UNIVERSITY OF LEEDS

Final Report

Early Prediction of Sepsis from Clinical Data

Jack Auty

Submitted in accordance with the requirements for the degree of MEng, BSc Computer Science (MENS)

2019/2020

40 credits

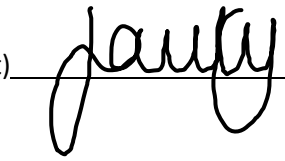
The candidate confirms that the following have been submitted*:*

Items	**Format**	**Recipient(s) and Date**
Deliverable 1 – Final Report	*Report*	*Minerva (16:55 25/05/2020)*
Deliverable 2 – Neural Network	*Software code (classes.py and neuralnetwork.py)*	*Supervisor, assessor (16:55 25/05/2020)*

Type of Project: Theoretical Study

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) 

Summary

Sepsis refers to “a life-threatening condition that arises when the body’s response to an infection injures its own tissues and organs”[1], and is one of the leading causes of death globally today. Like many other serious diseases, the stage of detection is critical when ascertaining the patient outcome of Sepsis. With each hour of delayed treatment representing a 7.6% increase to mortality rate on average, it is clear that an early diagnosis is inherently valuable as a predictor of patient outcome.[7]

This report describes my exploration into the application of a machine learning system in order to attempt to classify sepsis earlier than through clinical methods. It documents each stage in the process and provides a critical evaluation of the final implementation that I produced.

Chapter 1 Introduction

1.1 Aims

The primary aim of this project is to understand how to utilise machine learning to allow for the early detection of sepsis from clinical data; to try and implement this such that sepsis is detected 6 hours earlier than through purely clinical methods. Particularly, if requirements were set, the first would be that a binary prediction should be made at each time interval within patient data, and a risk of sepsis apiece should be discernible. In order to meet the overall goal, the other requirements that I would have to meet include ensuring that the output is 'above challenge level', meaning that the accuracy of my prediction was at least 90%, and that the output format is as specified by Kaggle. That is, where a file is generated per patient, within which rows for the respective rows in the patient file should be output – with their content being the predicted risk of sepsis at that time, along with the binary prediction made.

1.2 Initial Plan and Methodology

The initial plan for the project, as detailed below, was to be followed under an iterative development approach – namely, the incremental build model. Generally this methodology involves splitting the overall project into multiple development cycles whose purpose is to fulfil specific requirements. Furthermore each cycle is broken into easily manageable modules, and "each module passes through the requirements, design, implementation and testing phases."[12]

Under this methodology the project was split into three cycles, named in this instance as the 'background research' cycle, the 'implementation' cycle and the 'evaluation' cycle. Each of these cycles has been covered in further detail within their own chapter of this report – expressly their material implementation and outcome.

The first cycle, 'background research', divided into at first three major areas (modules) for research: sepsis, the use of machine learning for medical purposes, and machine learning platforms. After the conduction of comprehensive research into these areas, the intermediate report could be completed, and an informed choice could be made as to which platform I would be utilising henceforth.

The 'implementation' cycle was to be spread across two modules: constructing the validation and training sets for use within the model and the actual development of the model itself.

Lastly the 'evaluation' cycle was from the outset segmented into four sections: which aims were achieved, which aims were not achieved, how my approach (or its execution) could have been improved, and finally an ethical evaluation.

1.3 Revisions to the Plan

Succeeding a few weeks and further consideration it became evident that some adjustments would have to be made to the initial plan, the updated plan can be seen in Figure 1 (Appendix). Principally that, a not insignificant amount of time would have to be spent on analysing the dataset, to build my knowledge of its structure and to document this, and thus a fourth 'data analysation' cycle arose. Additionally, at this time it was blatant that other development cycles were also in need of further specification, especially the 'background research' and 'implementation' cycles.

Within this cycle, I had to bridge the gaps between the knowledge I had obtained from the background research cycle and my knowledge of the provided dataset, in order to allow for the pursuit of an effective implementation cycle. Therefore the modules of this cycle were itemized to be 'understanding the structure of the data', a 'visual analysis' of the data to try and observe any trends where possible through graphs, and 'initial column determination' – where a rudimentary decision was made as to the supposed priority, of inclusion within the model, of each column within the data.

Upon choosing a platform within the background research cycle, the cultivation of a deeper understanding of that platform became discernibly quintessential to its use within later cycles. Consequently, a fourth module was added to this cycle – 'Understanding PyTorch' – which involved looking at previous examples of implementations through this platform, most notably those at least somewhat analogous to that which I would have to implement.

Only after the partial completion of both the aforementioned module and the 'data analysation' cycle did It become clear that an additional module in the 'implementation' cycle was necessary. Due to the nature of both the dataset, and the platform of choice, additional classes had to be built in order to feed the data into the network/model. These classes are the PyTorch specific dataset and dataloader classes, and thus the additional module within the 'implementation' cycle was devised.

Chapter 2 Background Research

In order to accomplish my objective for this project I need to garner some knowledge of the relevant areas, namely Sepsis, previous applications of machine learning within medicine, and suitable machine learning platforms that could be utilised.

2.1 Sepsis

Sepsis is one of the leading causes of death globally today; the most recent estimates indicate that 11 million people die from it a year, which means that one in five deaths can be attributed to sepsis.[6] Sepsis is defined, following the Sepsis-3 guidelines, as "life-threatening organ dysfunction caused by a dysregulated host response to infection", or in layman's terms "a life-threatening condition that arises when the body's response to an infection injures its own tissues and organs."[1] It is vital that Sepsis be treated as early as possible, as every hour of delay is associated with an average increase in mortality of 7.6%; 6 hour early detection could therefore translate to a 45.6% decreased mortality rate.[7]

2.2 Machine Learning applications in medicine

The use of machine learning across medicinal diagnostics is "in the early adoption phase"[2] and is not yet at a level where integration into the normal diagnostic system is widely considered viable for implementation. Despite this, there has been promising research especially in the Oncology and Pathology diagnostic fields – which encompass the detection of sepsis.

In pathological diagnostics, generally bodily fluids or tissue samples are analysed under microscope manually; with the application of machine learning techniques such as the use of convolutional neural networks (CNNs) this process can be supplemented. A team at Harvard Medical School have used a neural network capable of multiple image and speech recognition in order to diagnose tumours[2], which "proved accurate approximately 92 percent of the time"[3]. This compared to the general success rate of a human pathologist "whose results were 96 percent accurate"[3] is impressive enough, "But the truly exciting thing was when we combined the pathologist's analysis with our automated computational diagnostic method, the result improved to 99.5 percent accuracy"[3].

Under the umbrella of oncological diagnostics, researchers at Oregon State University have made use of deep learning for "… the extraction of meaningful features from gene expression data, which in turn enabled to classification of breast cancer cells."[4] Gene expression data, similarly to the clinical dataset used as a basis for this project, has "… high dimensionality and complexity …"[5] and thus certain aspects of the methodology exercised in this research can be considered applicable within this project. Specifically, the use of "Stacked Denoising Autoencoder (SDAE) to deeply extract functional features from high dimensional gene expressional profiles"[5] and the subsequent

evaluation of this “extracted representation through supervised classification models in order to verify the usefulness of the new features useful for cancer detection.”[5]

2.3 Machine Learning Platforms

2.3.1 Choosing a Platform

The two most popular open source machine learning platforms today are TensorFlow and PyTorch, which were created by Google and Facebook respectively, where “researchers are flocking to PyTorch in droves”[8], but in industry “TensorFlow is currently the platform of choice”.[8] The primary difference between the two is that computational graphs are defined statically in TensorFlow, while they are defined dynamically in PyTorch.[9] A computational graph is “an abstract way of describing computations as a directed graph”[9], the primary benefit of which is allowing parallelism which increases the speed and efficiency of the training process.[9] A statically defined computational graph has all of its nodes and edges predetermined before the model can be executed on it, while a dynamically defined graph can have its structure changed at run time – in the case of PyTorch its structure is determined by the forward computation.[11]

Another significant factor to consider is that "PyTorch optimizes performance by taking advantage of native support for asynchronous execution from Python”[9], meaning that using PyTorch will present the opportunity to spread the computations over multiple ‘devices’ allowing for more time efficient passes to be made through the network. ‘Devices’ in this case refers to the individual cores of the processing units employed for the computations, meaning that either multiple cores of the central processing unit (CPU) or graphical processing unit (GPU) can be used in parallel. This is especially useful when considering the use of a consumer GPU and more specifically its CUDA capabilities, for tensor calculations – which far exceed those of the vast majority of consumer CPUs.

Due to the clear benefits in favour of PyTorch, I decided to utilise this platform instead of TensorFlow in order to build my neuralnetwork.

Chapter 3 Dataset Analysis

3.1 Understanding the Structure of the data

Prior to the application of any machine learning process, it is necessary first for the provided data to be of the correct format, and therefore for an understanding of the structure of the data to be developed through manual and programmatic analysis. The dataset as provided is separated into two directories, one for each hospital from which the data was catalogued, and these directories are populated with a pipe-separated value (psv) file per patient. These text (psv) files contain a row of values, of the 41 recorded variables, for every time interval (hour) of that patient's ICU stay. In this case, 'correct format' refers to two things.

First, the handling of rows in which some of the aforementioned variables are missing, and thus recorded as 'NaN' meaning 'Not a Number'. If these NaN values are allowed to propagate through the network they essentially cause it to break, since gradients cannot be calculated for them, ensuring that no valuable output can be generated. This can be seen in figure 2, in which a loss value is supposed to have been calculated on the validation set subsequent to the evaluation of the model on the training set, but instead this loss value is 'NaN'.

Second, the normalisation of the provided data such that the values in each column lie between 0 and 1. This helps to ensure that each variable is taken into account in a more equal sense, since there is an often-significant difference in range between each variable. For example, O2Sat, the Pulse oximetry is measured as a percentage and thus has a range of 0-100%, whereas pH has a range of 0-14, and Lactate, the level of Lactic acid is measured in mg/dL and thus has a continuous range. Without normalisation this difference in range between variables means that variables with a higher general value will always have a greater impact on the result, despite not necessarily being more important as a predictor.[10]

3.2 A visual analysis of the data

Though not necessary I conducted further analysis of the data, visually, through the generation of graphs and observation of trends from these graphs to help me decide which of the columns were most important to consider for inclusion in my machine learning model. Ideally, the model should incorporate as many of the variables as possible, as their value to the model cannot be accurately determined with ease. However, it becomes obvious that certain columns hold greater significance than others when considering their NaN percentage, i.e. the percentage of values that were not recorded for this variable, as determined through frequency analysis, and their trends when compared between the sepsis set and the non-sepsis set. To produce these graphs the Python module 'matplotlib' and more specifically 'Pyplot' were used. Prior to this the dataset had to be manipulated in order to produce meaningful output, i.e. comparable graphs. This manipulation

consisted of initially producing one large file in which all of the records were collated; from this separate subsets containing only sepsis patients and only non-sepsis patients were created, and finally a set containing an equal number of sepsis patient records as non-sepsis patient records was put together. Sepsis patients refers to patients who at any point in their ICU stay had their 'sepsis_label' value increase to 1. In the creation of the unary subsets a substantial class imbalance was uncovered across the whole dataset, where only 2930 of the 40336 total patient records were found to be sepsis patient records, and the set consisting of equal parts sepsis and non-sepsis records originated to mitigate this class imbalance's effect on the visual analysis.

3.2.1 Histograms

Histograms were the principal form of visual analysis that I took advantage of in order to identify trends in the data. At the outset a histogram per variable, for each of the aforementioned 'collated', 'half', 'sepsis' and 'non-sepsis' datasets were generated – each of which annotated with a rudimentary frequency analysis on the number of NaN values cross referenced against the total number of values shown within that graph. Pyplot's 'hist' function allowed for the probability (density) of each value to be plotted instead of the raw value count, which made a comparative analysis between sepsis and non-sepsis values viable despite the massive class imbalance. Regardless, contrasting multiple singular set histograms proved inefficient at best in allowing for differences in patterns to be distinguished between sets, and thus side-by-side histograms – where multiple sets are plotted on the same axes – became appealing. Side-by-side histograms made some correlations immediately obvious, such as the tendency towards higher heart-rates in sepsis patients than in non-sepsis patients depicted in figure 3.

3.2.2 Scatter plots

Scatter plots were then generated in an effort to compare how multiple of the variables trend in frequency, when considered together as pairs. Taking into account that the values are considered in pairs, a function had to be developed in order to detect NaN values within either of the variable lists used to form these pairs, and then to perform a pairwise removal of those elements from the lists. Alone these plots provided marginal use – mainly allowing for the differences in spread between the two sets to be observed – with the majority of the values obfuscated by other overlapping values, particularly where the frequency is highest.

3.2.3 Combined Metric

Combining the two metrics, Histograms and Scatterplots, achieved the most efficient form of visual analysis, allowing for both the difference in spread and in trends between sepsis and non-sepsis patient values to be easily discernible. In general the spread of values for sepsis patients was wider, containing more extreme values, but the majority of the values remained within the same range as those of the non-sepsis patients.

3.3 Initial Column Determination

The output of each of these forms of analysis were used to determine the priority of each of the variables for inclusion in the machine learning model, with the most weight being placed on the frequency analysis and cases where trends are most obvious.

Chapter 4 Building the network

4.1 Building the training, validation sets

The first step in building the network was to compile the training and validation sets to use within the machine learning network. To accomplish this a seed-able function was created, using the Python module Numpy's 'random' function, to take a random 70% of the sepsis patient records along with the same number of random non-sepsis records to construct a balanced training set. This in order to combat the effects of class imbalance when training the model. The validation set was generated in the same function from the remaining 30% of the sepsis patient records and the rest of the non-sepsis patient records. The unique Ids (UIDs) of the respective records were recorded in list form to pipe-separated values files for both the training and validation sets; these files were then used to rebuild the lists within my neural network so that they could be used to construct the training and validation instances of my 'SepsisDataset' class.

4.2 Building the model

After experimenting with Pytorch following the official Pytorch guides I found that complex Pytorch machine learning networks were usually split across two files, one containing the model and dataset classes, and one the script that makes use of these classes, performing gradient descent on the model.[13] In this case the model was defined in the SepsisModel class which inherits from the 'nn.Module' class of the torch module, and the dataset was specified in the SepsisDataset class which inherits from the 'data.Dataset' class of torch.utils module.

Within the initialisation function of the SepsisModel class potential layers were defined. Two linear transformations, one from a tensor of the input size to the hidden layer size, and one from the hidden layer size to the output size. A variety of activation functions such as ReLU (rectified linear unit layer), Sigmoid and Softmax were additionally implemented, along with two Dropout layers with dropout probabilities of 20% and 50%. The purpose of a Dropout layers is to prevent over-fitting the model on the training set – they work by randomly adjusting the weights of a certain percentage of input nodes to zero, thereby removing them from the calculations.

Then in the forward function, which determines the order in which the aforementioned layers are executed within a forward pass on the network, an at first mostly arbitrary order was implemented. The only constants between this original order and the current one are the hidden layer transformation and the output layer transformation, where the hidden layer transformation has an output tensor of the size of the hidden layer and the output layer transformation has an output of one dimension. The output size is one dimensional as only one value, the risk of sepsis, should be output per row.

4.3 Building the dataset/dataloader

The SepsisDataset class has three inherited functions which must be overwritten, the initialisation function, the length function and the get item function. The initialisation function takes a list of UIDs as input together with the sequence of columns to consider in calculations, these columns were initially those decided upon prior in the analysis section. The get item function is where the largest majority of project time was spent; this function takes a single UID as input and outputs an array of cleaned, normalised values, an array of those values' respective labels and passes through the UID of the patient.

Encompassed in this function is the 'removeRows' function which decides based on the NaN percentage of each row whether that row should be removed from consideration, or whether its NaN values should be replaced with an average of the values from the closest viable prior and subsequent rows. A row was considered viable if it had fewer than one shared NaN columns, i.e. column which for both rows did not have a recorded value. Functionality was also developed to allow for this decision to be made on a column by column basis – since some variables can be more easily and accurately determined than others, one example of which is temperature values.

Furthermore as mentioned previously, this is the function in which normalisation of the data should be carried out as, the output of this function is the tensor to be propagated through the network.

Chapter 5 Evaluation

5.1 Which aims have been met?

While it is clear that I have been unable to meet the primary objective for this project, six-hour early detection of sepsis from clinical data using a machine learning system, I have provided the foundation from which this objective could be accomplished. This foundation being a neural network that attempts to classify sepsis based on the provided data, calculating the risk of sepsis for the patient at each time interval within the patient' ICU stay. Since the accuracy of this classification is questionable, and the respective risk values are not output to file as specified within the Kaggle competition specification, nor are they output (individually) to console, it would be a groundless claim to say that the objective has been achieved. Instead of the of their output to console, a mean squared error (MSE) loss value is calculated over each patient's data which gives an idea as to how close the network was to achieving the correct result, averaged over all of the patient time intervals.

Despite this, a working neural network has been built that takes the training input data, as fed to it successfully by the dataloader, and carries out back-propagation of it against its respective labels while saving the gradients; it also then takes the validation input data and uses the stored gradients to evaluate the model on it, giving an individual prediction for each row, however inaccurate. In fact the accuracy of my predictions can be easily scrutinized to be worse than that of purely clinical methods.

Furthermore, a binary classification function has been implemented within the 'confusionMatrix' function of the neural network, which simply takes a cut off value for these individual predictions, and anything above that value is given a binary prediction of 1, predicting sepsis at this time interval, while anything below is given a value of 0. Therefore, it could be argued that I have achieved the first requirement that was set out but failed to achieved the remaining requirements for this objective.

5.2 How could my approach have been improved

My approach could have been improved in a multitude of ways, probably the most significant of which would have been to use a more in depth methodology such as SMART for determining the requirements of my project and sticking to them without getting too absorbed in one area and spending all too much time in focus on that area. Likewise my choice of PyTorch over TensorFlow, despite my background in TensorFlow, likely impacted the outcome, along with my timeliness – my delay in starting the project from October till February meant that I couldn't get ahead of my schedule and prevented me from producing the highest quality output that I otherwise could have achieved. Another thing would have been to follow my plan more closely, and to document changes to my plan more thoroughly – the majority of my documentation was through comments in code, which would then often later be truncated, if not deleted, once that section was complete.

5.3 Legal, Social, Ethical and Professional Issues

This project makes use of an anonymised medical dataset made public specifically for the purpose laid out in this report, and as such there are no legal or ethical boundaries set nor requirements to meet.

6 Conclusion

The outcome of my project, where there is a lack of requirements met towards the primary objective, can be mainly attributed to the poor planning procedures that I followed, and my failure to assign these requirements using a specific methodology, such as SMART, to ensure their viability within the time-frame of the project, and to make the documentation of my progress towards them more well-grounded.

Ultimately, while I have been unable to successfully thoroughly implement a machine learning network to fully accomplish the specified objective, I do not believe this to be completely out of the grasp of a project of this scale, and had I provided myself more time I would expect to meet all of the requirements of the project. Therefore, and considering that my understanding of this topic has been developed greatly, I would consider the project to have been at least partially successful. I have every confidence that a well thought out and implemented machine learning system could be used for the early prediction of sepsis from clinical data, such that it is detected six hours earlier than through purely clinical methods.

References

1. Singer, M et al. (2016) *The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3).* National Centre for Biotechnology Information [Online]. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4968574/ [Accessed: 3rd February 2020]

2. Sennaar, K. (2019) *Machine Learning for Medical Diagnostics – 4 Current Applications* [Online]. Available from: https://emerj.com/ai-sector-overviews/machine-learning-medical-diagnostics-4-current-applications/ [Accessed: 3rd February 2020]

3. Prescott, B. (2016) *Better Together* [Online]. Available from: https://hms.harvard.edu/news/better-together [Accessed: 4th February 2020]

4. Ali, A. (2019) *Deep Learning in Onocology - Applications in Fighting Cancer* [Online]. Available from: https://emerj.com/ai-sector-overviews/deep-learning-in-oncology/ [Accessed: 4th February 2020]

5. P, Danaee et al. (2017) *A DEEP LEARNING APPROACH FOR CANCER DETECTION AND RELEVANT GENE IDENTIFICATION.* [Online]. Available from: https://www.ncbi.nlm.nih.gov/pubmed/27896977 [Accessed: 4th February 2020]

6. Gallagher, J. (2020) *'Alarming' one in five deaths due to sepsis.* [Online]. Available from: https://www.bbc.co.uk/news/health-51138859 [Accessed: 7th February 2020]

7. Kumar, A et al. (2006) *Duration of hypotension before initiation of effective antimicrobial therapy is the critical determinant of survival in human septic shock.* National Centre for Biotechnology Information [Online]. Available from: https://www.ncbi.nlm.nih.gov/pubmed/16625125 [Accessed: 7th February 2020]

8. The Gradient. (2019) *The State of Machine Learning Frameworks in 2019.* [Online]. Available from: https://thegradient.pub/state-of-ml-frameworks-2019-pytorch-dominates-research-tensorflow-dominates-industry/ [Accessed: 8th February 2020]

9. Kurama, V. (2019) *PYTORCH VS. TENSORFLOW: WHICH FRAMEWORK IS BEST FOR YOUR DEEP LEARNING PROJECT?* [Online]. Available from: https://builtin.com/data-science/pytorch-vs-tensorflow [Accessed: 8th February 2020]

10. Jaitley, U. (2018) Why Data Normalization is necessary for Machine Learning models [Online]. Available from: https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029 [Accessed: 15th May 2020]

11. Ayman, O. (2019) Pytorch or Tensorflow, Dynamic vs Static computation graph [Online]. Available from: https://medium.com/analytics-vidhya/dynamic-vs-static-computation-graph-2579d1934ecf [Accessed 15th May 2020]

12. TRY QA. (2017) What is Incremental model- advantages, disadvantages and when to use it [Online]. Available from: http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/ [Accessed 15th May 2020]

13. PyTorch (2017) Developing Custom PyTorch Dataloaders [Online]. Available from: https://pytorch.org/tutorials/recipes/recipes/custom_dataset_transforms_loader.html?highlight=dataset [Accessed 21st April 2020]

Appendix

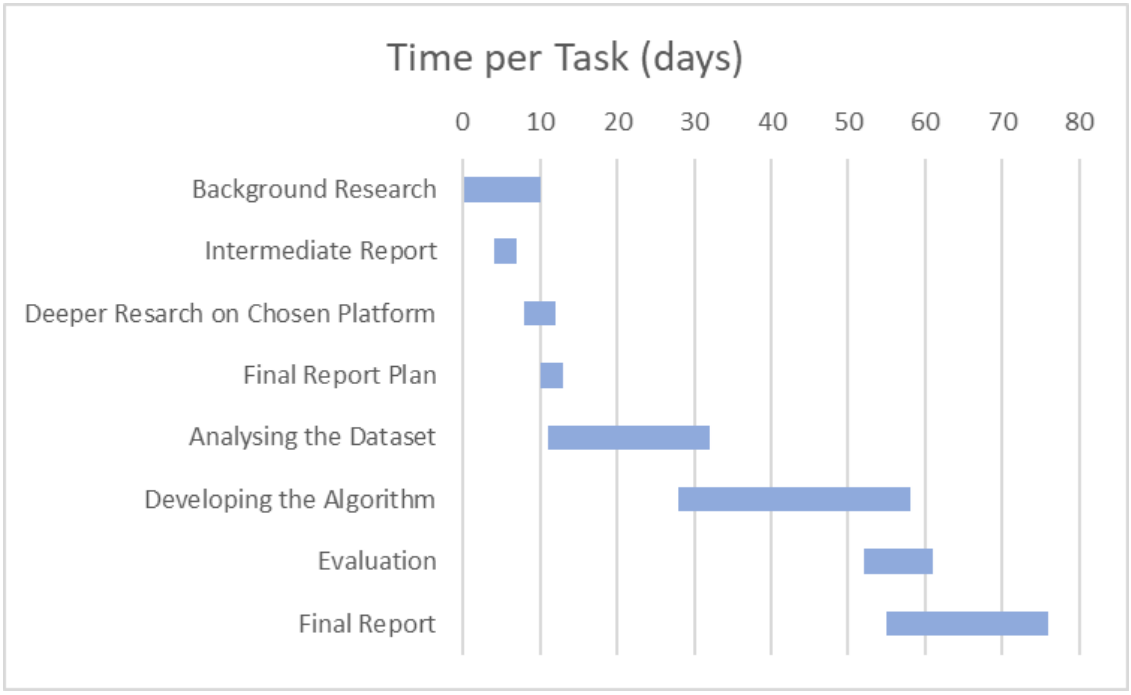
1. Gitlab URL: https://gitlab.com/JackAuty/comp3931-individual-project.git
2. 

Figure 1: Gantt Chart of the Updated Project Plan

3. 

Figure 2: NaN Propagation through the network

4.

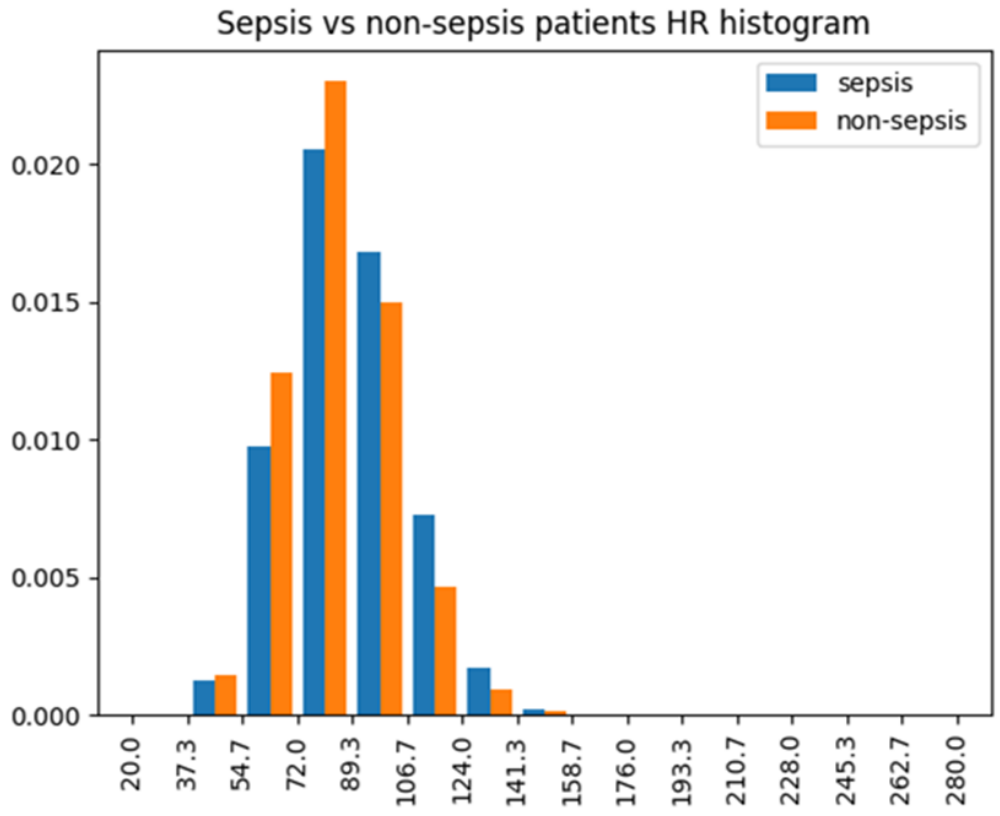


Figure 3: Side-by-side probability histogram

5.

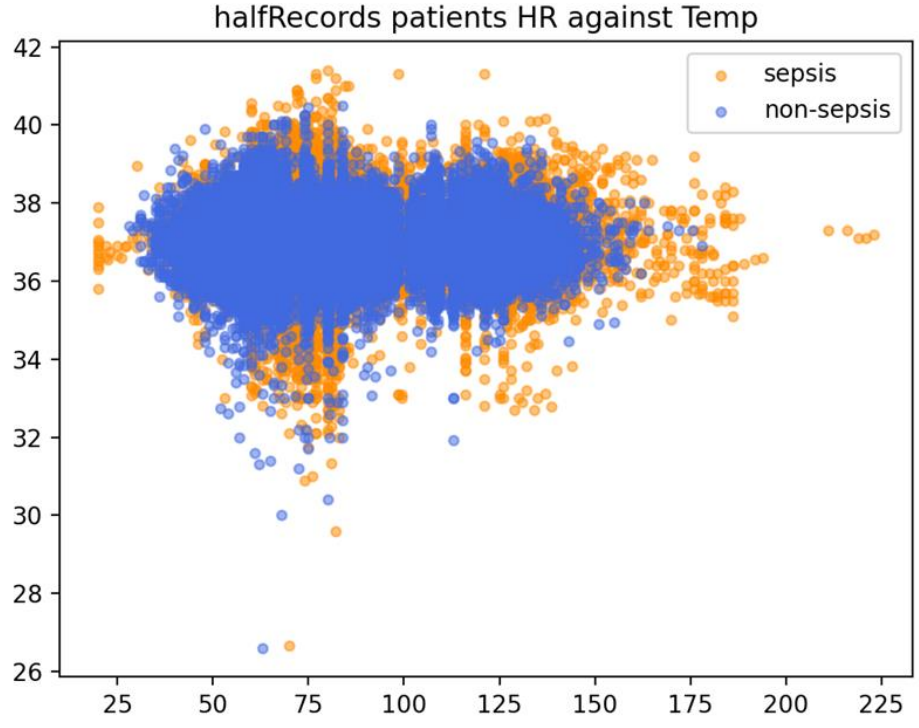
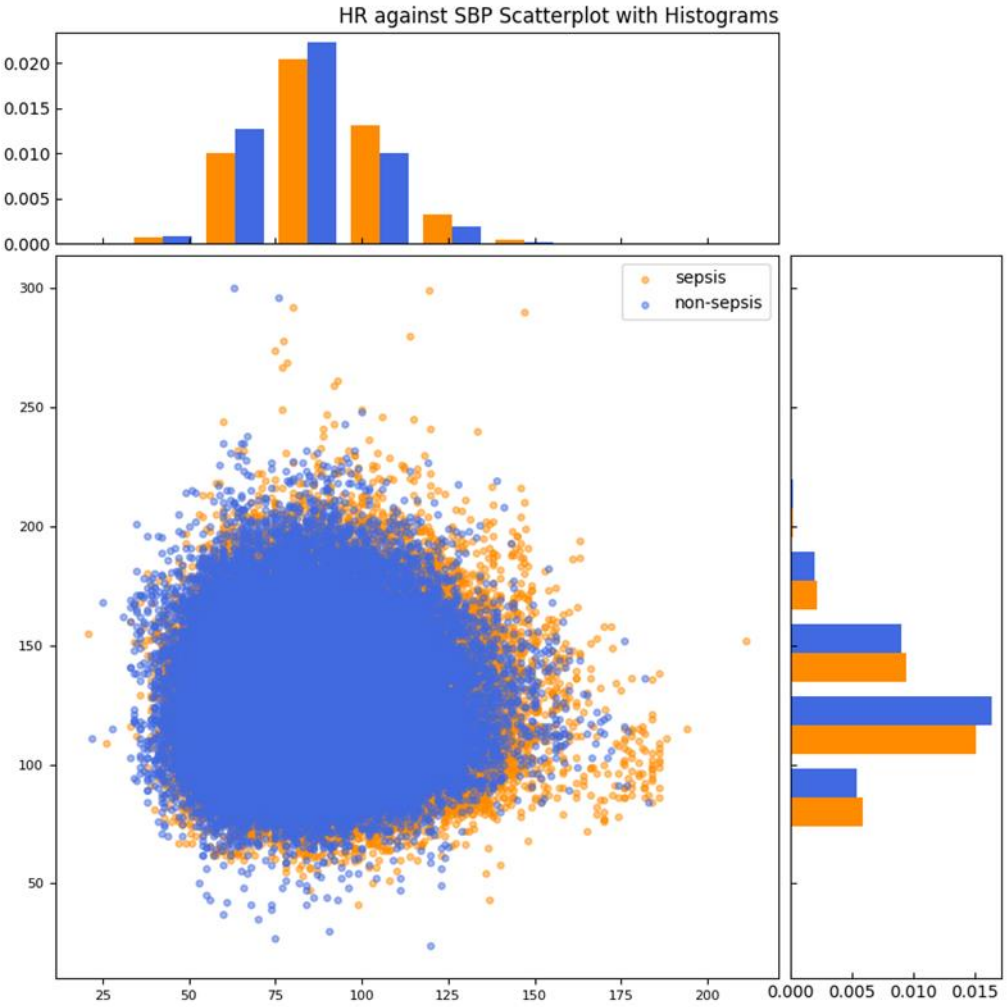


Figure 4: Scatterplot



6.

Figure 5: Combined Scatterplot and Side-by-side density histograms