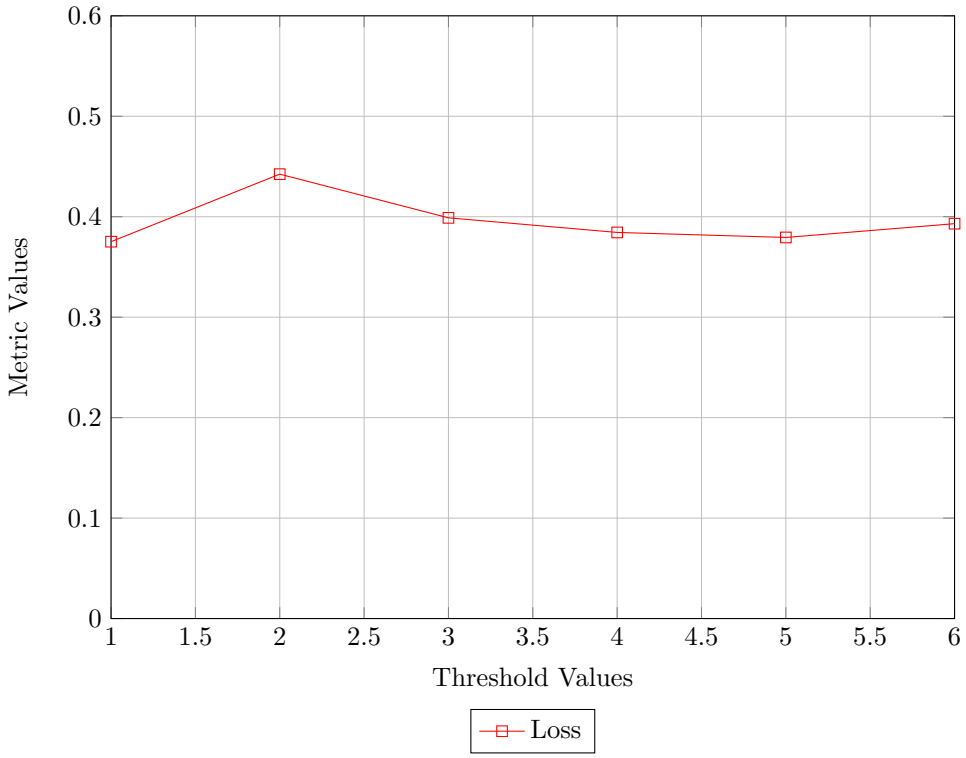


Figure A.25: Progression of Loss over Different Threshold Values

A.1.3 Blood Pressure

Below you will be able to find measurements concerning the blood pressure on separate days.

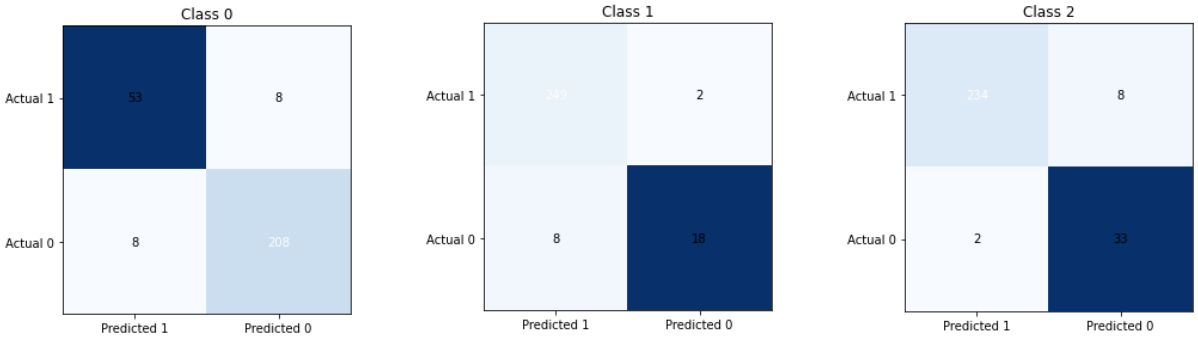
A.1.3.1 Evaluation Results on Day 1 Measurements

Below are the calculated testing metric results for the Blood Pressure model for measurements taken on Day 1 :

Table A.11: Testing Metrics

Metric	Value
Accuracy	0.9350180505415162
Categorical cross-entropy loss	0.29492478769896224
Precision	0.94
Recall	0.94
F1 Score	0.93

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.27: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Blood Pressure Observation based on Day 1 Measurements

ROC Curve:

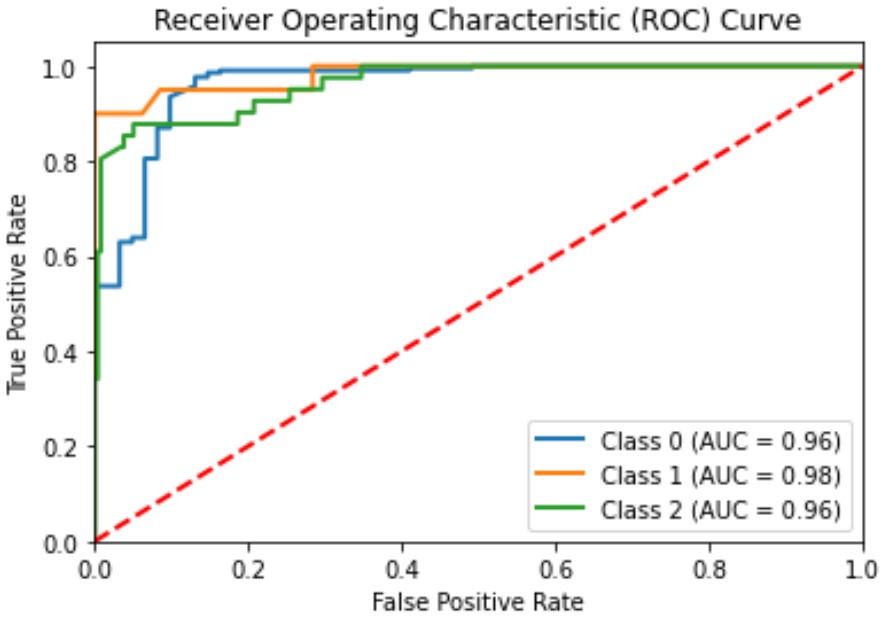


Figure A.28

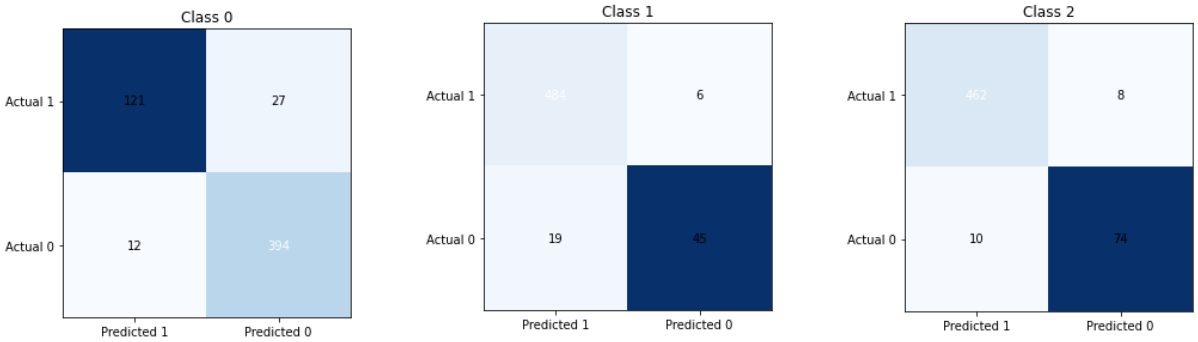
A.1.3.2 Evaluation Results on Day 1 and Day 2 Measurements

Below are the calculated testing metric results for the Blood Pressure model for measurements taken on Day 1 and Day 2:

Table A.12: Testing Metrics

Metric	Value
Accuracy	0.9259927797833934
Categorical cross-entropy loss	0.3219503200849507
Precision	0.99
Recall	0.90
F1 Score	0.95

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.29: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Blood Pressure Observation based on Day 1 and Day 2 Measurements

ROC Curve:

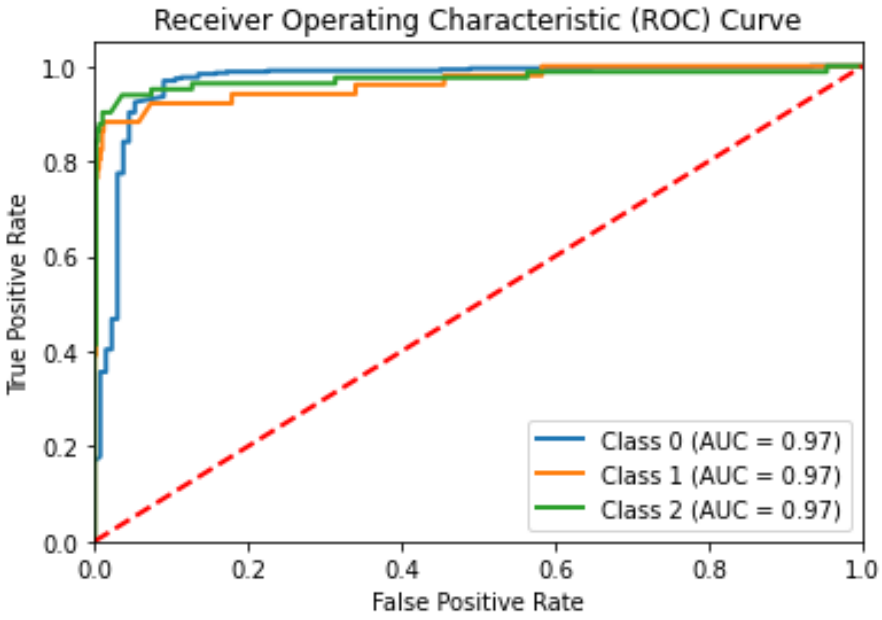


Figure A.30

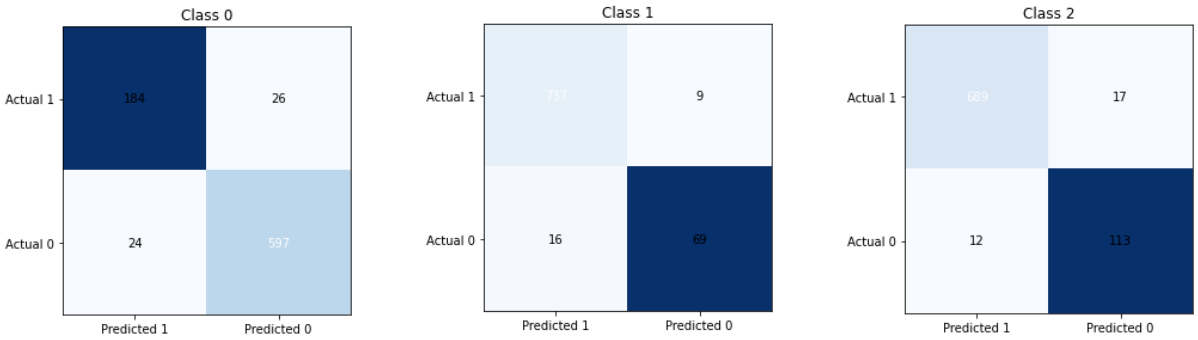
A.1.3.3 Evaluation Results on Day 1, Day 2, and Day 3 Measurements

Below are the calculated testing metric results for the Blood Pressure model for measurements taken on Day 1, Day 2, and Day 3:

Table A.13: Testing Metrics

Metric	Value
Accuracy	0.937424789410349
Categorical cross-entropy loss	0.315287687230861
Precision	0.94
Recall	0.94
F1 Score	0.94

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.31: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Blood Pressure Observation based on Day 1, Day 2, and Day 3 Measurements

ROC Curve:

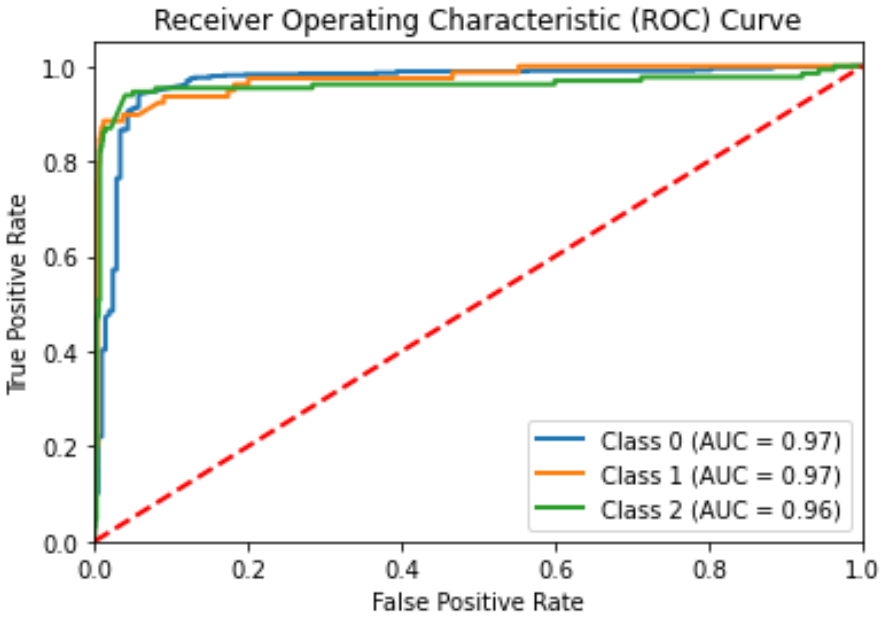


Figure A.32

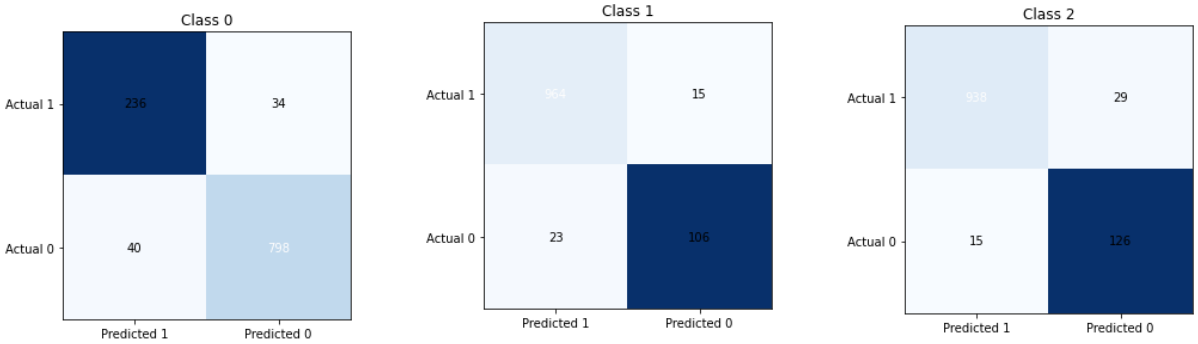
A.1.3.4 Evaluation Results on Day 1, Day 2, Day 3, and Day 4 Measurements

Below are the calculated testing metric results for the Blood Pressure model for measurements taken on Day 1, Day 2, Day 3, and Day 4:

Table A.14: Testing Metrics

Metric	Value
Accuracy	0.9296028880866426
Categorical cross-entropy loss	0.32192058750668245
Precision	0.93
Recall	0.93
F1 Score	0.93

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.33: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Blood Pressure Observation based on Day 1, Day 2, Day 3, and Day 4 Measurements

ROC Curve:

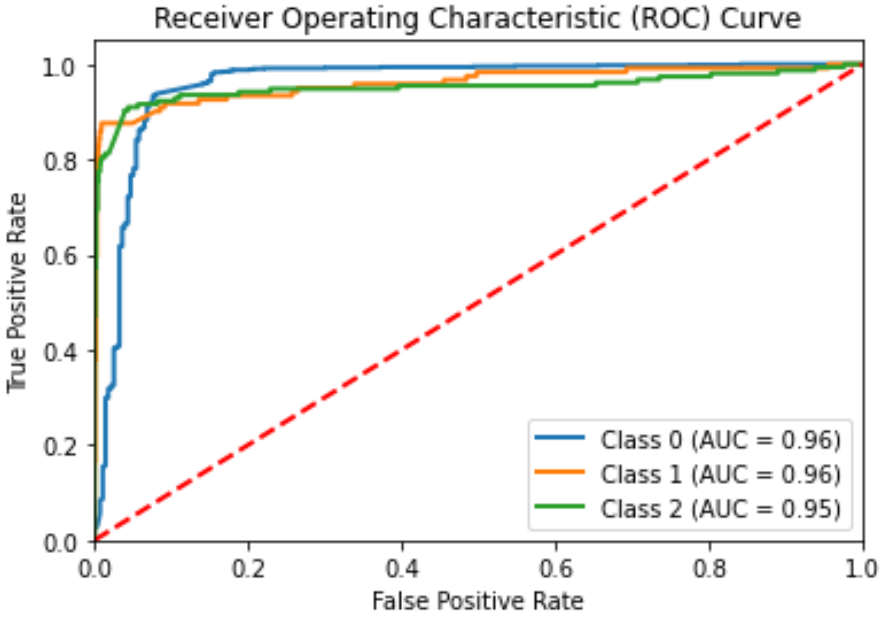


Figure A.34

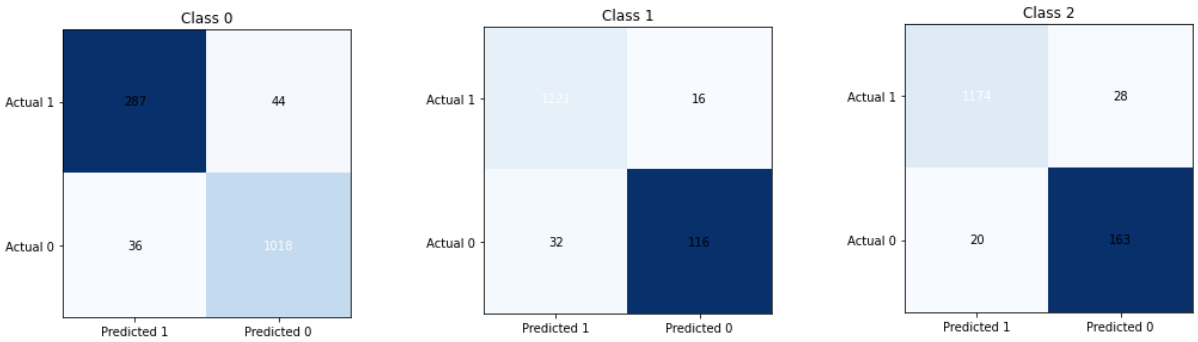
A.1.3.5 Evaluation Results on Day 1, Day 2, Day 3, Day 4, and Day 5 Measurements

Below are the calculated testing metric results for the Blood Pressure model for measurements taken on Day 1, Day 2, Day 3, Day 4, and Day 5:

Table A.15: Testing Metrics

Metric	Value
Accuracy	0.9364620938628159
Categorical cross-entropy loss	0.3103668821072699
Precision	0.94
Recall	0.94
F1 Score	0.94

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.35: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Blood Pressure Observation based on Day 1, Day 2, Day 3, Day 4, and Day 5 Measurements

ROC Curve:

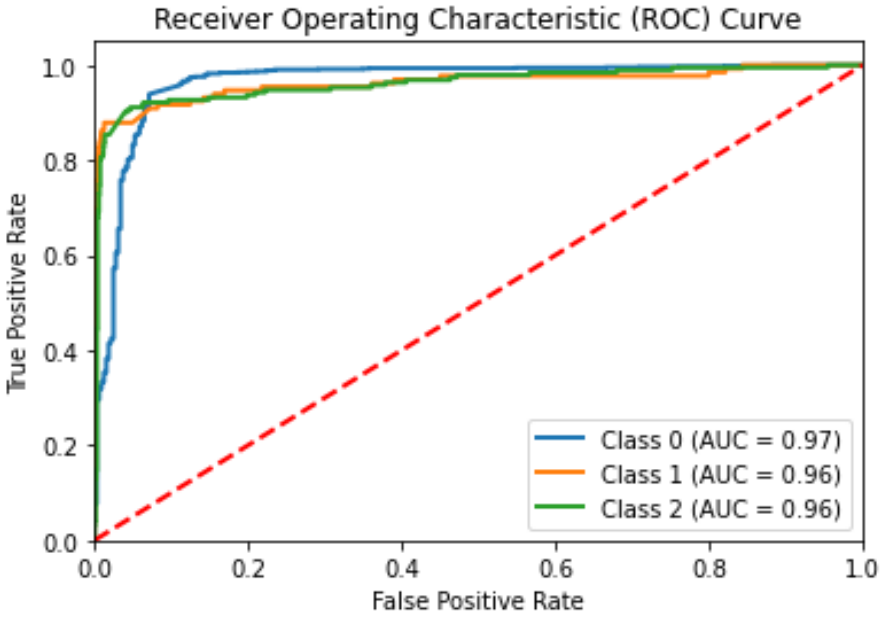


Figure A.36

A.1.3.6 Progression Of Metrics

In the below graph, you will be able to see how the metric values progress over 6 days according to various metrics:

The performance metrics for blood pressure moved in the range [0.90, 0.95]. It peaked around Day 6 and passed 0.95 but overall the results are pretty much stable over 6 day period. In the below graph you will be able to see the progression of loss:

The loss value was constant over the 6 days. In the below graph you will be able to see how the ROC curve for Class0, Class1, and Class2 progress over time:

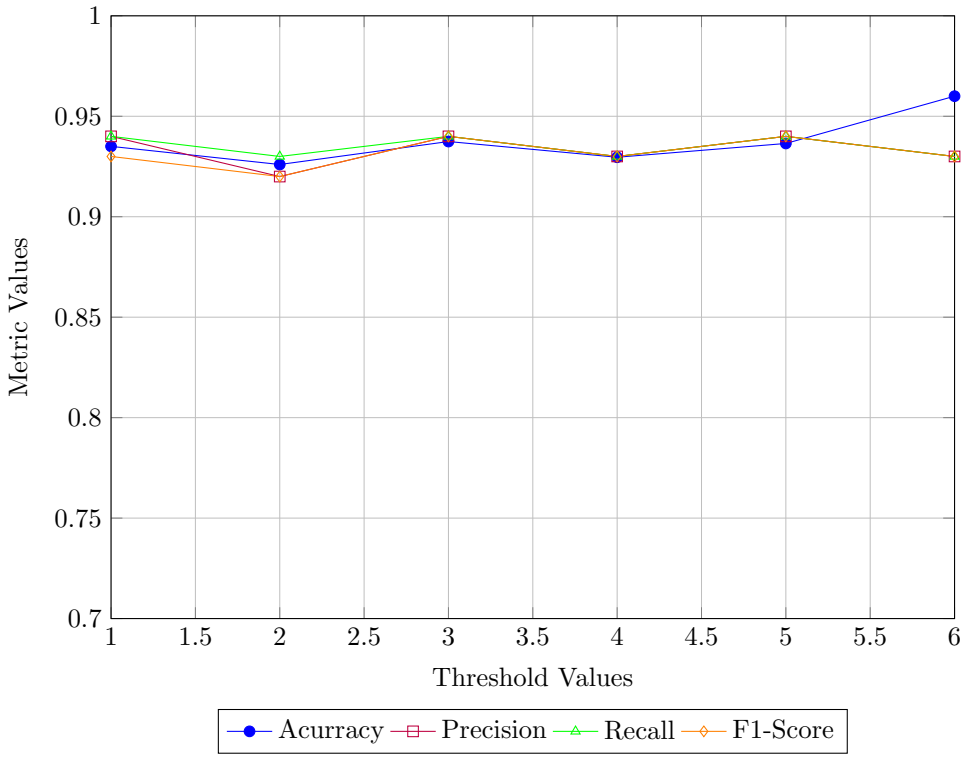


Figure A.37: Progression of Metrics over Different Threshold Values

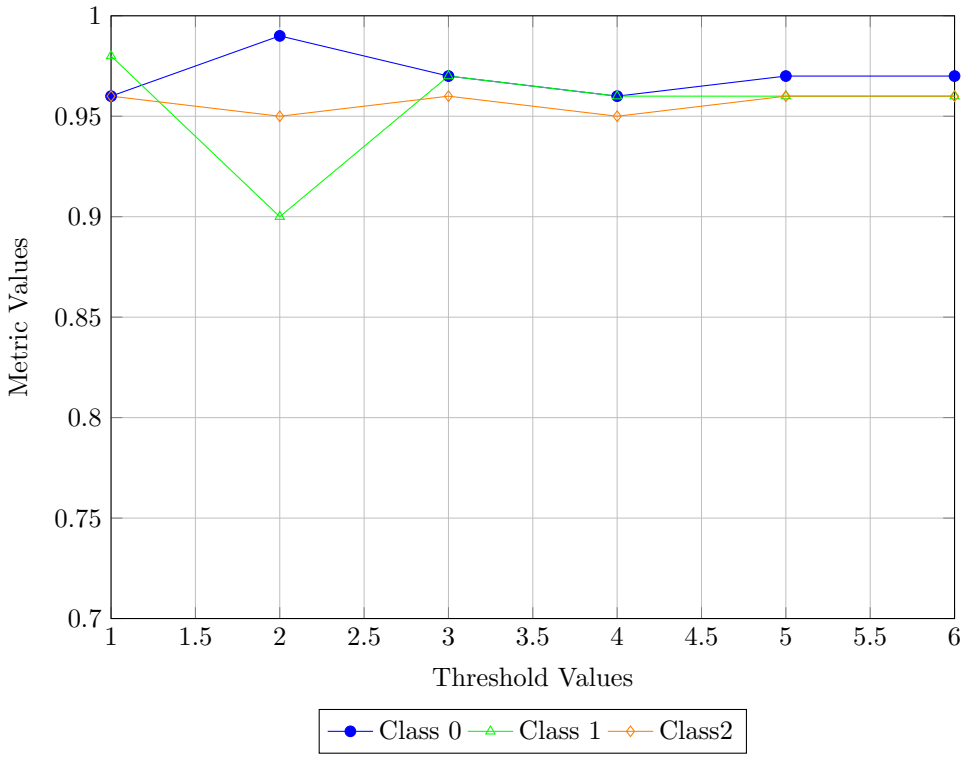


Figure A.39: Progression of AUC Score For Class0,Class1 and Class2

The AUC scores were again constant except for the Day 2 measurements. On Day 2, the Class 1 measurements experienced a sudden drop, and Class 0 measurements experienced a sudden increase, while Class 2 measurements remained consistent. After Day 2, the metrics returned to the usual trend.

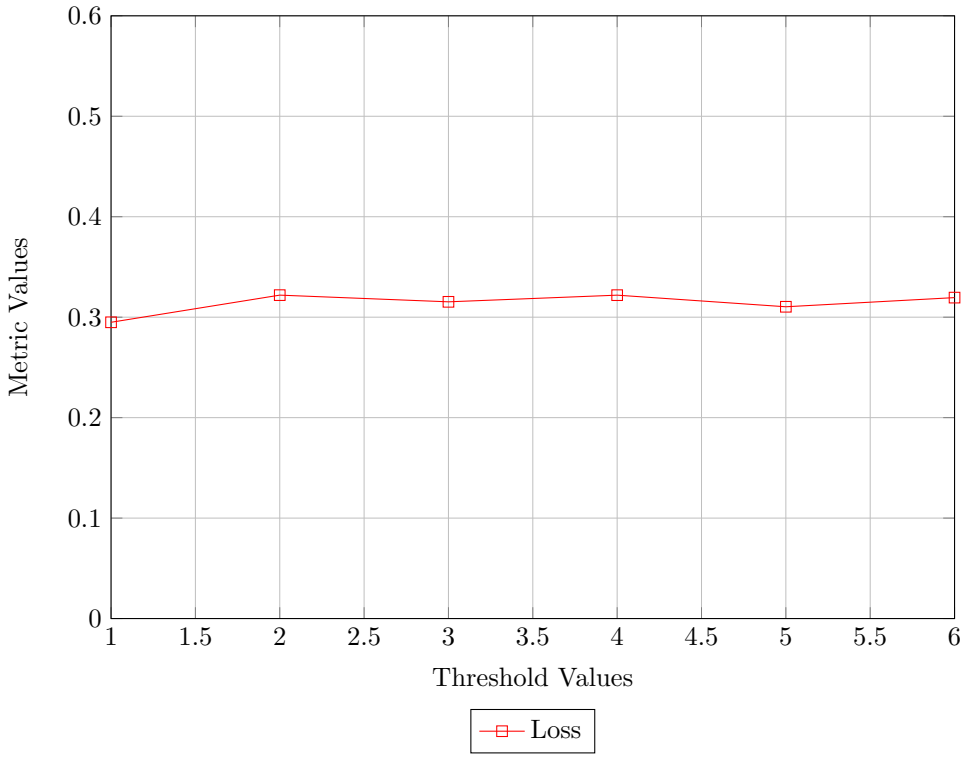


Figure A.38: Progression of Loss over Different Threshold Values

A.1.4 Temperature

Below you will be able to find measurements concerning the temperature on separate days.

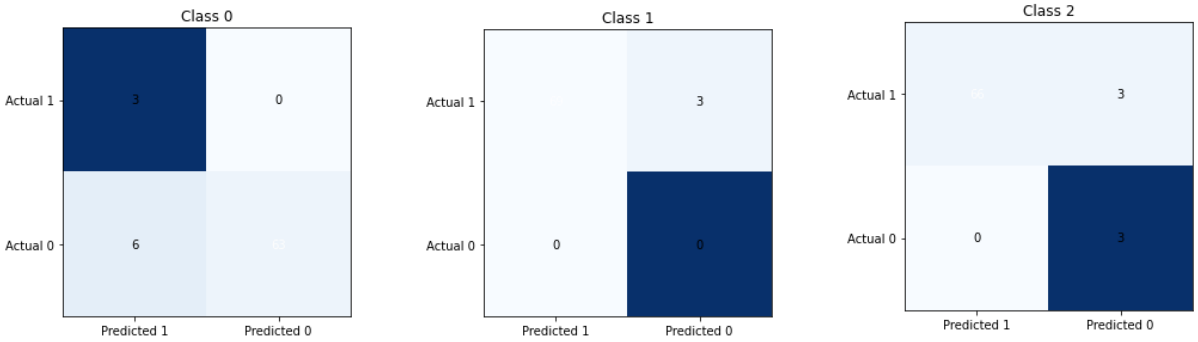
A.1.4.1 Evaluation Results on Day 1 Measurements

Below are the calculated testing metric results for the Temperature model for measurements taken on Day 1 :

Table A.16: Testing Metrics

Metric	Value
Accuracy	0.9166666666666666
Categorical cross-entropy loss	0.41507251105706344
Precision	0.97
Recall	0.92
F1 Score	0.94

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.40: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Temperature Observation based on Day 1 Measurements

ROC Curve:

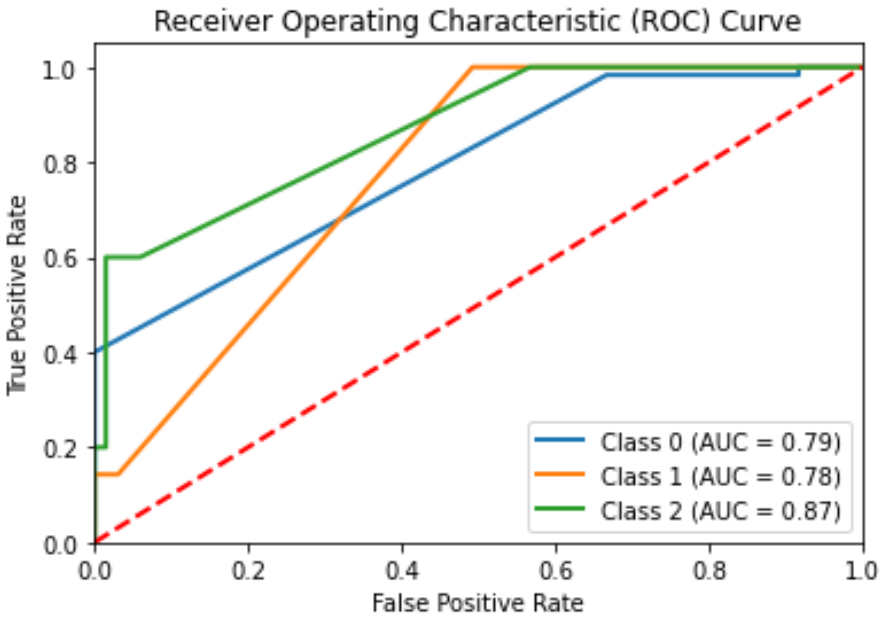


Figure A.41

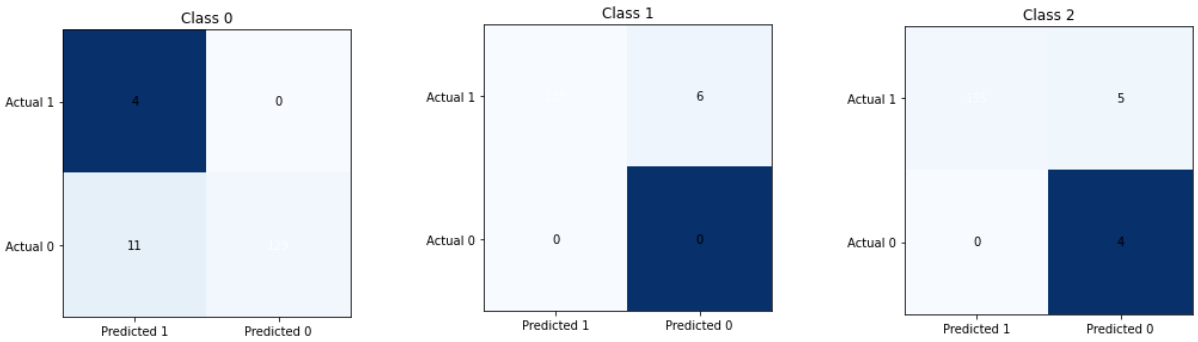
A.1.4.2 Evaluation Results on Day 1 and Day 2 Measurements

Below are the calculated testing metric results for the Temperature model for measurements taken on Day 1 and Day 2:

Table A.17: Testing Metrics

Metric	Value
Accuracy	0.9236111111111112
Categorical cross-entropy loss	0.42002248679837695
Precision	0.98
Recall	0.92
F1 Score	0.95

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.42: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Temperature Observation based on Day 1 and Day 2 Measurements

ROC Curve:

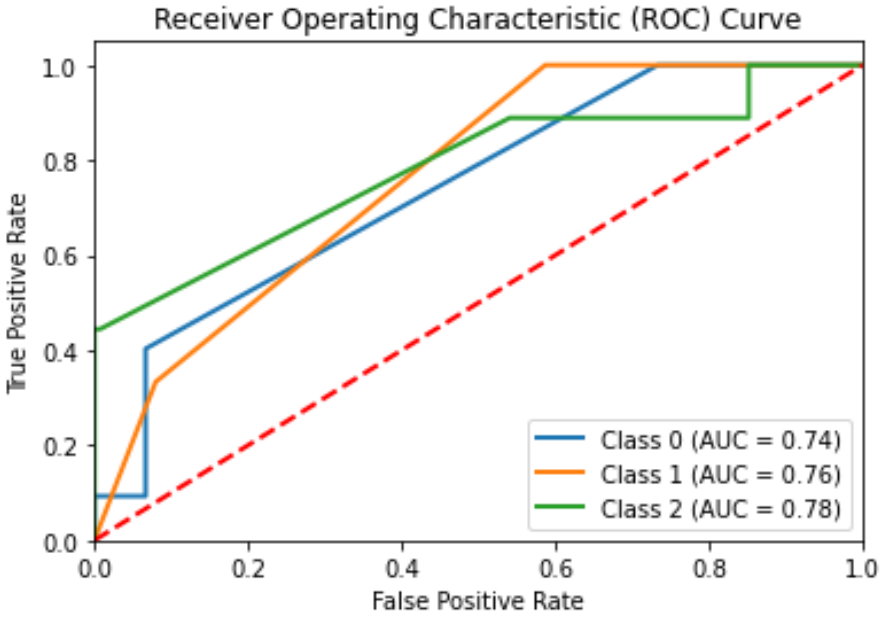


Figure A.43

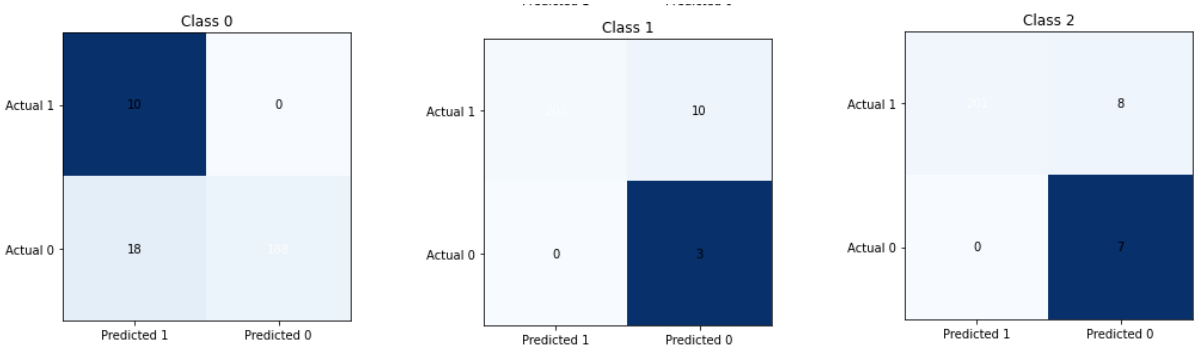
A.1.4.3 Evaluation Results on Day 1, Day 2, and Day 3 Measurements

Below are the calculated testing metric results for the Temperature model for measurements taken on Day 1, Day 2, and Day 3:

Table A.18: Testing Metrics

Metric	Value
Accuracy	0.9166666666666666
Categorical cross-entropy loss	0.43903515021747724
Precision	0.97
Recall	0.92
F1 Score	0.94

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.44: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Temperature Observation based on Day 1, Day 2, and Day 3 Measurements

ROC Curve:

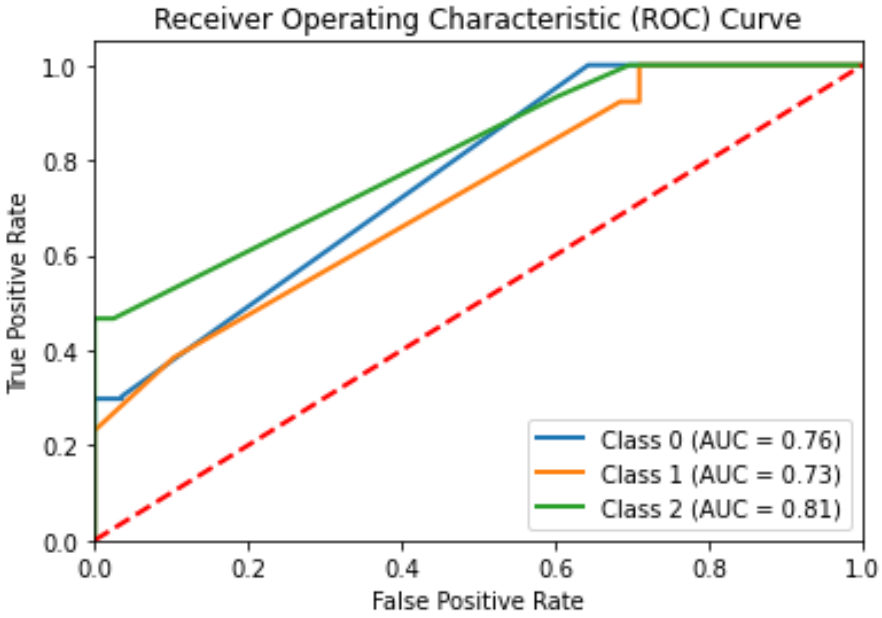


Figure A.45

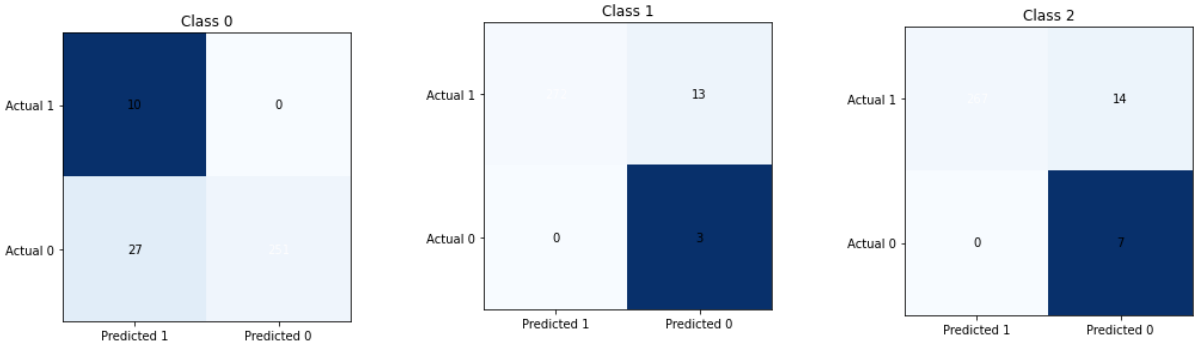
A.1.4.4 Evaluation Results on Day 1, Day 2, Day 3, and Day 4 Measurements

Below are the calculated testing metric results for the Temperature model for measurements taken on Day 1, Day 2, Day 3, and Day 4:

Table A.19: Testing Metrics

Metric	Value
Accuracy	0.90625
Categorical cross-entropy loss	0.4774239293082604
Precision	0.98
Recall	0.91
F1 Score	0.93

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.46: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Temperature Observation based on Day 1, Day 2, Day 3, and Day 4 Measurements

ROC Curve:

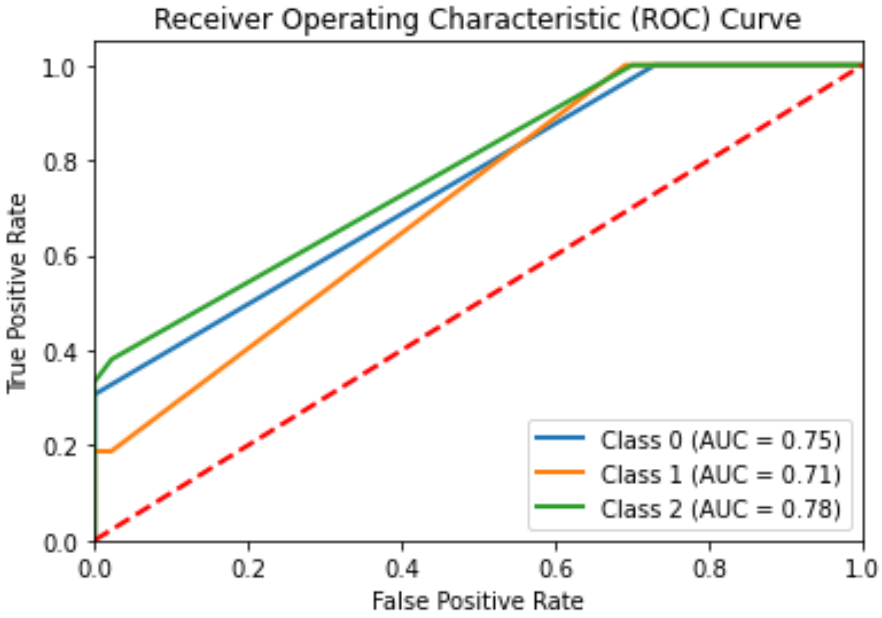


Figure A.47

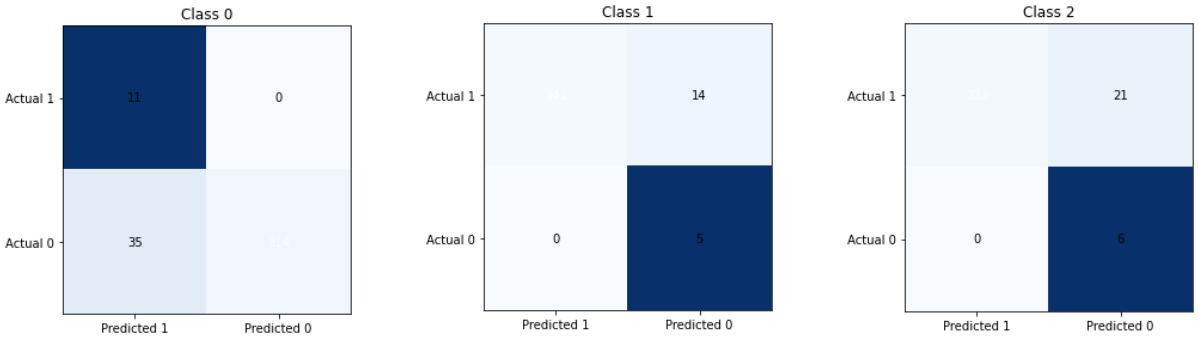
A.1.4.5 Evaluation Results on Day 1, Day 2, Day 3, Day 4, and Day 5 Measurements

Below are the calculated testing metric results for the Temperature model for measurements taken on Day 1, Day 2, Day 3, Day 4, and Day 5:

Table A.20: Testing Metrics

Metric	Value
Accuracy	0.9027777777777778
Categorical cross-entropy loss	0.4966367431580178
Precision	0.98
Recall	0.90
F1 Score	0.93

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.48: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Temperature Observation based on Day 1, Day 2, Day 3, Day 4, and Day 5 Measurements

ROC Curve:

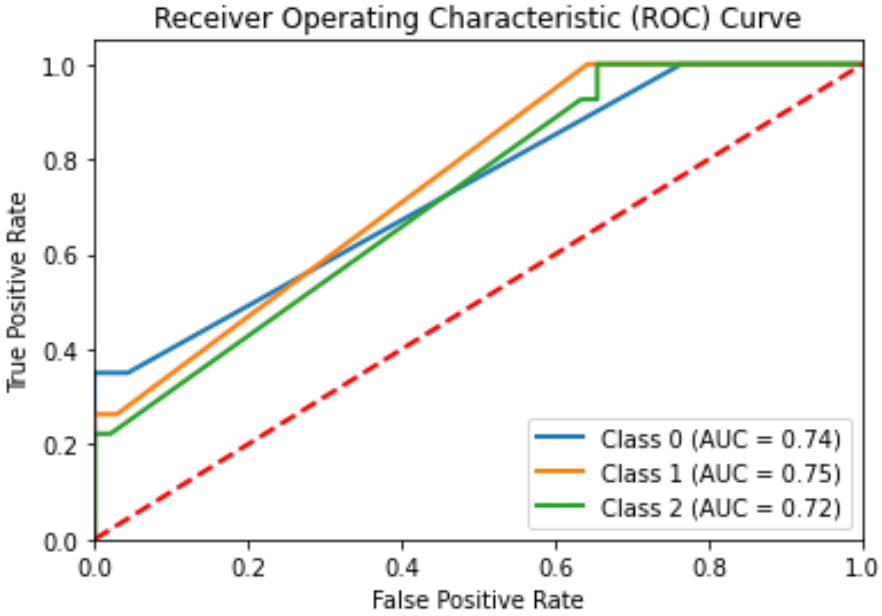


Figure A.49

A.1.4.6 Progression Of Metrics

In the below graph, you will be able to see how the metric values progress over 6 days according to various metrics:

The metrics for the temperature measurement remained in their original range of +- 0.1 over the whole period. In the below graph, you will be able to see the progression of loss:

The loss values increased over the 6 days. This is because the number of measurements we take into account increased. Although the model started to make more accurate choices, the margin of error in the misclassified samples also increased. In the below graph, you will be able to see how the ROC curve for Class0, Class1, and Class2 progress over time:

Similar to the loss metric, the AUC scores also show a performance decrement.

A.1.5 Respiration Frequency

Below you will be able to find measurements concerning the respiration frequency on separate days.

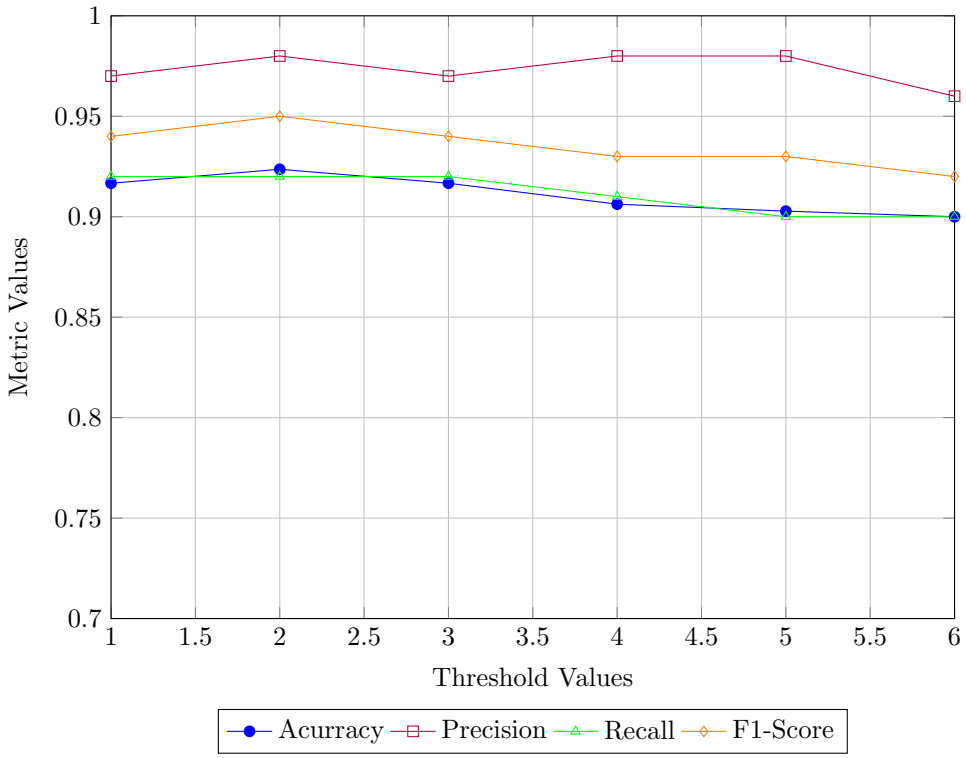


Figure A.50: Progression of Metrics over Different Threshold Values

A.1.5.1 Evaluation Results on Day 1 Measurements

Below are the calculated testing metric results for the Respiration Frequency model for measurements taken on Day 1 :

Table A.21: Testing Metrics

Metric	Value
Accuracy	0.9389067524115756
Categorical cross-entropy loss	0.2941545378990318
Precision	0.95
Recall	0.94
F1 Score	0.94

Confusion Matrix:

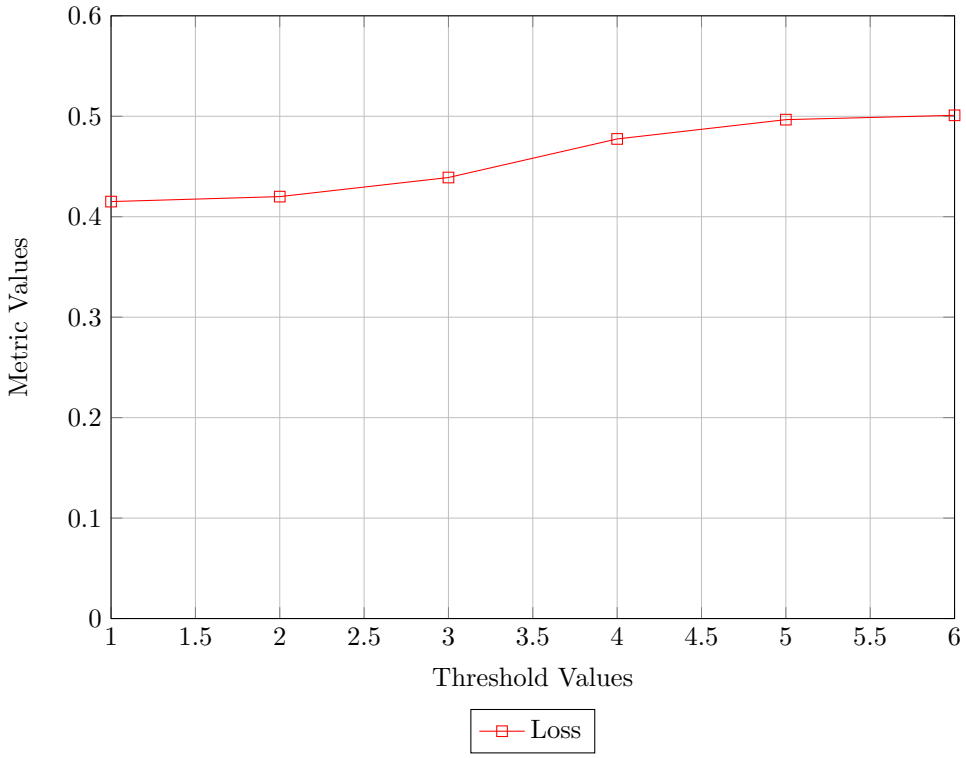
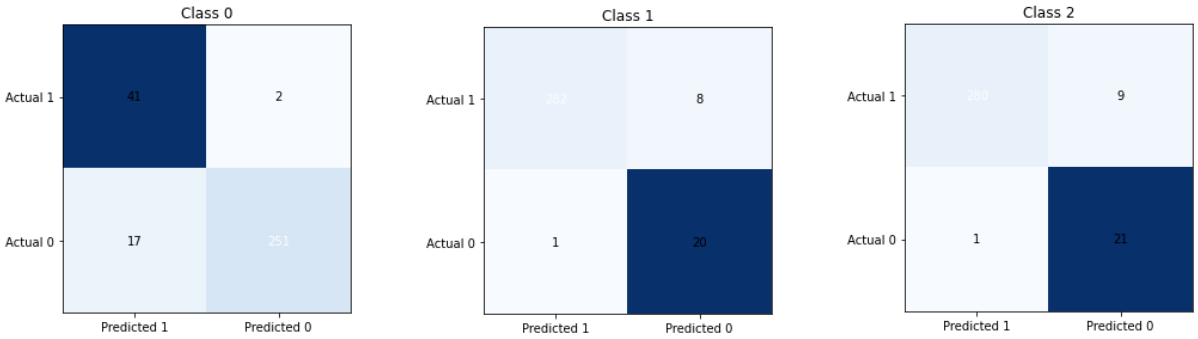


Figure A.51: Progression of Loss over Different Threshold Values



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.53: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Respiration Frequency Observation based on Day 1 Measurements

ROC Curve:

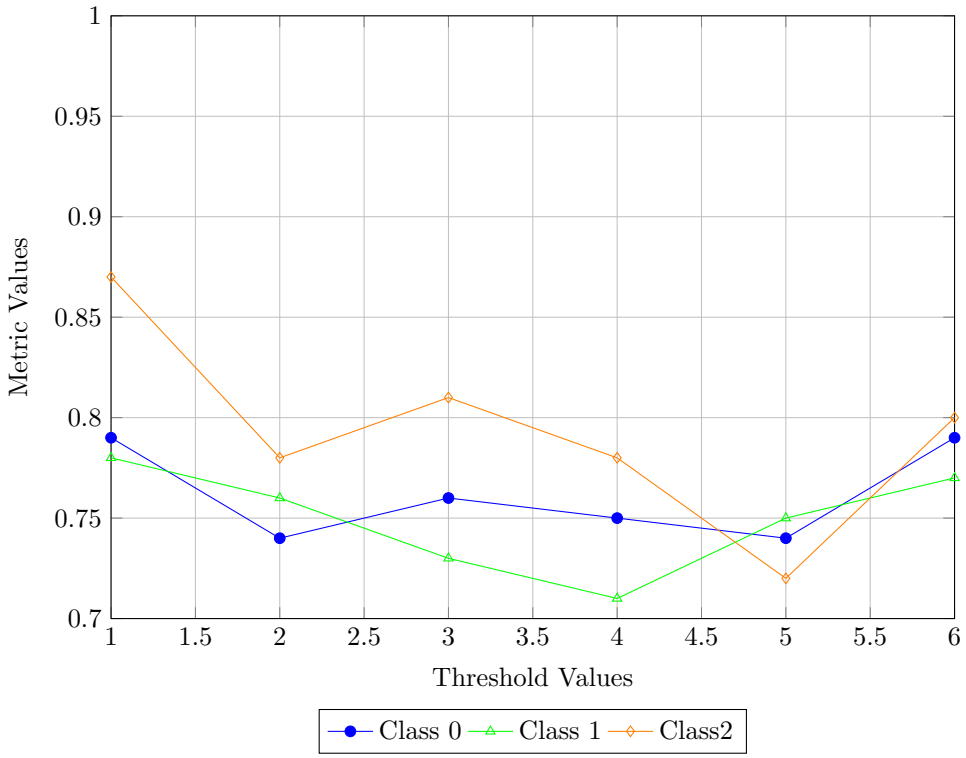


Figure A.52: Progression of AUC Score For Class0,Class1 and Class2

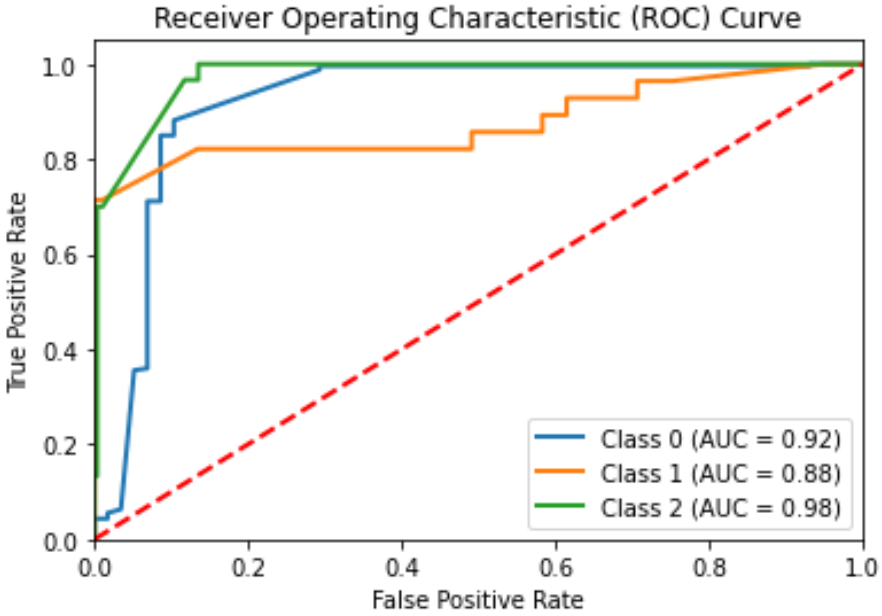


Figure A.54

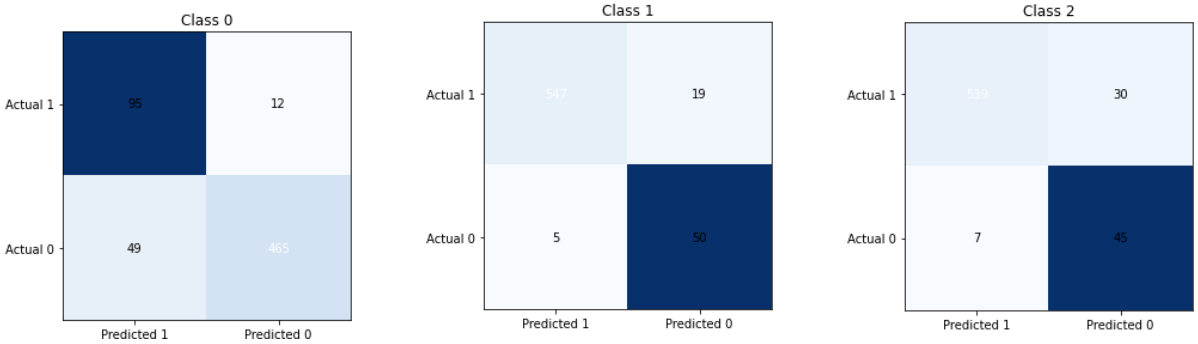
A.1.5.2 Evaluation Results on Day 1 and Day 2 Measurements

Below are the calculated testing metric results for the Respiration Frequency model for measurements taken on Day 1 and Day 2 :

Confusion Matrix:

Table A.22: Testing Metrics

Metric	Value
Accuracy	0.9017713365539453
Categorical cross-entropy loss	0.34345482048626136
Precision	0.92
Recall	0.90
F1 Score	0.91



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.55: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Respiration Frequency Observation based on Day 1 and Day 2 Measurements

ROC Curve:

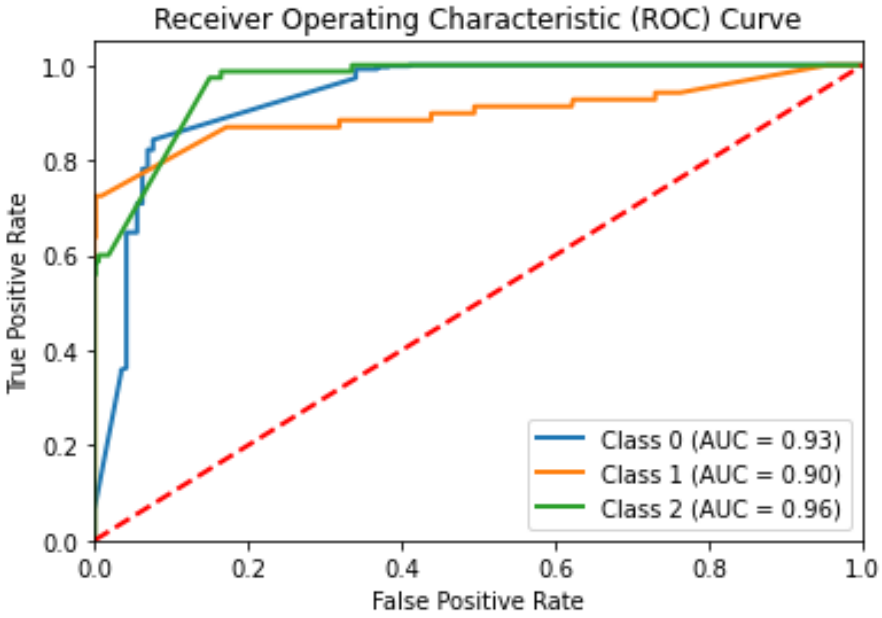


Figure A.56

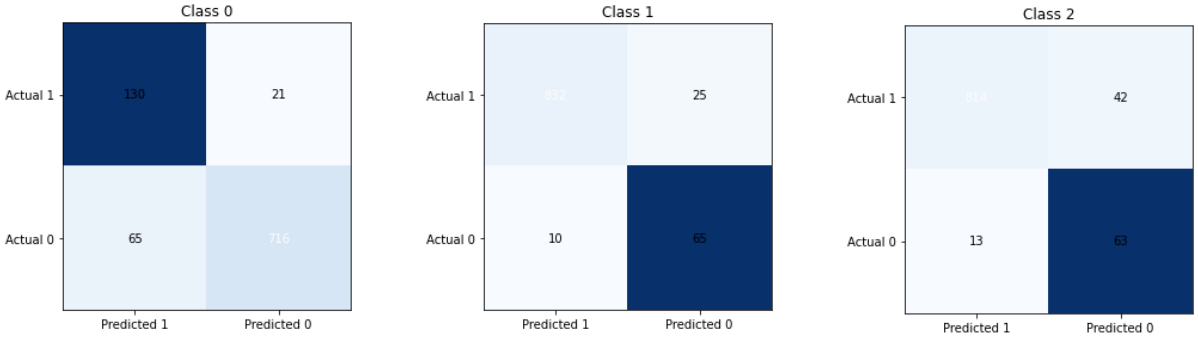
A.1.5.3 Evaluation Results on Day 1, Day 2, and Day 3 Measurements

Below are the calculated testing metric results for the Respiration Frequency model for measurements taken on Day 1, Day 2, and Day 3 :

Confusion Matrix:

Table A.23: Testing Metrics

Metric	Value
Accuracy	0.9055793991416309
Categorical cross-entropy loss	0.28494883076647914
Precision	0.92
Recall	0.91
F1 Score	0.91



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.57: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Respiration Frequency Observation based on Day 1, Day 2, and Day 3 Measurements

ROC Curve:

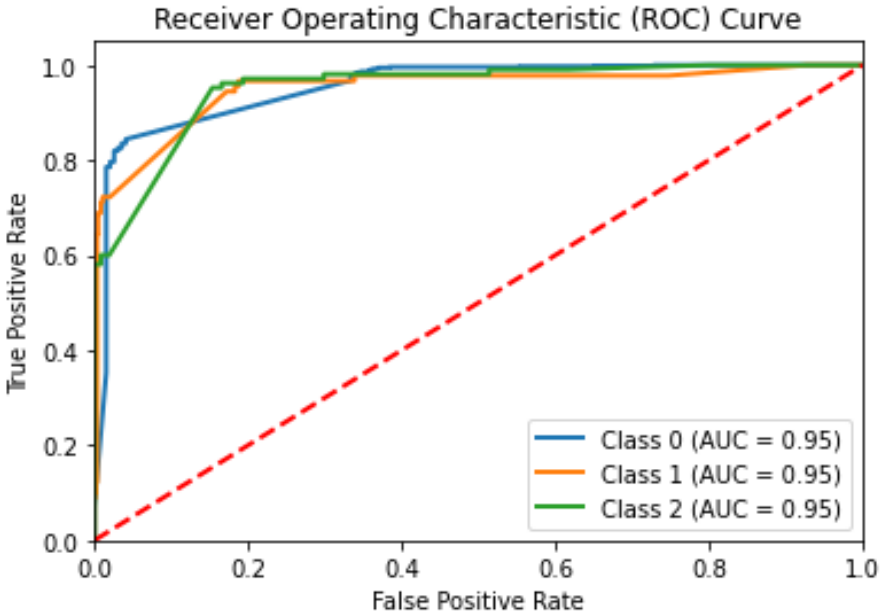


Figure A.58

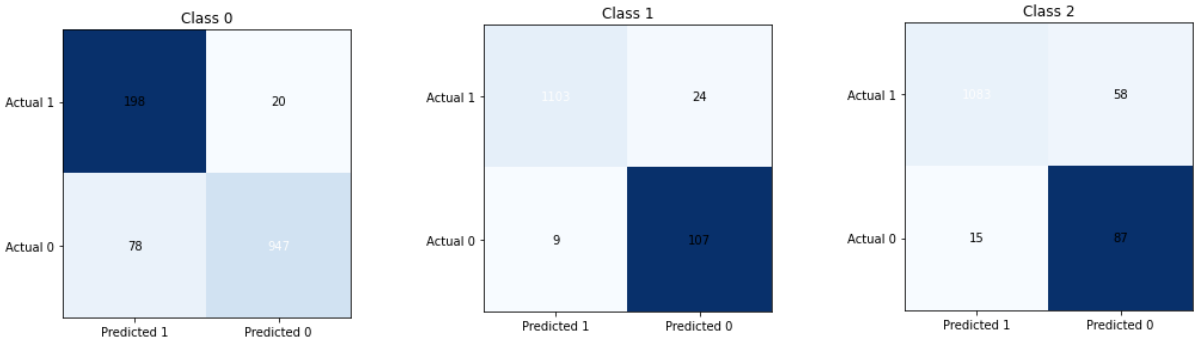
A.1.5.4 Evaluation Results on Day 1, Day 2, Day 3, and Day 4 Measurements

Below are the calculated testing metric results for the Respiration Frequency model for measurements taken on Day 1, Day 2, Day 3, and Day 4:

Confusion Matrix:

Table A.24: Testing Metrics

Metric	Value
Accuracy	0.917940466613033
Categorical cross-entropy loss	0.28912195702461396
Precision	0.93
Recall	0.92
F1 Score	0.92



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.59: Confusion Matrix results displaying statistics about Class0, Class1, and Class2 for Respiration Frequency Observation based on Day 1, Day 2, Day 3, and Day 4 Measurements

ROC Curve:

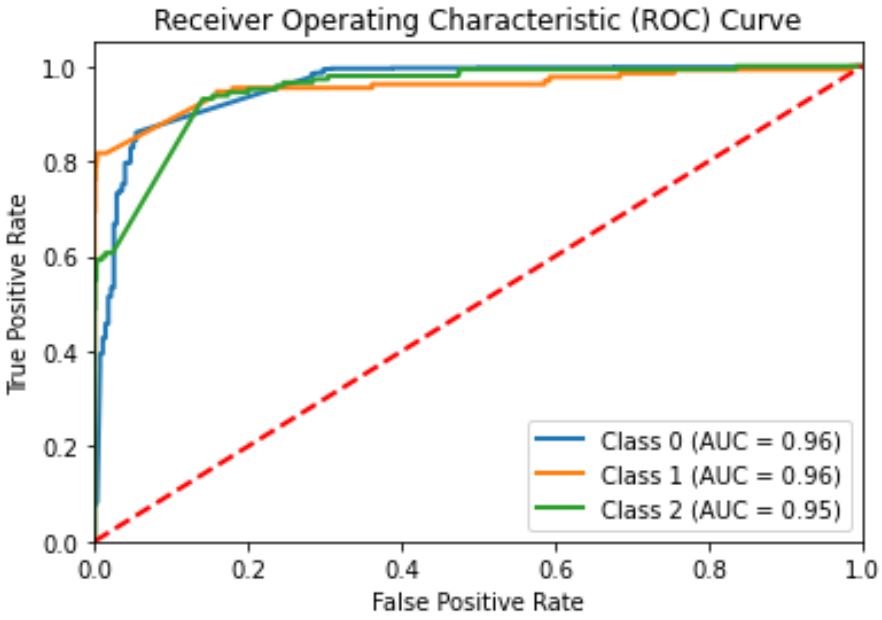


Figure A.60

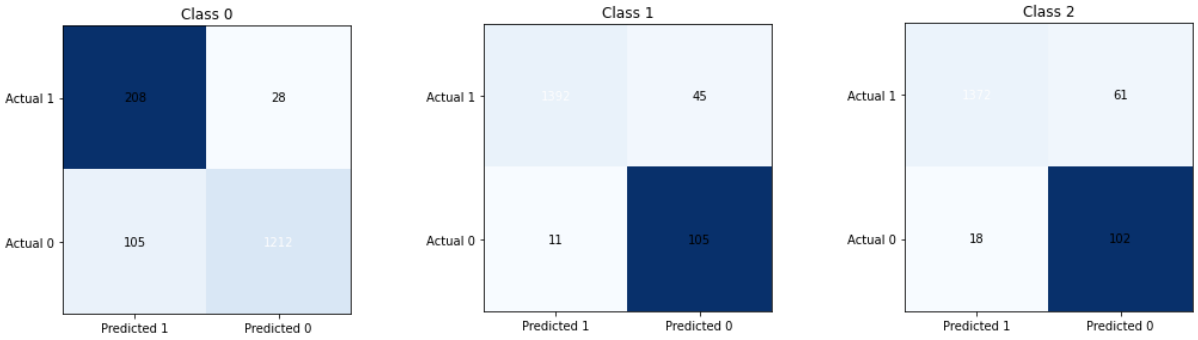
A.1.5.5 Evaluation Results on Day 1, Day 2, Day 3, Day 4, and Day 5 Measurements

Below are the calculated testing metric results for the Respiration Frequency model for measurements taken on Days 1, 2,3,4, and 5 :

Table A.25: Testing Metrics

Metric	Value
Accuracy	0.9137153895685769
Categorical cross-entropy loss	0.2736852965928897
Precision	0.93
Recall	0.91
F1 Score	0.92

Confusion Matrix:



(a) TP, TN, FP, and FN for Class 0

(b) TP, TN, FP, and FN for Class 1

(c) TP, TN, FP, and FN for Class 2

Figure A.61: Confusion Matrix results displaying statistics about Class0, Class1 and Class2 for Respiration Frequency Observation based on Day 1,2,3,4 and 5 Measurements

ROC Curve:

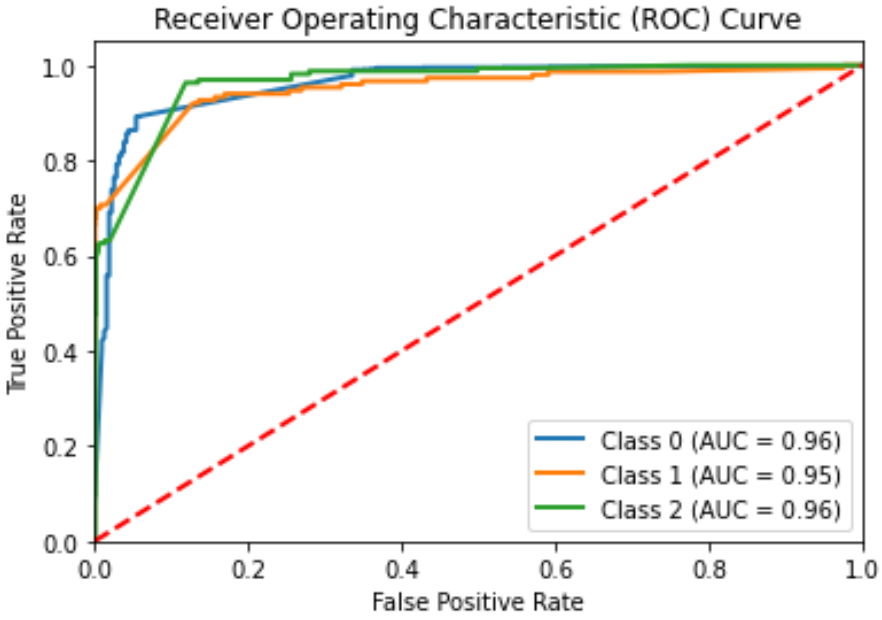


Figure A.62

A.1.5.6 Progression Of Metrics

In the below graph, you will be able to see how the metric values progress over 6 days according to various metrics:

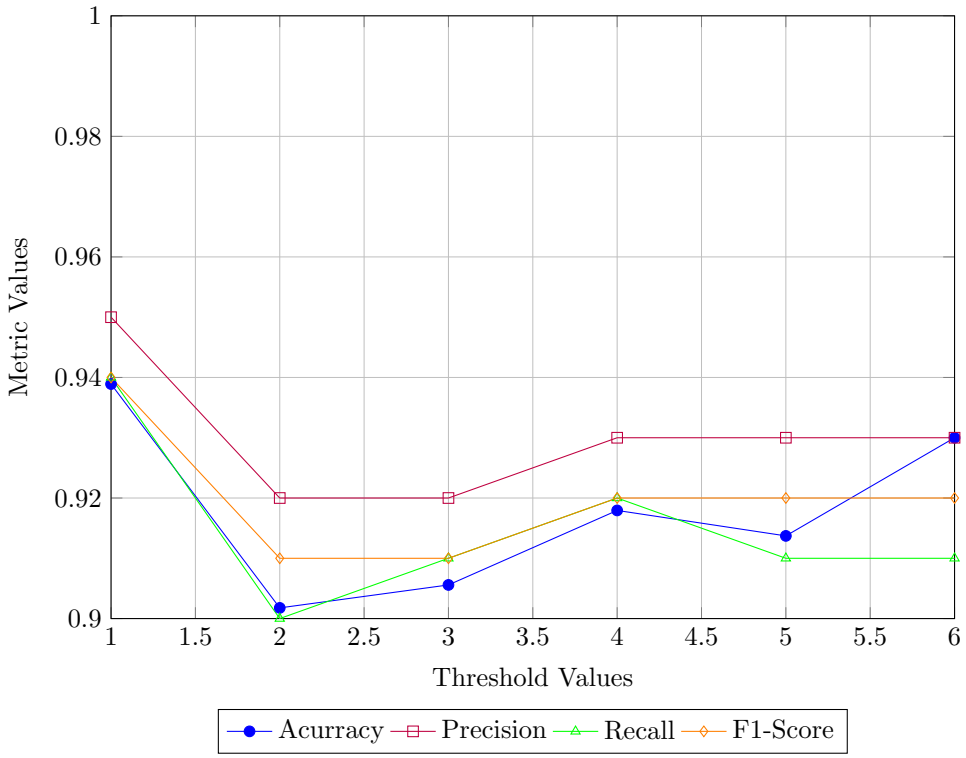


Figure A.63: Progression of Metrics over Different Threshold Values

The performance metrics for the respiration frequency experienced a drop on Day 2 but stabilized around Day 4. In the below graph, you will be able to see the progression of loss:

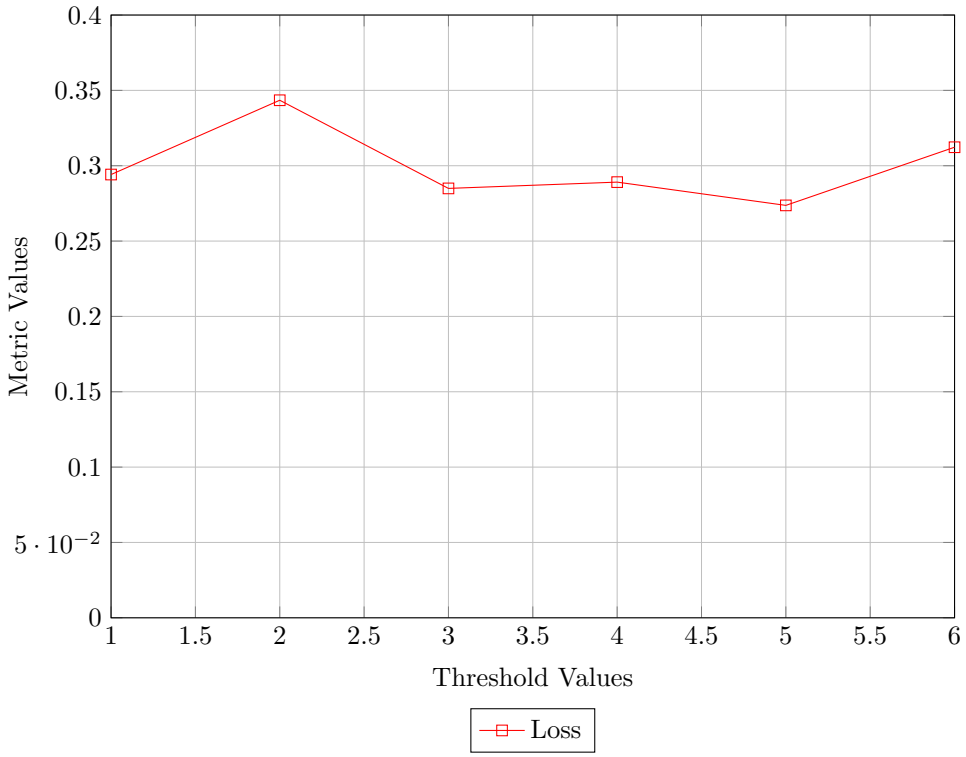


Figure A.64: Progression of Loss over Different Threshold Values

The loss metric was almost stable, except for Day 2 where it first experienced a +0.5 change followed by a - 0.8 change. The remaining values were stable. In the below graph you will be able to see how the

ROC curve for Class0, Class1, and Class2 progress over time:

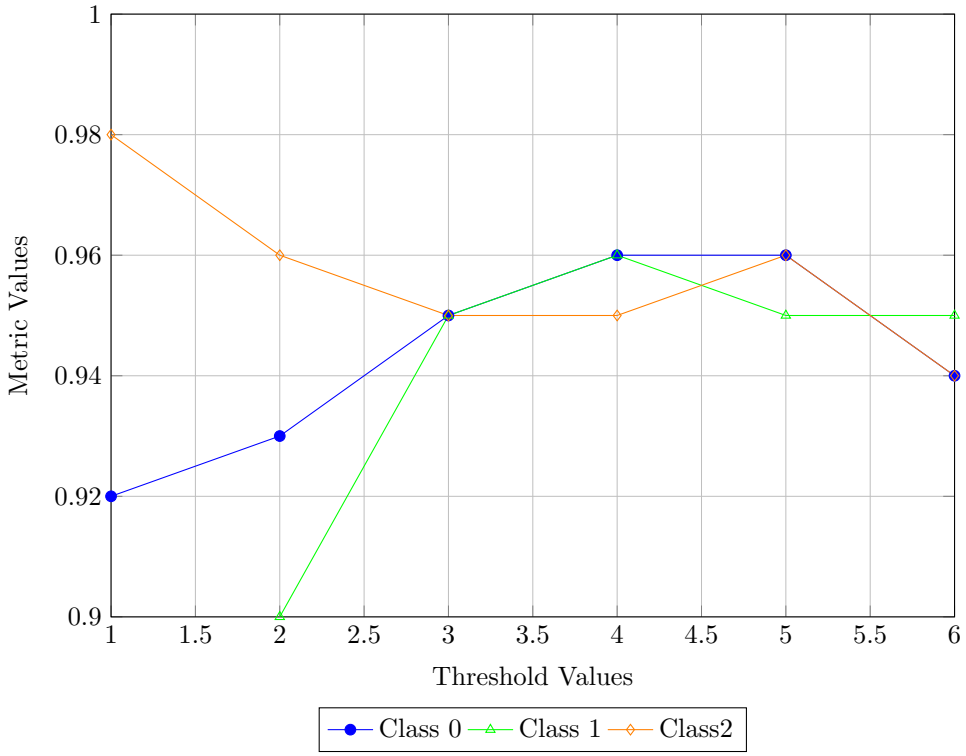


Figure A.65: Progression of AUC Score For Class0,Class1 and Class2

The AUC Scores for Respiration Frequency were unsteady. Class 0 and Class 1 measurements experienced approximately +0.6 change while Class 2 experienced -0.3 change. After a day of stable period, class 0 and Class 2 experienced a -0.2 while Class 1 measurements kept stable.