



A Comparative study between neural network models and standard machine learning models in heart disease prediction

Dhanush Vinay Uttarkar

10611336

Applied Research Project submitted in partial fulfilment of the

requirements for the degree of

Masters of Science in Data Analytics

at Dublin Business School

Supervisor: Dr. Vivek Kshirsagar

August 2023

DECLARATION

I, Dhanush Vinay Uttarkar, declare that the dissertation that I have submitted to Dublin Business School for the award of MSc in Data Analytics is the result of my own investigations, except where otherwise stated, where it is clearly acknowledged by references. Furthermore, this work has not been submitted in whole or in part for any other degree or qualification.

Signed: Dhanush Vinay Uttarkar

Student Number: 10611336

Date: 29/08/2023

ACKNOWLEDGMENTS

First, I would like to thank my parents and my brother for their continuous support and help without who I would never be here. Their continuous support has been critical in my time away from home achieving my educational goals.

Next, I would like to thank my guide Dr Vivek Kshirsagar who helped me and guided me through the entirety of this project, helped me solve any problems I encountered and rid any doubt I had with regard to this applied research project. I would also like to extend my thanks to the teaching staff who imparted their knowledge to us through the year teaching us new and important things which is fundamental to my growth without who this project would be very difficult to complete.

Last but not least, I would like to thank Dublin Business School for giving me this amazing opportunity to not only showcase my skill but also implement what we learnt throughout the year in the course and special thanks to my friends and classmates who helped me and collaborated with me to enhance the learning experience.

Table of Contents

DECLARATION	1
ACKNOWLEDGMENTS	2
ABSTRACT	4
PROBLEM STATEMENT	5
KEY LITERATURE	6
3.1 Introduction	6
3.2 Literature Review	6
3.3 Our Contribution	9
RESEARCH APPROACH	10
4.1 Data source	10
4.2 Design	15
4.2.1 Artificial Neural Networks	15
4.2.2 Convolutional Neural Networks	20
4.2.3 Recurrent Neural Networks	25
4.2.4 Training an Artificial Neural Network	32
4.2.5 Approximation functions	35
4.3 Procedure	37
4.3.1 Background	37
4.3.2 Data pre-processing	38
4.3.3 Feature selection	41
4.3.4 Modelling	42
4.5 Ethics	43
4.6 Data Analysis	43
RESULT	46
5.1 Introduction	46
5.2 Descriptive statistics	46
5.3 Inferential statistics	48
CONCLUSION	49
REFERENCES	50

CHAPTER 1

ABSTRACT

Cardiovascular diseases are the leading cause of death across the world. There has been a gradual increase in cardiovascular diseases. It is also called a ‘Silent Killer’ because most people who have it do not have obvious symptoms. An artificial neural network is a part of machine learning with a potential solution to identify or detect the onset of this disease effectively. Artificial neural network (ANN) is a technological advancement in the field of machine learning that is gaining a lot of traction because of its design which allows it to solve many complex problems. ANNs are playing a vital role in many sectors of the industry, it is used for financial data analysis, speech recognition, emotion detection, disease prediction, image generation, and many other such applications. Artificial neural networks are made to imitate the human brain and function like one. It computes data like the human brain. The nodes of the artificial neural network resemble the individual neurons of the human brain. Our report analyses and compares the prediction of heart diseases between a few previously implemented standard machine learning models such as the logistic regression model, an ensemble model like random forest and decision trees versus neural networks like Long Short Term Memory Networks (LSTM), which is a type of recurrent neural network (RNN), a simple Convolutional Neural Network (CNN), a simple recurrent neural network, and a feed-forward neural network. To perform this experiment of comparative analysis we use the Cleveland heart disease dataset available at the UCI (University of California, Irvine) machine learning repository donated by Peter Turney, we compare their precision, accuracy, sensitivity, and specificity. The results we obtained were 72.49 % accuracy for the LSTM model, 75.73% accuracy for the CNN model, 74.43% accuracy with the feed forward neural network, 72.17% accuracy for the RNN model.

CHAPTER 2

PROBLEM STATEMENT

The problem statement is to perform a comparative study between the accuracy of predicting heart disease using traditional machine learning techniques like logistic regression, random forest, Support vector machines and a few others and neural networks models like feed forward neural networks, convolutional neural networks (CNN), recurrent neural networks (RNN) and a special RNN called a long short-term memory (LSTM) model.

While traditional models were implemented in previous literature, there were no conclusive studies conducted where different neural networks were applied to predict heart disease which was the core reason to conduct the experiment.

CHAPTER 3

KEY LITERATURE

3.1 Introduction

Heart disease prediction has been a topic of research for many years now and many heart disease prediction systems are built using both machine learning as well as neural networks (deep learning) algorithms. Some of these works have been discussed in the “Literature Review” section.

3.2 Literature Review

In (Rani, et al. 2021) the authors approached the problem by using multivariate imputation by chained equations algorithm to handle missing values while on the other hand it used a hybridized feature selection algorithm to grasp all the important features in the dataset. This is done by combining the Genetic Algorithm (GA) and the recursive feature elimination technique for selection of suitable features from the available dataset. The authors have also used SMOTE and a standard scalar for data pre-processing and compared five machine learning classifier algorithms (SVM, Naive Bayes, logistic regression, random forest, and AdaBoost) to develop a hybrid decision support system for heart disease prediction. They found that the random forest classifier gave the most accurate results, achieving 86.6% accuracy on the Cleveland heart disease dataset from the UCI machine learning repository. The proposed system outperformed some existing heart disease prediction systems in the literature.

Various algorithms have been used by (Ahmad, et al. 2022) such as Logistic Regression (LR), K Nearest Neighbors (KNN), Support Vector Machines (SVM), Gradient Boosted Classifier (GBC) and GridSearchCV to predict heart diseases. From this group of algorithms, the Extreme Gradient Boosting Classifier with GridSearchCV produces the results with the

highest accuracy of 100% and 99.03% respectively for both the “Hungary, Switzerland & Long Beach V” and the “Heart Disease UCI Kaggle” Datasets. The approach considered here has the best accuracy of the four models that have been used. In this study the four mentioned algorithms i.e., LR, KNN, SVM and Extreme GBC (XGB) have been compared with and without using techniques such as hyperparameter tuning. The modification of the parameters of XGB with GridSearchCV led to obtaining an accuracy of 99.03% and without GridSearchCV, the accuracy was noted to be 98.05%.

From (Olaniyi, et al, 2015) it can be observed that the dataset has been broken down into training, test and validation sets while being fed into a feed-forward multilayer perceptron as well as support vector machine models. In this study, the goal was to determine which model suited best for the dataset. From the experiments, the feed-forward model produced an accuracy of 85% while the support vector machine had an accuracy of 87.5%. Thus, the authors concluded that the support vector machine was best suited to model the heart disease dataset.

The authors (Das, et al, 2009) proposed a method which involves using a SAS based software to diagnose heart disease for the Cleveland heart disease dataset. A neural network ensemble method is used to create more effective models on which experiments were performed. The results for the Cleveland heart disease database produced 89.01% classification accuracy along with an 80.95 and 95.91 sensitivity and specificity values.

The authors (Mohan, et al, 2019) of “Heart Disease Prediction using Hybrid Machine Learning Model” as the title suggests is a recent research paper that proposes a hybrid random forest with a linear model (HRFLM) which is a novel method for predicting heart disease. By utilizing the Cleveland heart disease dataset, which contains 303 instances with 14 features, including age, sex, cholesterol level, blood pressure and other medical parameters, experiments are performed and the results are recorded. It was observed that the method proposed in this

paper obtained an accuracy of 88.7% for heart disease prediction which outperformed other classification techniques such as decision trees, random forests and support vector machines. Using this novel approach, the HRFLM model is produced by combining random forest and linear regression models wherein the accuracy of heart disease prediction is significantly improved. The paper also used techniques such as feature selection and cross-validation to improve the performance of the models. By using machine learning techniques and introducing a novel approach, the significance of this paper in the domain of machine learning is high. This promising approach allows for early detection of heart disease and prevention of diseases related to cardiovascular health. The findings in this paper along with their methodology could have greater impact and using techniques such as feature selection and cross-validation could be used with other techniques further improve accuracy of the model.

In this paper (Al Ahdal, et al, 2023) "Monitoring Cardiovascular Problems in Heart Patients Using Machine Learning " investigates machine learning techniques and their uses for detection of cardiovascular diseases in heart patients during early stages. For this work, the UCI machine learning heart disease dataset is used and various algorithms are evaluated on it. The algorithms are random forest classifier and extreme gradient boost while achieving high accuracy rates of 96.72% and 95.08% respectively. This work places significance on the importance of computer-aided machine learning diagnosis along with detection applications in the healthcare industry particularly. Early detection in most cases can drastically improve the outcome of the state of the patient as the disease progresses. This particular research contributes to computer-aided diagnosis in healthcare by highlighting its potential for machine learning algorithm usage such as monitoring and detecting cardiovascular problems in heart patients. The broader research of this study concentrates on other data mining techniques for heart diseases. The paper concludes by stating the accuracy and stating early detection of such diseases would be the best for patients.

3.3 Our Contribution

The goal for this project was to compare the outcomes of classifying and prediction the possibility of heart diseases using artificial neural networks, unlike other papers that used machine learning models we decided to apply. A wide variety of artificial neural networks such as the feed forward neural network, Convolutional neural network (CNN), Recurrent neural network (RNN) and a special type of RNN called the Long Short-Term Memory (LSTM) model were applied to the dataset to perform a comparative study between these and the traditional machine learning models used in other studies performed previously in different literature. Not all neural networks may be suited in classifying this type of data it was interesting to see how these models fared against their traditional counterparts, what kind of data preparation they require, the hyper-parameters that need to be set and how they affect the final result.

CHAPTER 4

RESEARCH APPROACH

4.1 Data source

For this project, the “Cleaveland heart disease” dataset available on the UCI website has been used, which provides us with a large enough dataset to experiment on. The dataset was originally created by Andras Janosi, William Steinbrunn, Matthias Pfisterer, and Robert Detrano and donated to UCI by Peter Turney. The experiment (Detrano, et al, 1989) consisted of three patient test groups being tested for **angiographic coronary disease** at the Cleveland Clinic between May 1981 and September 1984. No patient had a history or electrocardiographic evidence of previous myocardial infarction or valvular or cardiomyopathic disease with the mean age of the patients being 54 years. The patients underwent physical exams and provided their medical history, the tests they underwent were the electrocardiogram at rest, fasting blood sugar determination and serum cholesterol determination coupled with this and after receiving the patient’s informed consent, they were given three additional non-invasive tests which were namely exercise electrocardiogram, thallium scintigraphy and cardiac fluoroscopy. The results of these tests were not interpreted until coronary angiograms. The historical data was recorded and coded before information about the noninvasive or angiographic test data was relayed to the team recording the historical data which indicates that there was no work-up bias not present.

The dataset contains multiple sets of data from different cities across the world but for this project, the Cleaveland heart disease dataset was chosen. The dataset contains seventy-six variables each correlated to the target variable num which indicates if the patient suffered from a heart attack or not. The dataset is quite popular among researchers as well with sixty-four

citations. The original Cleaveland heart disease dataset at the time of selection for this project was unfortunately corrupted but fortunately, a copy of the dataset was reuploaded in its raw form so there was a lot of formatting needed to make sure the data was in a usable format. The initial data exists on a “.data” file which needed to be converted to a CSV format to make it easier to work with.

The figure 1.1 below depicts how the data looks in its raw form before pre-processing

```

1 15943882 63 1 -9 -9 -9
-27 1 145 1 233 -9 50 20
1 0 1 2 2 3 1981 0
0 0 0 0 1 10.5 6 13
150 60 190 90 145 85 0 0
2.3 3 -9 -9 0 -9 -9 -9
-9 -9 -9 6 -9 -9 -9 2
16 1981 0 1 1 1 -9 1
-9 1 -9 1 1 1 1 1
1 1 -9 -9 0 -9 -9 -9
-9 -9 -9 -9 -9 -9 0 0
0 0 name
2 15964847 67 1 -9 -9 -9
-27 4 160 1 286 -9 40 40
0 0 1 2 3 5 1981 0
1 0 0 0 1 9.5 6 13
108 64 160 90 160 90 1 0
1.5 2 -9 -9 3 -9 -9 -9
-9 -9 -9 3 -9 -9 -9 2
5 1981 2 1 2 2 -9 2
-9 1 -9 1 1 1 1 1
1 1 -9 -9 0 -9 -9 -9
-9 -9 -9 -9 -9 -9 0 0
0 0 name

```

Figure 1.1 Cleaveland Heart disease dataset before pre-processing

The 76 variables of the dataset are as follows:

1. id: patient identification number

2. ccf: social security number (I replaced this with a dummy value of 0)
3. age: age in years
4. sex: sex (1 = male; 0 = female)
5. painloc: chest pain location (1 = substernal; 0 = otherwise)
6. painexer (1 = provoked by exertion; 0 = otherwise)
7. relrest (1 = relieved after rest; 0 = otherwise)
8. pncaden (sum of 5, 6, and 7)
9. cp: chest pain type, Value 1: typical angina, Value 2: atypical angina, Value 3: non-anginal pain, Value 4: asymptomatic
10. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
11. htn: Undefined
12. chol: serum cholesterol in mg/dl
13. smoke: I believe this is 1 = yes; 0 = no (is or is not a smoker)
14. cigs (cigarettes per day)
15. years (number of years as a smoker)
16. fbs: (fasting blood sugar > 120 mg/dl), (1 = true; 0 = false)
17. dm (1 = history of diabetes; 0 = no such history)
18. famhist: family history of coronary artery disease (1 = yes; 0 = no)

19. restecg: resting electrocardiographic results, Value 0: normal, Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation depression of > 0.05 mV), Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
20. ekgmo: month of exercise ECG reading
21. ekgday: day of exercise ECG reading
22. ekgyr: year of exercise ECG reading
23. dig digitalis used during exercise ECG: 1 = yes; 0 = no
24. prop (Beta-blocker used during exercise ECG: 1 = yes; 0 = no)
25. nitr (nitrates used during exercise ECG: 1 = yes; 0 = no)
26. pro (calcium channel blocker used during exercise ECG: 1 = yes; 0 = no)
27. diuretic (diuretic used during exercise ECG: 1 = yes; 0 = no)
28. proto: exercise protocol 1 = Bruce, 2 = Kottus, 3 = McHenry, 4 = fast Balke, 5 = Balke, 6 = Noughton, 7 = bike 150 kpa min/min, 8 = bike 125 kpa min/min, 9 = bike 100 kpa min/min, 10 = bike 75 kpa min/min, 11 = bike 50 kpa min/min, 12 = arm ergometer
29. thaldur: duration of exercise test in minutes
30. thaltime: time when the ST measure depression, was noted
31. met: mets achieved
32. thalach: maximum heart rate achieved
33. thalrest: resting heart rate
34. tpeakbps: peak exercise blood pressure (first of 2 parts)
35. tpeakbpd: peak exercise blood pressure (second of 2 parts)

36. this is a dummy value
37. trestbpd: resting blood pressure
38. exang: exercise-induced angina (1 = yes; 0 = no)
39. xhypo: (1 = yes; 0 = no), Unfortunately, the data does not describe what this variable is.
40. oldpeak = ST depression induced by exercise relative to rest
41. slope: the slope of the peak exercise ST segment, Value 1: upsloping, Value 2: flat, Value 3: downsloping
42. rldv5: height at rest
43. rldv5e: height at peak exercise
44. ca: number of major vessels (0-3) colored by fluoroscopy
45. restckm: irrelevant
46. exerckm: deemed irrelevant by the author
47. restef: rest radionuclide (sp) ejection fraction
48. restwm: rest wall (sp) motion abnormality, 0 = none, 1 = mild or moderate, 2 = moderate or severe, 3 = akinesis (sp)
49. exeref: exercise radionuclide (sp) ejection fraction
50. exerwm: exercise wall (sp) motion
51. thal: 3 = normal, 6 = fixed defect, 7 = reversible defect
- 52 to 54 are not defined either and it is possible this data was lost when the dataset was corrupted or these values were simply never defined from the start.

52. thalsev, 53. thalpul, and 54. Earlobe are not defined

55. cmo: month of cardiac call

56. cday: day of cardiac call

57. cyr: year of cardiac call

58. num: diagnosis of heart disease (angiographic disease status), Value 0: < 50% diameter narrowing, Value 1: > 50% diameter narrowing

In any major vessel: attributes 59 through 68 are vessels and are values that are not defined.

59 lmt, 60. ladprox, 61. laddist, 62. diag, 63. cxmain, 64. ramus, 65. om1, 66. om2, 67. rcaprox, 68. rcadist, 69. lvx1, 70. lvx2, 71. lvx3, 72. lvx4, 73. lvf, 74. cathef, 75. junk, 76. name: last name of patient (has been replaced with the dummy string "name" for some cases where the data was previously corrupted)

These are the 76 variables that are in the dataset with the target variable being “num” which tells if the patient suffered from heart disease or not.

4.2 Design

4.2.1 Artificial Neural Networks

A neural network of ℓ layers is considered and denote the model as $\theta = \{(W^0, b^0), (W^1, b^1), \dots, (W^{\ell-1}, b^{\ell-1})\}$ where W^i is the weight matrix and b^i is the bias vector, $0 \leq i \leq \ell-1$. The neural network training uses multiple functions and runs the input data through multiple layers made of neurons. The neurons make up the input layer, hidden layers and the output layer of a neural network as described in the Figure 1.2 below.

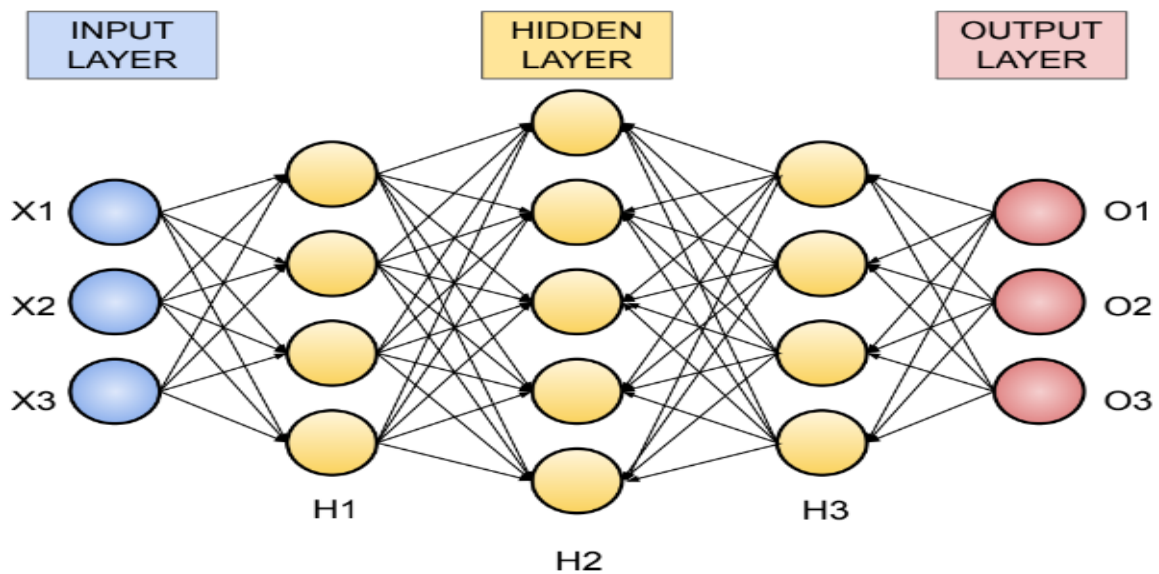


Figure 1.2: Artificial neural network

All these layers consist of a combination of activation functions and linear transformations (such as vector or matrix multiplication). In the training process, the gradient computation for input has two main phases: the forward propagation phase (or forward pass) and the backward propagation phase (or the backward pass). Each phase of the neural network implements a linear layer and a non-linear layer for each hidden layer. Below is the description of each layer in detail.

Linear Layer Computation:

Given the weight matrix and the bias vector (W^i, b^i) for a linear layer and an input a_i to it, the output of the linear layer is $c_i = W^i \cdot a_i + b^i, 0 \leq i \leq \ell - 1$, where $W^i \cdot a_i$ is the matrix-vector multiplication.

Activation/Non-linear Functions:

Activation functions are used after the execution of each linear layer. These functions are introduced to bring non-linearity to the entire computation. This is an important feature of

neural networks that differentiates them from other machine learning algorithms. The non-linearity in the model plays a major role in understanding the data better and allows for an appropriate model weight estimation which consequently leads to better accuracy in predicting the output classes. The forward propagation utilizes the activation functions while the backward propagation utilizes the derivative of the respective activation function. There are many activation functions but the most commonly used ones are summarized below.

Sigmoid: The sigmoid function (logistic) is a differentiable monotonically increasing function. It takes a real-valued number that can be of any magnitude and maps it to a range between [0, 1]. For negative numbers, it maps the value to 0 and for large and positive numbers, it maps the value to 1 as seen in Figure 1.3. The equation for Sigmoid is as written below:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

where σ is the activation function, and z is the input to the activation function.

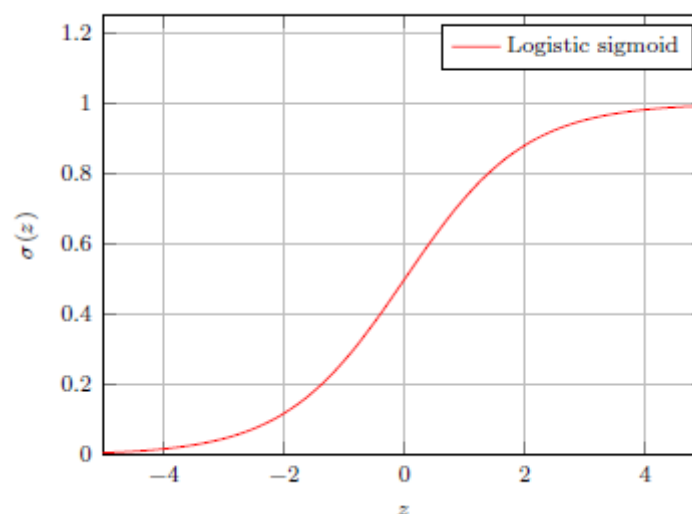


Figure 1.3: Logistic Sigmoid Activation Function

Hyperbolic Tangent: The hyperbolic tangent (tanh) function is another differentiable monotonically increasing function. It takes in a real-valued input of any magnitude and maps it to a range $[-1, 1]$ seen in Figure 1.4. tanh is chosen when small incremental or decremental changes are required in data modelling. The equation is given below:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{1 - e^{-2z}}{1 + e^{-2z}}$$

where tanh is the activation function, and z is the input to the activation function.

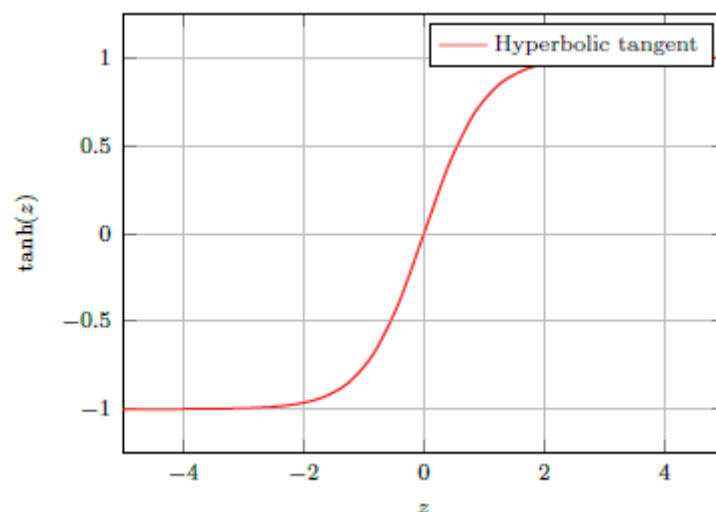


Figure 1.4 Hyperbolic Tangent Activation Function

Rectified Linear Unit: Rectified linear unit (ReLU) is currently the most popular non-linear activation function being used and it is represented in Figure 1.5. ReLU is preferred because of the improvement in performance as well as simplicity in its usage. It is defined below:

$$\text{ReLU}(z) = \max(0, z)$$

where ReLU is the activation function, and z is the input to the activation function.

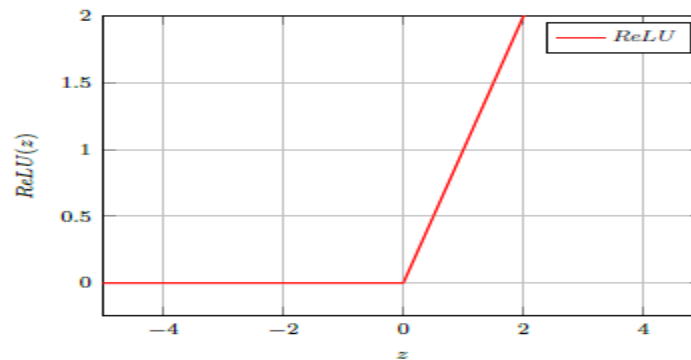


Figure 1.5: Rectified Linear Unit Activation Function

Softmax: The softmax function is mostly used for categorical probability distribution. This classifies the final layer nodes of the neural network into output classes containing the probability for each class. The function is defined as follows:

$$\sigma_{sm}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

where σ_{sm} is the activation function and z_i is the input where $i \in (1, 2, 3, \dots, K)$ and K is the total number of data points.

Loss Function:

Loss functions are used in determining the difference between the computed value and the target value. Using the loss function, the model is able to back-propagate the loss and adjust the values such that they are closer to the target value. Common loss functions are the mean squared error and cross-entropy. They are explained below:

Mean Squared Error Loss: Mean squared error loss (MSE) is calculated by taking the squares of the differences between the actual and computed values. The output will be positive at all times. It is usually used in regression-type algorithms. The equation is given below:

$$MSE = \frac{1}{D} \sum_{i=1}^D (y_i - \hat{y}_i)^2$$

where D is the number of output classes, \hat{y}_i is predicted class values and y_i is the actual class values.

Cross-Entropy Loss: The cross-entropy (CE) loss function (logarithm loss) is mostly used for classification between multiple output classes. It works by penalizing the probability difference between the target and computed values. It is also called logarithmic loss because the function uses a logarithmic approach to calculate the difference which gives appropriate results for classification.

$$CE = - \sum_{i=1}^M y_i \log(\hat{y}_i)$$

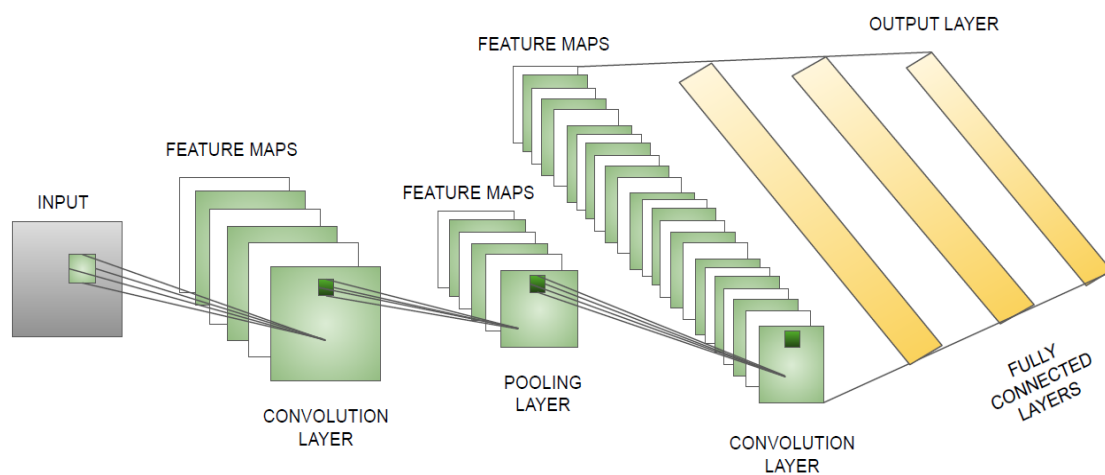
Where M is the number of classes, \hat{y}_i is predicted class values and y_i is the actual class values.

4.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) fall under the domain of deep learning algorithms that are designed to process grid-like data such as images. The application of CNNs is wide-ranging from image recognition to object detection and facial recognition. Some other applications are non-image related such as sequential data from which spatial features and patterns can be

extracted. This can be done after performing dimensionality reduction techniques such as kernel PCA.

Each network consists of multiple layers just like an artificial neural network and they can be divided into linear and non-linear layers. The linear layer consists of linear transformations to reduce dimensionality and the non-linear layer is used to introduce non-linearity in the network to prevent linear decision making. These linear layers are the convolutional layer, pooling layers and fully connected layers while the non-linear layer uses an activation function such as sigmoid or Rectified Linear Unit (ReLU). The functional breakdown of each layer is given below and depicted by figure 1.6:



CONVOLUTIONAL NEURAL NETWORK

Figure 1.6 Convolutional neural network

Convolutional layer: This layer is the most important layer and it distinguishes the CNN from an average ANN. It is responsible for detecting the local features from input data which could be edges or textures. Using a feature map, this layer convolves the input into a concise matrix. The feature maps are called filters and they are learned through the training process of the CNN.

By doing this the CNN is able to learn about particular data and its features to provide better inference services. Figure 1.7 Depicts the convolutional layer

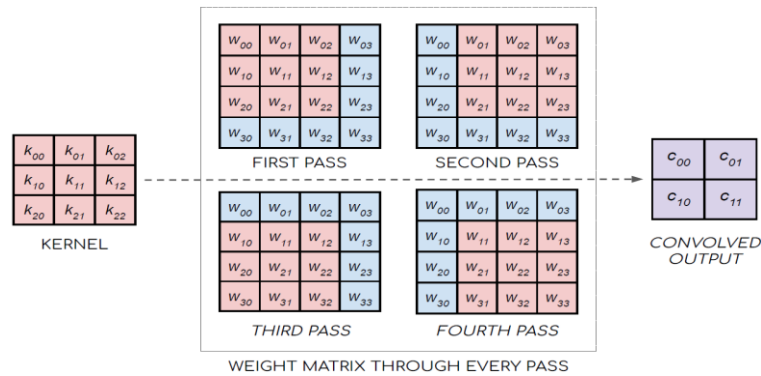


Figure 1.7 Convolutional layer

Pooling layer: The pooling layer always comes after the convolutional layer and they help in reducing the dimensionality of the convolved matrix. By reducing the spatial dimension of the matrix, the computation is more efficient and the features are summarized. There are various pooling techniques such as average pooling or max pooling. Average pooling picks the average of the pixel values whereas max pooling picks the maximum pixel values as depicted in figure 1.8.

Activation function: The activation function layer is applied after each linear layer. By introducing non-linearity in the network, more complex patterns and features can be represented. ReLU replaces all negative values with 0 while sigmoid places the value between a range of 0 and 1.

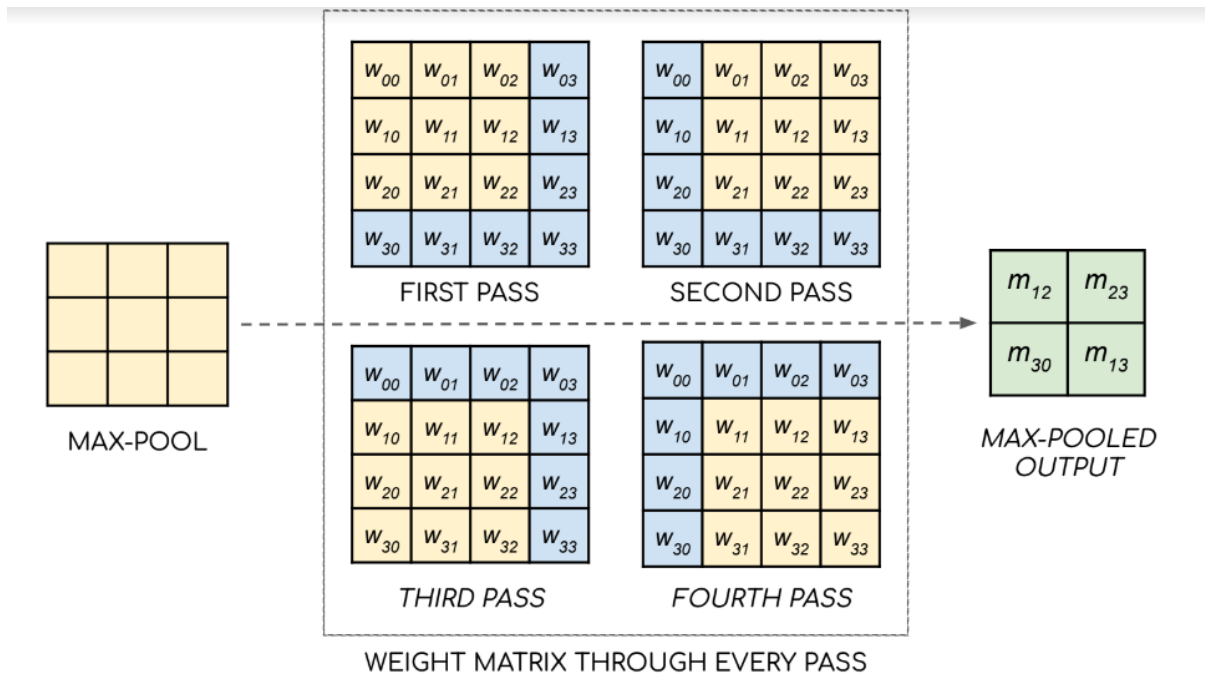


Figure 1.8 Pooling layer

Fully connected layer: After passing through all the previous layers, the fully connected layer brings it all together. The final convolved matrix is flattened using a transformation process called flattening. This is then passed through multiple fully connected layers where each and every neuron is connected in each layer. Then it passes through a non-linear layer and the final output layer consisting of the computed values is passed through the softmax function to obtain the probabilities for prediction. The figure 1.9 below describes the flattening.

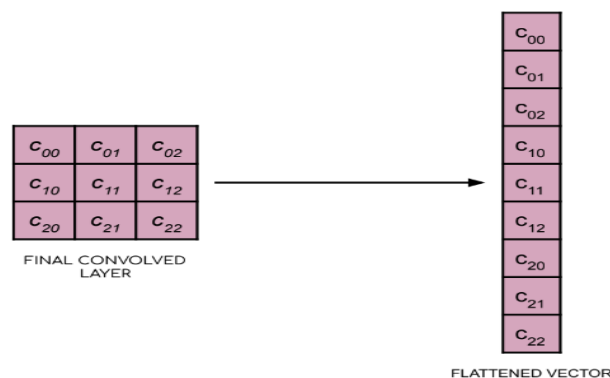


Figure 1.9 Flattened Vector

After the whole process is complete, it goes through the regular process of training and inference wherein multiple rounds of training improve the performance of the neural network and can finally be deployed to perform inference services.

While CNNs are traditionally used for tasks related to image data, they can also be effectively used for any type of data by slightly modifying the CNN. In one-dimensional CNNs each filter searches for specific patterns in the data that are correlated to each other. The key is to have the data points with some form of local correlation where nearby data points are more related to each other than data points that are further away. Just like with image data where the filter recognizes edges and other features of the image so too with one dimensional data can the filter detect a sudden spike in time series data or a specific sequence of events allowing us to utilize the same pattern recognition but for one dimensional data and solve classification problems. This shows us how versatile CNNs can be, which makes them suited for a wide range of tasks. The CNN that has been designed is a one-dimensional CNN unlike the two-dimensional ones used for image datasets, to find patterns in the Cleveland heart disease dataset, the hyperparameters were tuned to have 64 filters and a kernel size of 3. The filters indicate that they slide and convolve to transform the data, producing a feature map making it easier network to understand the data that has been provided as input to it and the kernel size 3 means each window of the filter will consider 3 contiguous elements at the same time the pooling and flattening layers work the same as described above for general CNNs while the dense layer is the one that performs classification on the features extracted using convolution, in this model there are two dense layers the first layer with 50 neurons and the second with a single neuron which is the standard amount when performing binary classification on a dataset. For the activation function the CNN uses Rectified Linear Unit (ReLU) which is defined in the above sections and is currently the most popular activation function used due to its efficiency in

helping models learn from their errors and make adjustments which is essential for a complex thinking structure.

4.2.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are also a type of artificial neural network that handle or process sequential data such as time series data. Their common uses are in problems that are ordinal or temporal such as language translation, natural language processing (NLP), speech recognition and image captioning. Any task that requires understanding context or relationships between segmented data can be handled using RNN for which they are well-suited.

RNNs differ from CNNs and ANNs in their ability to maintain internal memory. By using this internal memory, RNNs are able to retain important information about the input they are given, using which they are able to make accurate and precise predictions based on context. The nodes in RNN are connected in a cycle-like structure wherein the output of some nodes affects the subsequent input to the same nodes.

The structure of an RNN consists of the input, hidden and output layers just like an ANN. The hidden layer in the case of RNNs contains recurrent connections and maintains the internal memory. At each step, the RNN processes the current input and combines it with the information from the previous step which results in short-term memory. By using this technique, RNNs improve the overall output. Hence, only one hidden layer is looped multiple times without the need for multiple hidden layers.

A basic RNN cell is the core of an RNN it is designed as follows. It starts by taking an input vector X_t at each time step t and produces an output vector Y_t . The RNN cell also has a hidden state H_t which is updated at each time step. This hidden state is what provides an RNN with its special “memory” feature that makes it so different from other artificial neural networks. The equation for how this works is described below:

$$H_t = \tanh(W_{hh} \cdot H_{t-1} + W_{xh} \cdot X_t + b_h)$$

$$Y_t = W_{hy} \cdot H_t + b_y$$

Where W_{hh} , W_{xh} and W_{hy} are weighted matrices and b_h and b_y are the bias vectors. The function \tanh is the hyperbolic tangent activation function.

The RNN used is a simple RNN with three layers, two of them being simple RNN layers consisting of fifty neurons each and the last layer being a dense layer consisting of 1 neuron and a sigmoid activation function.

RNN's just like any other neural network aren't perfect though, they do have provide us with limitations creating some challenges. Some of the challenges faced when using an RNN are Vanishing gradients where during back propagation the gradients slowly disappear especially in long sequences. Memory limitation is another problem that RNNs have which means it is difficult for them to capture long range dependencies in data.

RNNs come in a few varieties, one being the Long Short-Term Memory Model (LSTM) which has been used in this project and Gated Recurrent Units (GRU) which is a simplified form of an LSTM.

Long Short-Term Memory Model: LSTM is a special type of RNN as stated above, it was created by Hochreiter and Schmidhuber in 1997 to prevent the problems of a regular RNN which is the vanishing gradient and exploding gradient problems as stated above in the challenges faced RNNs section, which make it difficult to train long sequences. This also makes the task computationally inefficient for large-scale problems. LSTMs are highly efficient in sequence modeling tasks such as machine translation, text summarization, and time series prediction. An LSTM similar to an RNN has memory but it also has other features that provide

a longer memory than the traditional RNN the breakdown of an LSTM model as displayed in Figure 1.10 is as follows,

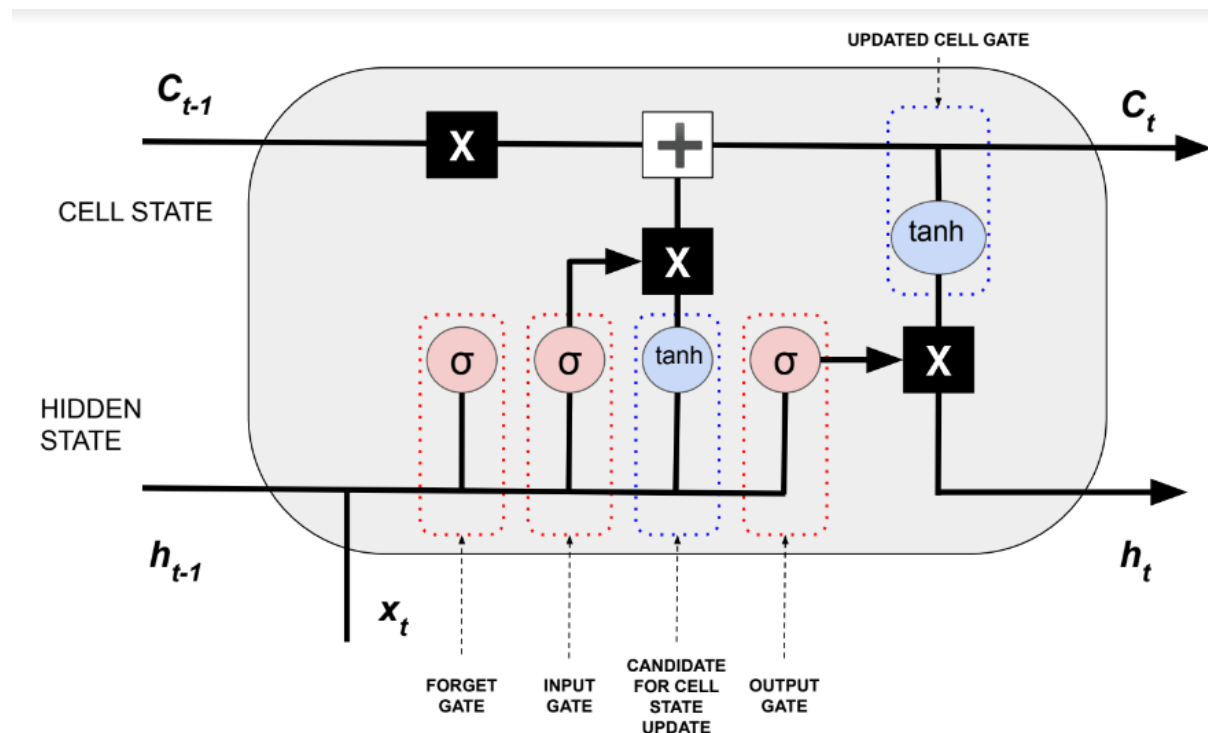


Figure 1.10 LSTM model

it comprises of 4 main parts forget gate, input gate, cell state and the output gate these 4 parts work in unison to provide the model to make the LSTM model more efficient in handling long sequences of data. The forget gate as the name implies is to forget unnecessary data from the cell state which is the core part of the model and help keep the model from slowing down, mathematically the forget gate can be defined as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Where x_t is the input, σ is the sigmoid activation function and the weights W and biases b are learned during training.

The next part is the input gate, this controls what new information is passed and stored in the cell state and can be mathematically defined as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Where σ is the sigmoid activation function, \tanh is the hyperbolic tangent activation function and the weights W and biases b are learned during training.

The cell state is the core part of the LSTM and is responsible for what information is carried over a time period the update function is the part of the cell state that is responsible for carrying over the information which can be mathematically defined as:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Where C_{t-1} is the previous cell state and C_t is the current cell state.

Finally, there is the output gate which controls the state of the output based on the cell state and the input. The mathematical formula for the output gate's control mechanism is defined as follows:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Where σ is the sigmoid activation function, \tanh is the hyperbolic tangent activation function, the weights W and biases b are learned during training and C_t is the cell state.

LSTMs help us tackle the problems faced by the RNN model allowing us to deal with long term data dependencies and providing us flexible memory. LSTM models while being a

specialized RNN itself also has its variations namely it has 4 variations Unidirectional LSTMs, Bidirectional LSTMs, Stacked LSTMs, and Attention Mechanisms. For the purposes of this project, unidirectional model was chosen since the other models don't seem to be well suited for the task. A Unidirectional model the model processes information from the past to predict the future or understand the present it takes information only from one direction which is the starting point and passes it in sequence through the layers to the end. In a bi-directional model however, the sequence is processed in both directions from the past to the future and from the future to the past this is accomplished by having to separate LSTM models one which processes information like the unidirectional model (left to right) which does past to future processing and the other that does the reverse (right to left) which is for the future to past processing. The next variation is the stacked LSTM which is just LSTM cells stacked one on top of the other for higher levels of abstraction and the final variant is the attention mechanism which is like an attachment to the LSTM model which weighs how important different parts of the input are compared to each other.

Training an RNN model: while training any RNN model be it a simple RNN or an LSTM model, the Optimizers, Regularization and Dropout, Loss Functions, Backpropagation Through Time (BPTT), Learning Rate and Batch Size, Sequence Length are considered a very crucial part. The multitude of hyperparameters that need to be set and optimized make training and implementing RNNs a little more complex than other ANNs. There are a range of optimizers to choose from but for the model from which the adam optimizer was chosen. Regularization is a technique used to prevent overfitting in a model overfitting is a problem that occurs when a model thinks it has learnt well but gives us the wrong output when applied to the actual data due to its training data including noise, outliers and a multitude of other factors which the model is then trained on. Regularization works by adding penalties to the different parameters of the model, reducing the freedom of the model making it harder for noise

and other unnecessary elements to be part of the training data input to the model due to their high variance.

Regularization in the context of LSTMs usually takes a couple of forms, namely L1 and L2 regularization.

L1 regularization is the method chosen for our model and it works by adding a penalty to the loss function based on the absolute value of the weights provided to the model. Mathematically written as follows:

$$L' = L + \lambda \sum |w|$$

Where L is the Loss function, λ is the regularization strength, w is the weights and L' is the new loss function with the applied regularization. In our model the L1 regularization is applied to the third dense layer with a coefficient of 0.001, where 0.001 is value of λ , indicating the regularization strength.

L2 regularization is the other type of regularization technique that can be applied to an LSTM and it works very similarly to the L1 method with the difference being that in the L2 method instead of adding a penalty based on the absolute value of the weights provided we do it based on the square of the value of the weights and it is mathematically written as

$$L' = L + \lambda \sum w^2$$

Where L is the Loss function, λ is the regularization strength, w is the weights and L' is the new loss function with the applied regularization. These formulae can be applied to the input, output and recurrent weights of an LSTM model.

Next, we move on to dropout which is another type of regularization technique. Dropout works by setting a random fraction of the inputs sent into the LSTM model to 0 during each update step which helps prevent overfitting and what makes it so great is its flexibility which allows it to be applied to any layer of the LSTM model and have been effective. In our model the dropout has been set to 0.2 which means a random 20% of the input will be set to 0.

Back propagation through time (BPTT) is another a special extension of the traditional back propagation usually used in RNNs, it is used to update the model's parameters to minimize the loss. BPTTs usually come into play in the third stage of the five stages of running a model. The first stage is forward pass where the model processes inputs one timestep at a time while also maintaining its memory so as to capture temporal patterns in the sequence, using this hidden state memory, the model's cell state creates an output for each step. The next stage is the compute loss stage where we compute the loss once the entire sequence has been processed. The loss serves as a measure as to how well the model performs. The third step is backward pass, here the algorithm computes the loss function's gradient with respect to each model parameter this is the phase where BPTT comes into play, what BPTT does here is identifies how much each parameter contributed to the loss by iterating through each time step and calculating it. The fourth stage is gradient clipping which is a technique used to mitigate any exploding or vanishing gradient problem which may occur, while LSTMs help reduce this, they are still not immune to it hence this technique is implemented to prevent it completely. It works by setting a gradient and scaling the model parameters that do not fit the gradient before they are updated. The final step is parameter updating here the model parameters are simply updated using an optimization algorithm like adam.

After this the learning rate is set which is a hyperparameter mean for optimization and it fine tunes the learning rate of a model who help with efficiency and to increase the precision

of the model and finally we have the sequence length that helps us make the training more feasible and prevent overfitting although this depends on the problem at hand and its context.

4.2.4 Training an Artificial Neural Network

A key task in training a Neural Network is to compute the gradient on each input in the dataset.

We now summarize the main steps to compute the gradient for a data point (x, y) .

Forward Propagation: In the forward propagation phase, the input data is passed through linear and non-linear layers to learn about its features and store these values in its neurons (also known as weight matrices) to use later in the backward propagation phase. In the linear layer, given an input vector x (with $a_0 = x$), the neural network learns the feature of this input data by running it through a linear transformation with the weight matrices W_i and bias vector b_i corresponding to that particular hidden layer. For the i^{th} layer with $0 \leq i \leq \ell - 1$, the linear layer computation is given by:

$$c_i = W^i \cdot a_i + b^i \text{ with } a_0 = x$$

and the non-linear layer computation is:

$$a_{i+1} = \sigma(c_i)$$

where σ is the activation function, and in the last layer it is the softmax function. At the final layer, the output is $\hat{y}_i = a_\ell$.

Backward Propagation: In the backward propagation phase, the stored weight matrices are updated by calculating the error in the final output layer \hat{y} using a loss function and back-propagating the loss through the entire network in reverse order. This phase also consists of the linear layer and non-linear layers where the linear layer computes the error γ_i using the value computed by the loss function in a linear transformation. The non-linear layer α_i , computes the

error using the derivative of the respective activation function used in the forward propagation phase. In the backward propagation, for the i^{th} linear layer computation is:

$$\gamma_{\ell-1-i} = (\mathbf{W}^{\ell-1-i})^T \alpha_{\ell-1-i}, 0 \leq i \leq \ell - 2,$$

Where $(W_i)^T$ is the transpose of the weight matrix W_i and $\alpha^2 = a^3 - y$. The non-linear layer computation is:

$$\alpha_{i-1} = \sigma'(c_{i-1}) \odot \gamma_i$$

where σ' is the derivative of σ and \odot is the component-wise multiplication or the Hadamard product of two vectors.

Gradient Computation: Using the forward and backward propagation computations, the gradient for the weight matrix and bias vector is computed as $\nabla b^{\ell-1-i} = \alpha \alpha_{\ell-1-i}$ and

$\nabla W^{\ell-1-i} = \alpha_{\ell-1-i} \cdot a_{\ell-1-i}^T$ for $0 \leq i \leq \ell - 1$. Depending upon different gradient algorithms, a gradient on the dataset is computed to iteratively update the model. Three well-known gradient descent algorithms are mentioned below. Figure 3.4. Consider a three-layer ANN where the model parameters are denoted as $\theta = \{(W^0, b^0), (W^1, b^1), (W^2, b^2)\}$ Given an input x , the gradient computation is given in Figure 1.11 where ∇W_i and ∇b_i denote the gradient for W_i and b_i , respectively, for $0 \leq i \leq 2$ and $a^3 = \hat{y}$.

Forward propagation	Backward Propagation	Gradient
$c_0 = \mathbf{W}^0 \mathbf{x} + \mathbf{b}^0$ $\mathbf{a}_1 = \sigma(c_0)$	$\alpha_2 = \mathbf{a}_3 - y$	$\nabla \mathbf{W}^2 \leftarrow \alpha_2 \cdot \mathbf{a}_2^T$ $\nabla \mathbf{b}^2 \leftarrow \alpha_2$
$c_1 = \mathbf{W}^1 \mathbf{a}_1 + \mathbf{b}^1$ $\mathbf{a}_2 = \sigma(c_1)$	$\gamma_2 = (\mathbf{W}^2)^T \alpha_2$ $\alpha_1 = \sigma'(c_1) \odot \gamma_2$	$\nabla \mathbf{W}^1 \leftarrow \alpha_1 \cdot \mathbf{a}_1^T$ $\nabla \mathbf{b}^1 \leftarrow \alpha_1$
$c_2 = \mathbf{W}^2 \mathbf{a}_2 + \mathbf{b}^2$ $\mathbf{a}_3 = \sigma(c_2)$	$\gamma_1 = (\mathbf{W}^1)^T \alpha_1$ $\alpha_0 = \sigma'(c_0) \odot \gamma_1$	$\nabla \mathbf{W}^0 \leftarrow \alpha_0 \cdot \mathbf{a}_0^T$ $\nabla \mathbf{b}^0 \leftarrow \alpha_0$

Figure 1.11 Gradient Computation for a Two-layer ANN for an Input (x, y)

Gradient Descent Algorithms: The gradient descent (GD) algorithm is used to update the weights using the error computed by the backpropagation phase. There are different types of algorithms such as batch gradient descent (BGD), stochastic gradient descent (SGD), and mini-batch gradient descent (MBGD). These optimization algorithms are used to find the local minima and minimize loss functions to obtain accurate models.

Batch Gradient Descent: Batch gradient considers an entire batch of data samples and updates the model only after it minimizes the loss function for all the samples as elaborated in.

$$\theta = \theta - \alpha \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} J_i(\theta)$$

where α is the learning rate, m is the number of training examples and $J_i(\theta)$ is the loss function for i .

Stochastic Gradient Descent: Stochastic gradient descent considers random samples from the input and minimizes the loss function. It updates the model for every sample until all samples are covered. It can be represented by:

$$\theta = \theta - \alpha \nabla_{\theta} J_i(\theta)$$

where α is the learning rate, m is the number of training examples and $J_i(\theta)$ is the loss function for i .

Mini-batch Gradient Descent: Mini-batch gradient descent divides the data samples into small batches and minimizes the loss function. It updates the model for each batch until all samples are covered. This is represented by:

$$\theta = \theta - \alpha \nabla_{\theta} J_b(\theta)$$

where α is the learning rate, θ are the model parameters, $J_b(\theta)$ is the loss function computed on a mini-batch of size b and $\nabla_{\theta} J_i(\theta)$ is the gradient with respect to θ . In summary, the batch gradient descent processes all training data at once while stochastic gradient descent processes one example at a time. Mini-batch gradient descent processes small batches of data at once which reduces noise compared to SGD while still being computationally efficient.

This entire process is conducted for a given number of epochs and iterations until satisfactory accuracy is obtained and the model is ready for use.

4.2.5 Approximation functions

Activation functions and loss functions in the non-linear layers of a neural network tend to deal with exponentiation and mathematical operations. Although this might cause a reduction in the accuracy, the reduction is relatively minor if the function is approximated appropriately. In this

project, the sigmoid and ReLU (Rectified Linear Unit) activation functions have been considered majority of the time.

Sigmoid Approximation: We consider the following sigmoid approximation function:

$$\sigma_{\text{apx}}(x) = f(x) = \begin{cases} 0 & x < -\frac{1}{2} \\ x + \frac{1}{2} & -\frac{1}{2} \leq x \leq \frac{1}{2} \\ 1 & x > \frac{1}{2}. \end{cases}$$

The derivative of the above approximated sigmoid function is:

$$\sigma'_{\text{apx}}(x) = f'(x) = \begin{cases} 1 & -\frac{1}{2} \leq x \leq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

Softmax Approximation: The softmax approximation function while not exactly an approximation function can be used as such. We consider the following softmax approximation function:

$$\sigma_{\text{softmax}}(x_i) = f(x_i) = \frac{1 + x_i + 0.5x_i^2}{\sum_j^K 1 + x_j + 0.5x_j^2}$$

for $i = 1, \dots, K$ and $\mathbf{x} = (x_1, \dots, x_k) \in R^k$ and the Taylor series of e^{x_i} is given by:

$$e^{x_i} = 1 + x_i + 0.5x_i^2$$

ReLU (Rectified Linear Unit): We have already defined ReLU in the previous activation function phase, but it can also be used as an approximation function.

4.3 Procedure

4.3.1 Background

After exploring different methodologies like the “**Cross Industry Standard Process for Data Mining (CRISP-DM)**” model or the “**Knowledge Discovery in Databases (KDD)**” Process, it was decided that the best data science project model for this project would be “**Sample, Explore, Modify, Model, and Access (SEMMA)**” model. The **SEMMA** model follows a similar pathway to the **CRISP-DM** model in most parts with the exception of the business understanding and deployment phase since this project focuses more on the data, data pre-processing, modelling and evaluation phase and less on the business understanding and deployment since it is not a commercial project.

The **SEMMA** model consists of 5 main steps Sample, Explore, Modify, Model and Assess. In the Sample phase, required dataset is selected from the multitude of datasets available to us or generate the required dataset via experimentation and group studies, while also trying and identify the conditional or dependent and autonomous or independent features that influence the modelling process which is critical in us performing any type of analysis or prediction. The goal of the sample phase is to select a dataset and capture the essential characteristics without having to work with the whole dataset. The next phase is the Explore, we do as the name suggests in this phase and explore the selected dataset to see what type of values the data contains, if the data needs any special operations performed on it such as scaling, checking to see if data needs to be converted from ordinal to nominal data, while also look for missing values, and analyzing the relation between the different variables and how they correlate to the target variable. The end goal of the explore phase is to analyze and record the observations for all the features that could impact the outcome. We then move on to the Modify phase where the data pre-processing is performed on the selected dataset from the exploration

phase, knowing what processes need to be applied to the dataset, be it data transformation, missing value imputation, data balancing or any other data pre-processing methods. After deciding on all the data pre-processing methods that need to be applied, Implementation is then performed and applied to the dataset, A check is then performed on the data to ensure the quality meets the project's requirement for modelling and if the data doesn't meet the standard, the explore phase is checked again to choose the data pre-processing techniques required to make sure it meets standards. The Model phase comes after the Modify phase, here the appropriate model required to solve the problem is chose after which they are implemented and the processed dataset is run through them. Finally, the assess phase is reached where the main goal is to evaluate the results from the chosen models and choose the models with the best result. The results are also analyzed using different methods such as accuracy, precision, recall, F1 score, mean square error and a few others to make sure the results don't have problems such as overfitting which exaggerates measures such as accuracy or precision. If problems like that do occur, we go back to either the modelling or exploration phase to check what the issue might be and fix it.

In conclusion, the **SEMMA** model helps give the project a direction to proceed in and helps figure out the steps to follow when working on a large project with different moving parts but isn't necessarily meant to up to industry standards since this project doesn't need a business understanding or a deployment phase.

4.3.2 Data pre-processing

Going by the phases of the SEMMA model, the heart disease dataset was chosen and from it was chosen the Cleaveland Heart disease data but the data was not in a usable format hence the data transformed to suit our requirements. To do this several steps were followed, the first of these several steps was to change its format so it was malleable to work with. The data initially

came in a .data file separated only by space so first the is stored from the .data file in a .txt or text file which meant it was easier to manipulate. The end goal of this first data cleaning phase was to convert the all the data in the .data file into a csv file with rows and columns to which the standard data preprocessing techniques could be applied.

Since the data was separated by only spaces it meant there were no new lines meaning all the data was simply considered a single row when converting to a csv file which had no value to us, hence a distinguishing factor needed to be found by which we could split it. Analyzing the data, it was observed that the other values of the file were all numeric while the last column was ordinal data which meant that could act as a unique identifier. Using the “last name” column the trailing and tailing white spaces were removed and a new line added after each name. Through this process the data was split into different rows but seeing as to how the data was space separated and not comma separated, it was decided to replace all the spaces with commas which gave us data that looked like the data in figure 3.5

```
1,15943882,63,1,-9,-9,-9-27,1,145,1,233,-9,50,201,0,1,2,2,3,1981,00,0,0,0,1,10.5,6,13150,60,190,90,145,85,0,02.3,3,-9,-9,0,-9,-9,-9,-9,-9,6,-9,-9,216,1981,0,1,1,1,
-9,1,-9,1,-9,1,1,1,1,11,1,-9,-9,0,-9,-9,-9,-9,-9,-9,-9,-9,0,00,0,name
2,15964847,67,1,-9,-9,-9-27,4,160,1,286,-9,40,400,0,1,2,3,5,1981,01,0,0,0,1,9.5,6,13108,64,160,90,160,90,1,01.5,2,-9,-9,3,-9,-9,-9,-9,-9,3,-9,-9,25,1981,2,1,2,2,
-9,2-9,1,-9,1,1,1,1,11,1,-9,-9,0,-9,-9,-9,-9,-9,-9,-9,-9,0,00,0,name
3,15952199,67,1,-9,-9,-9-27,4,120,1,229,-9,20,350,0,1,2,2,19,1981,01,0,0,0,1,8.5,6,10129,78,140,80,120,80,1,02.6,2,-9,-9,2,-9,-9,-9,-9,-9,7,-9,-9,220,1981,1,1,1,1,
-9,1,-9,1,-9,2,2,1,1,17,3,-9,-9,0,-9,-9,-9,-9,-9,-9,-9,-9,0,00,0,name
4,15929464,37,1,-9,-9,-9-27,3,130,0,250,-9,0,00,0,1,0,2,13,1981,01,0,0,0,1,13,13,17187,84,195,68,130,78,0,03.5,3,-9,-9,0,-9,-9,-9,-9,-9,3,-9,-9,24,1981,0,1,1,1,-9,
1,-9,1,-9,1,1,1,1,11,1,-9,-9,0,-9,-9,-9,-9,-9,-9,-9,-9,0,00,0,name
```

Figure 2.0 Data in text file after replacing spaces

This while appears good enough to use was still not clean because it still contained unnecessary new lines. Using a few regular expressions to further remove the new lines and format the data, the required format was partially reached which was still far from clean but was good enough to convert to a csv file. The result as shown in Figure 2.1 was a dataset with 1540 rows and 90 columns.

	1	15943882	63	1.1	-9	-9.1	-9.2	-27	1.2	145	...	-9.24	-9.25	-9.26	-9.27	-9.28	0.11	0.12	0.13	0.14	name
0	2	15964847	67	1	-9	-9	-9	-27	4	160	...	-9	-9	-9	-9	-9	0	0	0	0	name
1	3	15952199	67	1	-9	-9	-9	-27	4	120	...	-9	-9	-9	-9	-9	0	0	0	0	name
2	4	15929464	37	1	-9	-9	-9	-27	3	130	...	-9	-9	-9	-9	-9	0	0	0	0	name
3	6	11961207	41	0	-9	-9	-9	-27	2	130	...	-9	-9	-9	-9	-9	0	0	0	0	name
4	7	15970464	56	1	-9	-9	-9	-27	2	120	...	-9	-9	-9	-9	-9	0	0	0	0	name
...
1535	1666	24309	47	0	1	-9	-9	-9	2	110	...	-9	-9	-9	-9	-9	-9	-9	-9	-9	Papp
1536	1667	22803	61	0	1	-9	-9	-9	1	120	...	-9	-9	-9	-9	-9	-9	-9	-9	-9	Friederi
1537	1668	13203	57	1	1	-9	-9	-9	4	130	...	-9	-9	-9	-9	-9	-9	-9	-9	-9	Torok
1538	1669	14205	47	1	1	-9	-9	-9	4	100	...	-9	-9	-9	-9	-9	-9	-9	-9	-9	Veszelsz
1539	1670	15403	36	1	1	-9	-9	-9	3	120	...	-9	-9	-9	-9	-9	-9	-9	-9	-9	Takacs

Figure 2.1 Dataset after converting to csv

The headers were manually added while removing the unnecessary columns which included fourteen dummy columns which simply had the placeholder value '-9'. Finally, the resultant dataset obtained had 1540 rows and 76 columns which met the requirement to begin applying the data pre-processing techniques. As for the data pre-processing techniques, first all the columns that were unnamed, junk variables and variables that weren't explained were dropped. After which the placeholder value '-9' were replaced with Na values and a missing value check was performed, replacing the missing values with the median since it seemed to have the best outcome compared to replacing the values with the median or mode. The next step was to check the distribution of the target variable 'num' which was unbalance. To resolve this, the distribution was checked where two extra values 3 and 4 were found which weren't mentioned in the dataset so anything that wasn't 1 or 0 were replaced with the required value using a lambda function.

The next step was scaling which was performed using the min-max scaler. Scaling is the process of normalizing the feature variables to help with data pre-processing, the main focus of scaling is to make sure the data fits into a certain range so the data trains the model well without noise or outliers. Mathematically min-max scaling can be written as follows:

$$\text{Scaled value} = \frac{\text{Original value} - \text{Minimum value}}{\text{Maximum value} - \text{Minimum value}}$$

This transforms the data to fit within the range of typically (0,1) or (-1,1) depending on the values that are required.

Now the dataset is split into train and test sets, for this project, it has been split the into a 4:1 ratio where the training dataset is 80% of the scaled dataset randomly selected and the test dataset is 20% of the scaled dataset again randomly selected, the stratify hyperparameter was also set to ensure the target variable has balanced class distribution concluding the data pre-processing step.

4.3.3 Feature selection

Feature selection is a very important step in any type of problem as selecting the wrong features like features that have a high correlation with the target variable causing the model to use that as the main point of solving the problem leading to the model to giving us an undesired outcome. Feature selection can be done in two possible ways the first is to do it manually by checking the correlation coefficient of each variable and seeing how they correlate with the target variable, removing those that are highly correlated. The other way is to allow an algorithm to do the feature selection for us. While there are multiple algorithms to choose from, it was decided that the chi-squared test was the best in terms of time, efficiency and output provided by the model.

The chi-squared test works by measuring the dependance between stochastic variables and weeds out the irrelevant variables keeping only the variables relevant to the classification problem at hand. The features selected by the algorithm were 'sex', 'painexer', 'cp', 'smoke', 'fbs', 'dm', 'prop', 'nitr', 'pro', 'proto', 'thalach', 'exang', 'slope', 'ca'.

4.3.4 Modelling

In the modelling stage the required models built and implement the models that are necessary for the task at hand. For the requirements of this project, ANN's such as feed forward neural networks, convolutional neural networks (CNN), recurrent neural networks (RNN) and the long short-term memory model (LSTM) to see how they fare against the standard machine learning models implemented in the other literatures discussed in the Key Literature chapter.

First let's begin with the LSTM model: the model implemented is a unidirectional stacked LSTM model with two LSTM Layers, two dropout layers and two dense layers, the model also has L1 regularization with a set dropout rate to prevent any sort of overfitting from occurring. The first layer is a LSTM layer with fifty neurons and return sequences set making it so each hidden state output is returned for each time step after this layer a dropout layer of 0.2 or 20%, the next layer is also an LSTM with fifty neurons but here the return sequences are turned off so it now becomes a sequence input and single output prediction. After which another dropout layer of 20% is set. Next, we have a dense layer with ten neurons and an L1 regularizer set to a strength on 0.001 using the ReLU activation function, and we have the final layer which is a dense layer with one neuron using the sigmoid activation function. This model is then compiled using the adam optimizer and a binary cross entropy loss function. The next model is a CNN which has five layers, the first layer is a one-dimensional convolutional layer that has 64 filters, a kernel size of 3 and uses ReLU as its activation function, the second layer is a pooling layer with a pool size of 2 which indicates maximum value over a window of size 2, the third layer is a flattening and the fourth and fifth layers are dense layers with the fourth layer having 50 neurons and uses a ReLU activation while the fifth is a dense layer with 1 neuron that uses the sigmoid activation function.

The third model is a simple feed-forward Artificial neural network with four dense layers the first layer has 128 neurons while the second has 64, the third has 32 and the last has one neuron. The first, second and third layers use the ReLU activation function and the fourth layer uses a sigmoid activation function. The fourth and final model is a simple RNN model with three layers, the first layer is an RNN with 50 neurons and return sequences set to true just like with the LSTM model, the second layer is the same and the third layer is dense layer with one neuron and uses a sigmoid activation function. All the models were compiled using an adam optimizer and a binary cross-entropy loss function. After the models are compiled, they are evaluated using their accuracy score, area under the curve (AUC) and loss.

4.5 Ethics

Ethics are especially important when it comes to anything related to the medical industry since it deals with human lives. The researchers that gathered the data for this dataset state that this data is ethically sourced and have listed the steps and procedures to make sure that their means of gathering the data ethically is documented in their paper. All the data collected for this dataset was done with the consent of the patients entering the trials according to the documentation, none of the patients seem to have had invasive procedures performed and the data on their medical history was also take with their consent, the tests they underwent were also performed only after their consent.

4.6 Data Analysis

From analyzing the dataset, the details obtained are used in tableau to create graphs and charts of these statistics.

First is the graph showing us the target variable distribution described in the figure 2.2

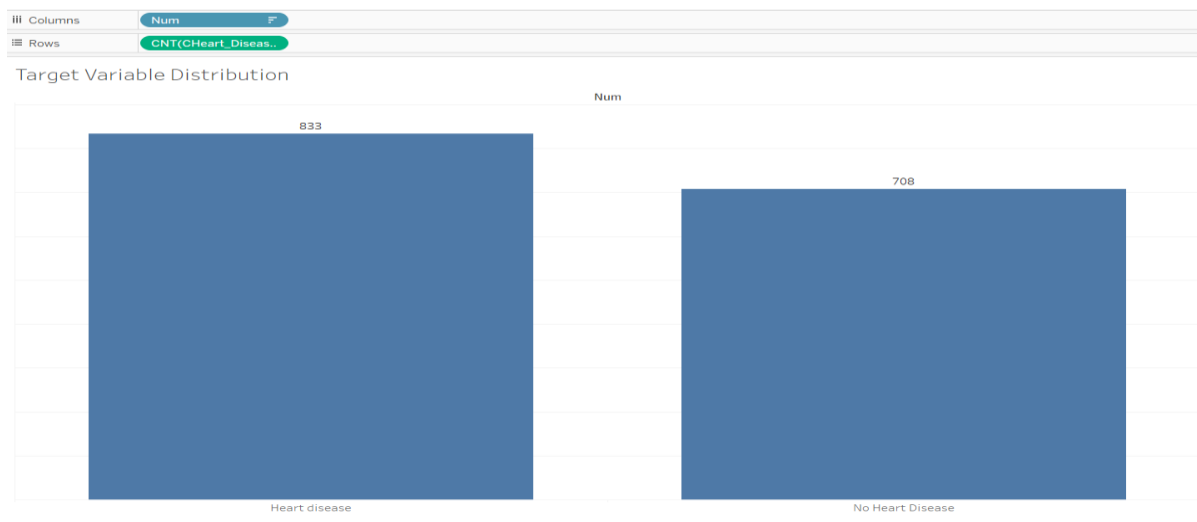


Figure 2.2 Target distribution of target variable num

Here we see that after the dataset is only slightly imbalanced making it ok to use for modelling. The data before balancing had a distribution of approximately 46% for class 0, 19% for class 1, 15% for class 3, 14% for class 2, and 5% for class 4 which showed us that the dataset was imbalanced.

Another analysis as described in figure 2.3 shows us the correlation between smoking and heart disease divided by sex.

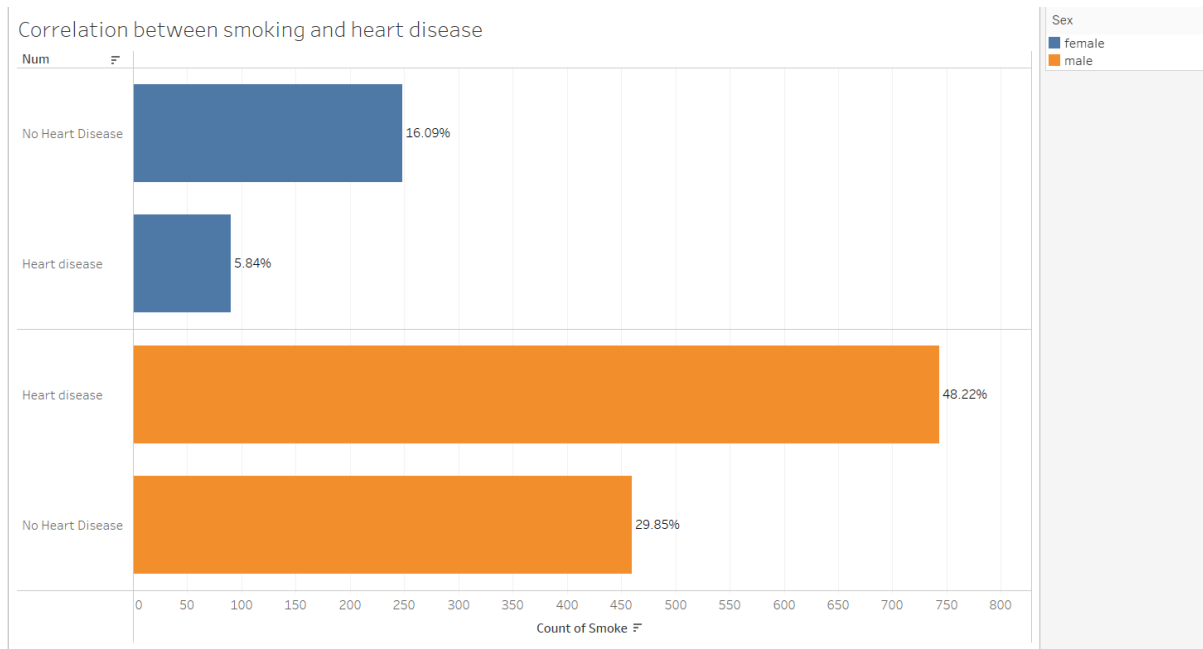


Figure 2.3 the correlation between smoking and heart disease based on sex

Here we see that of the total patients that participated the patient who were male and smoked seemed to be diagnosed with heart disease while the opposite was true for the females where the women who smoked don't seem to be diagnosed with heart disease but the women who didn't smoke seem to be diagnosed with heart disease.

CHAPTER 5

RESULT

5.1 Introduction

The results obtained from the experiment showed us that the Artificial neural networks did not fare as well as the traditional machine learning techniques implemented in the previous pieces of literature mentioned in the literature review section but at the same time the neural network models did not yield a very low accuracy either. An accuracy between 73 to 75 percent was obtained, an AUC score of 79 to 80 percent and a loss of between 51 to 60 percent. This could indicate that while the models may not have fared as well as traditional machine learning models the neural networks are still very flexible with the types of problems, they solve but there may also be room for improvement.

5.2 Descriptive statistics

Descriptive statistics refers to the statistical measures and tools used to summarize and describe the main features of a dataset. Here, summarized are the different statistic values of the selected features in the dataset.

Table 1.1: Descriptive statistics of the selected variables

	Mean	Standard deviation	Minimum	Maximum
sex	0.780662	0.413932	0.0	1.0
painexer	0.781311	0.413491	0.0	1.0
cp	3.158339	0.954608	1.0	4.0
smoke	0.119403	0.324367	0.0	1.0

fb	0.193381	0.395077	0.0	1.0
dm	0.205062	0.325128	0.0	1.0
prop	0.231019	0.421621	0.0	1.0
nitr	0.282284	0.450257	0.0	1.0
pro	0.120052	0.403878	0.0	1.0
proto	54.689812	49.084341	0.0	200.0
thalach	136.58209	23.17529	69.0	202.0
exang	0.303699	0.460003	0.0	1.0
slope	1.166775	0.802649	0.0	3.0
ca	0.135626	0.498474	0.0	3.0

The table 1.1 describes the statistics related to the variables chosen using the chi squared method. Observe here are the mean, standard deviation, the minimum and maximum values. These help us during the data preparation phase especially if data has missing values. These values are also the base values almost all machine learning and neural network algorithms are based on.

5.3 Inferential statistics

Figure 4.3 describes the heatmap showing us the correlation between different features

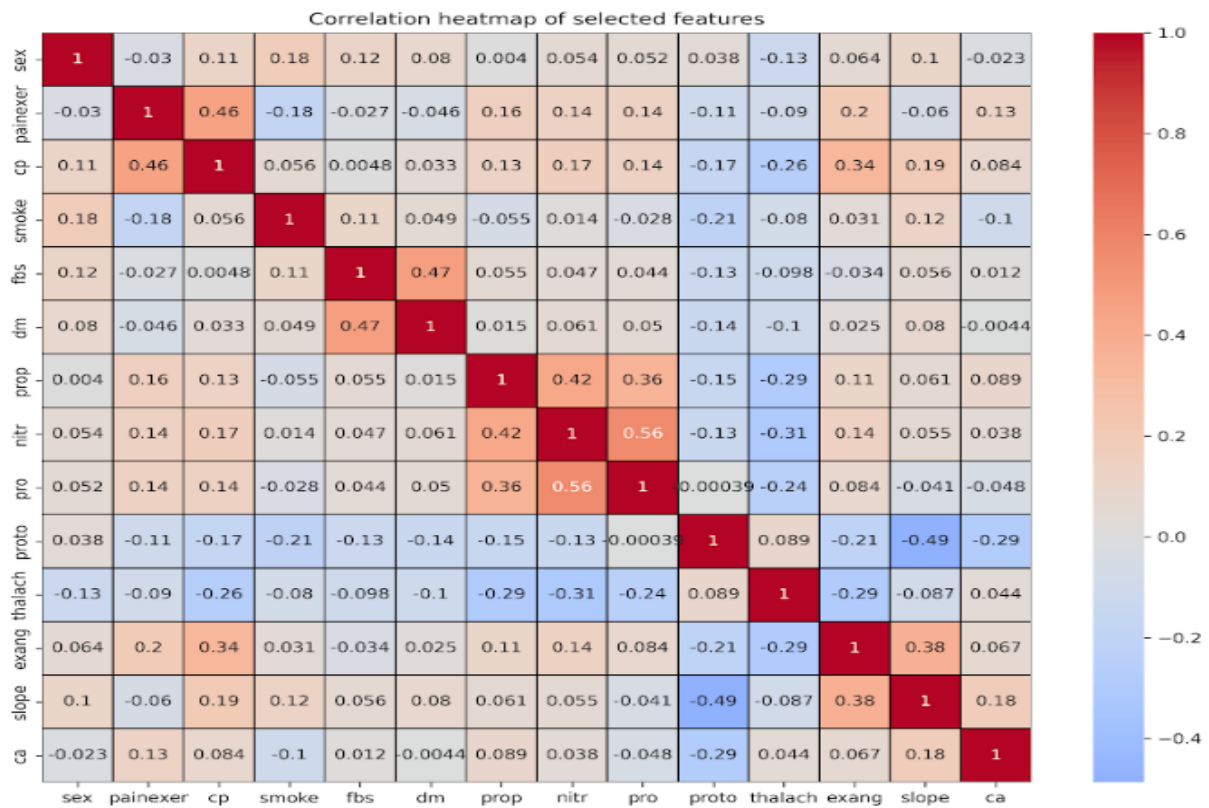


Figure 2.3 Heatmap showing correlation between independent and dependent variable

Here we can see the heatmap denoting the correlation between the variables and the target variable “num” these are the variables selected using the chi-squared test. Here we can see the variables all have quite a low correlation which is one of the reasons these variables were chosen by the algorithm.

CHAPTER 6

CONCLUSION

To conclude, the entire reason for perform this experiment was to do a comparative study between solving the Cleaveland Heart Disease classification problem using artificial neural networks and traditional machine learning techniques. The core part of the hypothesis was to compare the two algorithms to see which ones would come out on top. The neural networks fared fairly well with accuracies ranging from 70 to 76 percent which means they did not necessarily fare bad although the traditional models did come out on top with accuracies ranging up to 99 percent in (Ahmad, et al. 2022), although in some cases the neural networks did match up with some of the traditional models in terms of accuracy. The fact that the neural network models are slightly less accurate than the traditional models shows that there is room for improvement. These models are not normally used for these kinds of tasks but still fared fairly well displaying their flexibility and in the future, the accuracy may be improved using different feature selection techniques or using ensemble models of different neural networks combined together or even combining a traditional model with a neural network model.

REFERENCES

[1] Rani, P., Kumar, R., Ahmed, N.M.O.S., and Jain, A, A decision support system for heart disease prediction based upon machine learning, *Journal of Reliable Intelligent Environments* volume 7, 25 Jan 2021, pp. 263–275,

DOI: <https://doi.org/10.1007/s40860-021-00133-6>.

[2] Das, R., Turkoglu, I., Sengur, A., Effective diagnosis of heart disease through neural networks ensembles, *Expert Systems with Applications*, Volume 36, Issue 4, 2009, pp. 7675-7680,

DOI: <https://doi.org/10.1016/j.eswa.2008.09.013>.

[3] Olaniyi, E.O., Oyedotun, O.K. and Adnan, K.,. Heart Diseases Diagnosis Using Neural Networks Arbitration. *International Journal of Intelligent Systems and Applications (IJISA)*, 7(12), 2015, pp.75-82.

DOI: <https://doi.org/10.5815/ijisa.2015.12.08>.

[4] Ahmad, G. N., Fatima, H., Ullah, S., Salah Saidi, A., & Imdadullah. Efficient Medical Diagnosis of Human Heart Diseases Using Machine Learning Techniques With and Without GridSearchCV. *IEEE Access*, 10, 2022, 80151-80173.

DOI: <https://doi.org/10.1109/ACCESS.2022.3165792>.

[5] Mohan, S., Thirumalai, C., & Srivastava, G. (2019). Effective Heart Disease Prediction Using Hybrid Machine Learning Techniques. *IEEEAccess*, 7, 81542-81554.

[6] Al Ahdal, A., Rakhra, M., Rajendran, R.R., Arslan, F., Khder, M.A., Patel, B., Rajagopal, B.R., & Jain, R. (2023). Monitoring Cardiovascular Problems in Heart Patients Using Machine Learning. *Journal of Healthcare Engineering*, 2023.

[7] Detrano R, Janosi A, Steinbrunn W, Pfisterer M, Schmid JJ, Sandhu S, Guppy KH, Lee S, Froelicher V. International application of a new probability algorithm for the diagnosis of coronary artery disease. *Am J Cardiol*. 1989 Aug 1;64(5):304-10.

DOI: 10.1016/0002-9149(89)90524-9. PMID: 2756873.