|  |  |  |
| --- | --- | --- |
| University of Central Lancashire Logo | **School of Engineering** | **Coursework Cover Sheet**  ***MSc Project Dissertation*** |

*Students should add this coversheet to the start of their assessment before submission through Turnitin.*

| Seminar Tutor (If appropriate): Dr. Muqi Wulan | |
| --- | --- |
| Module Title: **MSc Project** | Course Title: MSc. Mechatronics and Intelligent Machines |
| Module Code: **EL4895** | Year of Study: **2021-2022** |

***Academic Misconduct / Plagiarism Declaration***

By attaching this front cover sheet to my assessment I confirm and declare that I am the sole author of this work, except where otherwise acknowledged by appropriate referencing and citation, and that I have taken all reasonable skill and care to ensure that no other person has been able, or allowed, to copy this work in either paper or electronic form, and that prior to submission I have read, understood and followed the University regulations as outlined in the Academic Integrity Policy and Procedure for Academic Misconduct available at the following link: <https://www.uclan.ac.uk/study_here/assets/assessment_handbook_2122.pdf>

| **Have you checked the following in order to maximise the grade you can achieve for this assignment?** | **Please mark X to confirm** |
| --- | --- |
| Learning Outcomes have been addressed | X |
| Similarity check via Turn-it-in | X |
| Referencing accuracy according to provided guide | X |
| Grammar | X |
| Spelling | X |
| Word count (or other length limitation as described in the brief) | X |

**WELLBEING**

|  |  |
| --- | --- |
| We wish to support any student who is experiencing mitigating circumstances which prevents students from performing to the best of their ability when completing or submitting assignments. If you are experiencing such circumstances, then you may apply for mitigating circumstances**.** Wherever possible this must be done prior to handing in your assignment. | I believe **that I do / I do not need** to apply for mitigating circumstances for this assignment at this moment in time  Please **delete** as appropriate  (You may still apply for mitigating circumstances if you subsequently feel that your performance has been adversely affected by issues that you may currently be unaware of). |

i

**Multiple Disease Prediction System Using Machine Learning and Web App Deployment Using Streamlit**

**EL4895: MSc Project**

**By**

**Kranthi Kumar Kadari**

**(Reg No: G20953779)**

Project Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in the Course of MSc Mechatronics and Intelligent Machines

School of Engineering

Faculty of Sciences and Technology

© [Kranthi Kumar Kadari]  
UNIVERSITY OF CENTRAL LANCASHIRE  
[semester 3, 2022]

Text

Description automatically generated

Copyright in this work rests with the author. Please ensure that any reproduction   
or re-use is done in accordance with the relevant national copyright legislation

ii

**Acknowledgement**

My sincere thanks to the **“University of Central Lancashire”** for giving a platform to do master’s project on the topic "Multiple Disease Prediction System Using Machine Learning and Web App Deployment Using Streamlit”. Firstly, I would like to thank Dr Essa Anas of the School of Engineering at University of Central Lancashire for giving me the opportunity to work on this project. Whenever I encountered a problem or had a query regarding the drafting of my study or project, Prof. Essa Anas was always helpful to resolve my issues by his weekly meetings, guidance, and immense support helped me to finish the project successfully. This project would not have completed without his motivation, constant support, and valuable suggestion. Every time he thought I needed it; he helped me in the right direction and allowed this paper to be my own work.

I would also want to specially acknowledge Dr Muqi Wulan of the School of Engineering at University of Central Lancashire as the second reader of this project, and I am gratefully indebted to her for her insightful comments on this project during her lectures.

Furthermore, I must convey my sincere thanks to my parents for their unwavering assistance and motivation during my years of education and the work of completing the research this thesis. This achievement would not have been achievable without their help.

Thank you.

Kranthi Kumar Kadari

iii

**Abstract**

These days, deep neural networks are strong, well-liked algorithms. One of the primary issues in deep learning research continues to be comprehending the architecture and how neural networks learn. Their success is largely attributable to the meticulous construction of the neural network architecture. A key component of deep learning, artificial neural networks (ANNs) are being utilised more frequently to create computational models for the neuroscience field. We will also learn about the complete structure of deep neural networks in this report. By using techniques from topological data analysis, we investigate how structure emerges in the weights and activation functions in this case.

On a variety of disease datasets, including those related to diabetes, the heart, and Parkinson's Disease, we train straightforward feedforward neural networks. We then track the weights' evolution and the fundamentals of the backpropagation technique. This study uses the ANN algorithm to construct the Multiple Disease Prediction System, which predicts the risk level of all the diseases stated above. The algorithm makes use of 23 variables in the dataset for Parkinson’s disease, 8 input factors for diabetes prediction, and 13 medical parameters to make predictions about heart disorders. The MDPS forecasts a patient's chance of contracting a disease. The results showed that the suggested diagnostic system can accurately forecast the severity of diseases.

**Keywords:** AI; ML; DL; ANN; Diabetes disease; heart disease; Parkinson’s disease; Accuracy score; Flask, and Streamlit.

iv

Table of Contents

[Acknowledgement iii](#_Toc67432605)

[Abstract](#_Toc67432606) iv

[Table of Contents v](#_Toc67432609)

[List of Figures vii](#_Toc67432610)

[List of Tables viii](#_Toc67432611)

[List of Acronyms ix](#_Toc67432612)

[Notations x](#_Toc67432614)

[Chapter 1.Introduction………………………………………………………………..](#_Toc67432616)....1

[1.1. Background and Context](#_Toc67432617) 1

[1.2. Scope and Objectives 2](#_Toc67432618)

[1.3. Statement of the Problem 2](#_Toc67432619)

[1.4. Achievements 2](#_Toc67432620)

[Chapter 2. Literature Review …………………………………………………………3](#_Toc67432621)

2.1 Theoretical background………………………………………………………………..3 2.2 Related Research…………………………………………………………………….5

[Chapter 3. Methodology and Solution………………………………………………...7](#_Toc67432622)

3.1 Theoretical Method for Modelling…………………………………………………….7

3.1.1 Forward Propagation…………………………………………………………………7

3.1.2 Backward Propagation……………………………………………………………….8

3.1.3 Updating Weights with ADAM and Back Propagation……………………………...8

3.1.4 Activation Function…………………………………………………………………..8

3.1.5 Weight Initialization techniques…………………………………………………….10

3.1.6 Loss function………………………………………………………………………...12

3.1.7 ADAM Optimizer……………………………………………………………………14

3.2 Architecture Design……………………………………………………………………15

3.3 System Analysis………………………………………………………………………..15

3.3.1 Functional Requirement………………………………………………………………15

3.3.2 Non-Functional Requirement…………………………………………………………16

3.4 Data Sources and Data Information…………………………………………………….16

3.4.1 Diabetes Disease Dataset Information………………………………………...............16

3.4.2 Heart Disease Dataset Information……………………………………………………17

v

3.4.3 Parkinson’s Disease Dataset Information……………………………………………...18

* 1. Deep Neural Network Workflow………………………………………………………...19

3.5.1 Data Acquiring……………………………………………………………………….20

3.5.2 Data Pre-processing……………………………………………………………………20

3.5.3 Splitting and Balancing the Dataset……………………………………………………21

3.5.4 Building and Training the Model………………………………………………………22

3.5.5 Evaluating Performance………………………………………………………………..23

3.5.6 Tuning Hyperparameters……………………………………………………………….23

3.5.7 Deployment…………………………………………………………………………….24

3.6 Architecture of Keras…………………………………………………………………….24

3.7 K-Fold Cross Validating Neural Networks………………………………………………24

[Chapter 4. Implementation 26](#_Toc67432623)

4.1 Algorithm used for predicting and model evaluation…………………………………….26

4.1.1 ANN for Diabetes Disease……………………………………………………………..26

4.1.2ANN for Heart Disease………………………………………………………...............30

4.1.3 ANN for Parkinson’s Disease………………………………………………………….33

4.1.4 Requirements………………………………………………………………….............33

[Chapter 5. Model Results 34](#_Toc67432624)

[Chapter 6. Discussion and Model Validation 38](#_Toc67432624)

[Chapter7. Conclusion and Future Scope 42](#_Toc67432625)

[6.4. Future Work…………………………………………………………………………...42](#_Toc67432629)

[References 43](#_Toc67432630)

[Appendix A: The title of Appendix A 45](#_Toc67432631)

**A Bar graph between Accuracy score and diseases………………………………………45**

**B) Keras……………………………………………………………………………………...45**

**C) Project Gantt Chart……………………………………………………………………..47**

vi

List of Figures

Figure 2.1: Artificial Intelligence classification……………………………………………3

Figure 2.2: DL and ML Algorithms performance over increase of the data……………….4

Figure 2.3: ANN Architecture……………………………………………………………..5

Figure 3.1: Single neuron with forward and backward pass……………………………….7

Figure 3.2: Sigmoid Function and derivative of sigmoid function…………………………9

Figure 3.3: ReLU Function and derivative of ReLU function…………………………….10

Figure 3.4**:** Impact of deep learning design on learning: effect of learning rate…………...11

Figure 3.5: Cost Function…………………………………………………………………..12

Figure 3.6:Block diagram to build the model……………………………………………..15

Figure 3.7: DL workflow…………………………………………………………………..19

Figure 3.8: Architecture of Keras………………………………………………………….25

Figure 4.1 ANN complete information for diabetes disease…………………………….....28

Figure 4.2: ANN Architecture for Diabetes Disease………………………………………30

Figure 4.3: ANN complete information for heart disease………………………………....31

Figure 4.4: ANN Architecture for Heart Disease………………………………………….31

Figure 4.5: ANN complete information for Parkinson’s disease………………………….32

Figure 4.6: ANN Architecture for Parkinson’s Disease…………………………………...33

Figure 5.1: Diabetes Disease Input Data…………………………………………………..35

Figure 5.2: Diabetes Disease Output Result………………………………………………35

Figure 5.3: Heart Disease Input Data……………………………………………………..36

Figure 5.4: Heart Disease Output Result………………………………………………….36

Figure 5.5: Parkinson’s Disease Input Data and displaying of Output Result……………37

Figure 6.1: GitHub repository page……………………………………………………….40

Figure 6.2: Deployment of a public web app using Streamlit cloud………………………40

Figure 6.3: User Interface design…………………………………………………………..41

vii

**List of Tables**

Table 3.1: Diabetes dataset detailed information………………………………………16

Table 3.2: Heart disease dataset detailed information………………………………….17

Table 3.3: Parkinson’s disease dataset detailed information…………………………...18

Table 5.1 Accuracy for the diseases……………………………………………………34

Table 5.2 Classification report for the diseases………………………………………...34

viii

List of Acronyms

AI Artificial Intelligence

ANN Artificial Neural Network

DNN Deep Neural Network

ML Machine Learning

DL Deep Learning

NLP Natural Language Processing

SVM Support Vector Machine

KNN K-Nearest Neighbor

MDPS Multiple Disease Prediction System

ReLU Rectified Linear Unit

ADAM Adaptive Moment Estimation Optimizer

RMSProp Root Mean Squared Propagation Optimizer

MSE Mean Squared Error

UI User Interface

API Application Programming Interface

URL Uniform Resource Locator

ix

**Notations**

Input feature or attribute

weights

b bias

σ(y) Activation Function

J(w) Loss Function

New Updated Weight

Old weight

SLOPE or GRADIENT

Ƞ Learning rate parameter

Weight Updation in Momentum

Bias Updation in Momentum

Weight Updation in RMSProp

Bias Updation in RMSProp

Beta1(Initial decay rate)

Beta2(Initial decay rate)

Epsilon (

x

Chapter 1

Introduction

1.1 Background and Context

Data is a valuable resource in our internet era, and vast amounts of data were produced across all industries. All patient-related data is included in the healthcare sector's data. Here, a general architecture for disease prediction in the healthcare sector has been proposed. Many of the current models focus on just one disease for each analysis. One analysis, for example, might be performed for heart, AD, cancer, and diabetes. There is currently no method that can analyse multiple diseases simultaneously. Therefore, my focus is on giving consumers rapid and precise disease predictions based on the symptoms they enter and the projected condition. I am outlining a method that makes use of Flask to forecast a variety of diseases. I will examine the study of the diseases Parkinson’s, diabetes, and the heart in this system. Later, many more disease could be added. DNN, Flask, and Streamlit will be used to construct multiple disease prediction systems. Python pickling is employed to save the model's behaviour. The significance of this system analysis is that it considers all the factors that contribute to the development of the diseases under study, making it possible to detect them more effectively and precisely. A python pickle file will be used to store the behaviour of the final model.

Many analyses of the current systems in the healthcare sector only took one disease into account at a time. One system may be used to analyse diabetes, another to evaluate diabetes retinopathy, and yet another to forecast heart disease, for instance. Most systems concentrate on a certain ailment. An organisation must use a variety of models when analysing the health reports of its patients. The method used in the current system is useful for studying only specific disorders. A user can analyse several diseases on a single webpage using the multiple diseases prediction method. The user doesn't have to travel to many locations to determine whether they have a specific ailment. In the various diseases prediction system, the user must choose the name of the specific disease, supply any relevant information, and then simply click the submit button. Invoking the proper deep learning model will cause it to forecast the result and display it on the screen.

Many of the currently used machine learning models for healthcare analysis focus on just one disease at a time. For instance, the analysis of the liver comes first, followed by those of the cancer and lung disorders, respectively. A person must visit various websites if he or she wants to predict more than one sickness. It is not possible to predict more than one disease using a single analysis in a common system. Some of the models' lesser accuracy can have a significant negative impact on patients' health. When a company wants to analyse patient health information, they must use numerous models, which in turn drives up the cost and time. Some of the systems now in use take extraordinarily little into account, which can lead to inaccurate findings.

1|Page

1.2 Scope and Objectives

It is possible to predict numerous diseases at once using multiple disease prediction. Therefore, the user does not have to visit various sites to predict the ailments. We are focusing on the disorders of the heart, Parkinson’s, and diabetes, because there is a correlation between the three disorders. DNN, streamlit, and Flask will be used to implement numerous disease analyses. The user must send both the illness name and its parameters when requesting access to this API. Flask will call the proper model and then return the patient's state.

The major purpose of this project is to forecast several diseases, such as Parkinson's disease, heart disease, and diabetes disease, by employing DNN on them and creating a web interface using Streamlit for implementing all the predicted models. This web application can determine whether a person will develop the condition or not. The following objectives must be met for this project to be completed:

* Evaluation and Predicting of the Models Using Deep Neural Network
* Deployment of Models into Web App (User Interface)
* Web App Deployment Using Streamlit

1.3 Statement of the Problem

A combination of dietary adjustments, medication, and, in some circumstances, surgery can be used to effectively treat diabetes, heart, and Parkinson's diseases. The symptoms of certain disorders can be lessened, and functioning can be enhanced with the right treatment. Predicted outcomes can be used to avoid, and hence lower, the need for expensive surgical procedures. Despite their ability to predict disease likelihood, the instruments are neither accurate nor affordable. With early disease identification, mortality rates and sequelae can be reduced. Since it takes more intelligence, time, and knowledge, it is not always possible to accurately check patients every day, and a doctor cannot consult with a patient for a whole 24 hours. Our world is filled with data today, so we can examine the data and look for hidden patterns with machine learning and deep learning algorithms. It is possible to diagnose diseases using hidden patterns in medical data.

1.4 Achievements

I have achieved better results comparing with the previous existed models. I have used ANN for model prediction and evaluation to train the model effectively. I got accuracies for diabetes disease, heart disease and Parkinson’s disease as 98.37%, 99.17% and 99.36% which means the model was trained and tested perfectly to predict the new dataset. Finally, I have successfully developed public ML web app by using streamlit and GitHub repository. By sharing URL to the users, they can easily navigate to the different diseases whether they are having the disease or not.

2|Page

Chapter 2

Literature review

2.1 Theoretical Background

Artificial intelligence research in the medical field has gradually made its way into the past few years, enabling people to see the potential of integrating AI and medicine. Some of these applications, including diseases and drug response prediction, have shown greater promise in deep learning. Medical disease prediction accuracy has continuously improved, and performance has also been substantially improved in all areas, since the first logistic regression model, and then machine learning, and finally deep learning today.

Deep learning, which is one of the most enthralling branches of machine learning, has emerged as one of the most important methods for processing complex data. Compared with pre-existing methods, deep learning's ability to process complex data, the ability to extract the main features of multidimensional data, the ability to deal with unstructured data efficiently, and the ability to create classification strategies with a higher level of accuracy are all better at these tasks. Deep learning can also be understood and accessed by more people. As deep learning technology has matured in recent years, it has become increasingly useful. It has caused turmoil in numerous domains, including image identification, speech recognition, and natural language processing. Simultaneously, the applications of artificial intelligence to diseases prediction have progressively gained popularity and has produced several amazing outcomes. The features of health data are extremely suitable with the deep learning algorithm, which can do different jobs more effectively.

Diagram

Description automatically generated

Figure 2.1: Artificial Intelligence classification

3|Page

Deep Learning can really solve complex problem statements like image classification, Object Detection, NLP, Chatbots etc. they solve just like that. Deep Learning is becoming so popular. I would like to mention quite a few important points why it is becoming so famous in this current generation.

* As the amount of the data was increasing w.r.t the older machine algorithms which was previously there. At a specific point of time, it started degrading down (almost remained constant). But, in case of deep learning, as the amount of data is increasing, the performance is also becoming pretty much amazing, that basically means **“**EXPONENTIAL GROWTH OF DATA**”** led us to create amazing DL models in terms of **“**ACCURACY**”.**
* Technology Upgradation- To train DL Models we require hardware’s. We can buy cheap and amazing hardware’s ex. NVIDIA-GPU or clouds.
* In ML project, we do feature extraction and train a model to get accuracy but in DL project, Feature extraction and ML algorithms combine.

Diagram

Description automatically generated

Figure 2.2: DL and ML Algorithms performance over increase of the data

The artificial neural network, also known as a perceptron, is the cornerstone of deep learning. It is a structure made of numerous neurons at various levels. An input layer, a hidden layer, and an output layer are all parts of a deep neural network. The back propagation technique used during training forms the basis of the conventional multi-layer perceptron neural network. It incorporates both the forward propagation of information and the back propagation method for loss function optimization. Visually, it is comparable to teaching a computer to create a set of programmes that mimic the functions of the human brain.

4|Page

This would train the computer's intelligence to emulate human vision, hearing, and other intelligent behaviours and allow it to improve over time. Like this, when predicting diseases, a computer can be used to mimic medical professionals to diagnose diseases and gather past expertise through continual practise to increase prediction accuracy, ultimately strengthening the model.

Chart

Description automatically generated

Figure 2.3: ANN Architecture

2.2 Related Research

The research focussed on diabetes since it is one of the deadliest diseases in the world and because it can lead to many distinct types of ailments, including blindness. In this study, machine learning techniques were used to identify the diabetic condition since it is simple and adaptable to predict whether a patient will be ill or not. The researchers wanted to develop a system that would accurately diagnose a patient's diabetes to aid them. In this case, the three main algorithms used were Decision Tree, Naive Bayes, and SVM. They analysed the algorithms' accuracy, which was 85, 77, and 77.3 percent, correspondingly [1].

The article has examined purpose is to prove the significance of the heart in living things. Therefore, it is imperative that heart-related disease be accurately diagnosed and predicted because it can result in deaths. AI/ machine learning thus aid in the prediction of all kinds of natural disasters. Therefore, they use the UCI repository data for training and testing in this research to calculate the accuracy of machine learning for heart disease prediction using k-nearest neighbour, decision tree, linear regression, and SVM. They also compared the algorithms' accuracy: SVM (83%), Decision Tree (79%), Linear Regression (78%), and KNN (87%) [2].

5|Page

According to the system, liver illnesses are a major cause of death in India and are also regarded as a serious health problem worldwide. thus, early liver disease detection is challenging. So, we can accurately diagnose liver disease utilising automated software and machine learning methods. For quantitative assessment, they used and contrasted the SVM, Decision Tree, and Random Forest algorithms, measuring precision, accuracy, and recall metrics. The levels of accuracy are 95, 87, and 92 percent, respectively [3].

As per Ordonez [11], Some fundamental patient features may be used to detect cardiac disease. In their research, scientists created a method to estimate a patient's chance of developing heart disease based on a total of 13 basic human features, including sex, blood pressure, cholesterol, and other factors. They added two additional characteristics, smoking habit and overweight, to the study database. Data-mining classification techniques including Decision Tree, Naive Bayes, and Neural Network are utilised to generate predictions, and the outcomes are evaluated on the heart disease database. Multi-Layer Perceptron with Back-Propagation is used to create an intelligent and successful heart attack prediction system in a research study by Shantakumar et al. [12]. As a result, the MAFIA algorithm is used to mine the frequency patterns of heart disease based on the retrieved data.

6|Page

Chapter-3

Methodology and solution

3.1 Theoretical Method for Modelling

As we all know that Deep Learning is very complicated to understand the basic concepts because of the indepthness in this field and we will be having fun as well if we get deeper into it. So, I want to discuss about theoretical concepts which we followed for the model prediction and development of our project. Following theoretical concepts and maths intuition which works internally to predict our model.

3.1.1 Forward Propagation

In forward propagation, in first stage the inputs will get multiplied by assigned weights (y = ) and in the second step, we will try to apply an activation function (σ(y)). So, we get an output (z). In the next step that output ‘z’ will be multiplied with assigned weight (z ()+ b) and activation function in second step (σ(z)).

Diagram

Description automatically generated

Figure 3.1: Single neuron with forward and backward pass

In forward propagation, we will be getting wrong output if we compare with actual output. In binary classification if it is less than 0.5 it is 0 and greater than 0.5 is 1. We define “LOSS FUNCTION”. There are many loss functions, but I have used MSE (Mean Squared Error) and Binary cross entropy for our case because our problem statement is binary classification. We will be reducing the loss function value till we get predicted output and actual output similar. So, for that we apply a concept called BACKWARD PROPAGATION.

7|Page

3.1.2 Backward Propagation

We will update all the weights in the backward propagation. That will be our first Iteration. Next, we will again do forward propagation and we calculate predicted output, loss function and we reduce loss function by backward propagation by updating all the weights. That will be our second iteration. We continue to do so many iterations until we reach actual output.

We use optimizers like ADAM, Gradient Descent, Stochastic Gradient Descent, Ada Delta, and Adagrad. These all are used to reduce loss function. I have used ADAM in our project because this is the most famous optimizer which combines Momentum and RMS prop techniques. Momentum helps for the smoothening and RMS PROP will be able to change learning rate (ƞ) in a sufficient manner. So that value will not go high.

3.1.3 Updating Weights with ADAM and Back Propagation

Weights which are connected to neuron interactions should be changed following forward data passes through the network to learn neural networks. For next forward passes, these weights are modified to assist balance out the inequalities between the actual and expected results. However, how precisely are the weights changed? Consider what would be required at every neuron to significantly alter a particular weight before we move on to the actual changes. The error might be a helpful measurement as we are talking about the differences between the actual and predicted values, therefore each neuron will need to have their individual error delivered backward through the network to them to facilitate the updating process; hence, backpropagation of error. After finishing the forward pass, the neuron weights will be updated to reflect the degree of error transmitted backward.

Why do we care about meticulously updating weights at all? Why not just try out a lot of different weights to determine which works best? Well, this is not a terrible notion when dealing with a single neuron and weight. In this case, backpropagation would not even be necessary. To balance and change a potentially very large number of weights and make informed estimates about how to fine-tune them would not be an incorrect decision.

3.1.4 Activation function

Sigmoid and Relu are the two activation functions used to predict the model in the deep neural network.

1. Sigmoid Function

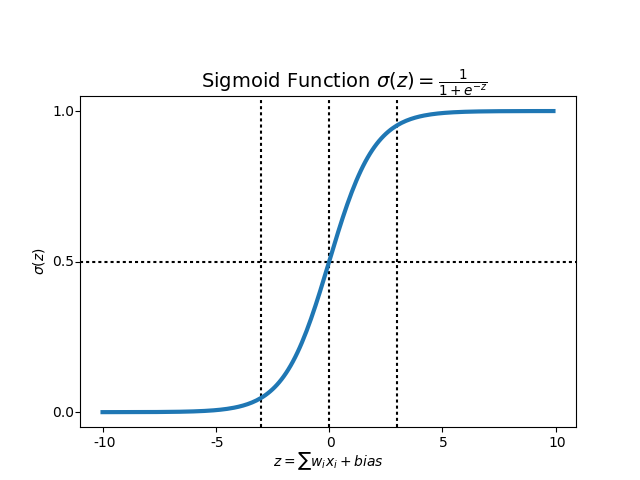
We employ the sigmoid activation function primarily since it occurs within (0 to 1). As a result, it is particularly utilised for algorithms whose outcome is a probabilistic prediction. The sigmoid is really the best alternative because anything has a likelihood that only occurs within 0 and 1.

8|Page

The function might take numerous forms. Therefore, we can determine the logistic curve's gradient between any two positions. Although the function is continuous, its derivation is not.

Sigmoid Function σ(z) = = [ 0:1]

Derivative of Sigmoid function: = [ 0:0.25]

Chart, histogram

Description automatically generated

Figure 3.2: Sigmoid Function and derivative of sigmoid function

Advantages:

1. Clear predictions i.e., very close to 1 or 0.
2. Derivative of sigmoid function ranges from 0 to 0.25.
3. Output values lies between 0 and 1 which normalizes the output of each neuron.
4. Smooth gradient which is helpful prevents jumps in output values.
5. ReLu Function

The ReLU **(Rectified Linear Unit Activation Function)** is the activation function that is employed most frequently worldwide. Considering that practically, it is used almost in all deep learning techniques.

As shown, the ReLU is only partially fixed (from bottom). When z is less than zero, f(z) equals zero, and when z is more than or equal to zero, f(z) equals z. The derivative and the function are both monotonic. However, the problem was that all negative numbers instantly turn zero, thus reduces the model's capacity to successfully fit or train from the data. In the structure, any negative input simply becomes zero when the negative values are improperly mapped to the ReLU activation function.

9|Page

Chart, line chart

Description automatically generated

Figure 3.3: ReLU Function and derivative of ReLU function

ReLu = max (0, x)

ReLu =

If derivative of ReLu is greater than 0 then

We can clearly see in the above graph.

3.1.5 Weight Initialization techniques

If the weights are not initialized properly **“Exploading Gradient Problem”** occurs. This problem occurs with huge numbers. It will never be able to reach global minima because every forward and backward propagation that weights getting updated it is added or subtracted by a bigger number. So, the weight updating will not happen properly. The weight updation must happen slowly even though we have taken smaller learning rate. So, for solving Exploading Gradient Problem we use **“Weight Initialization Techniques”.**

10|Page

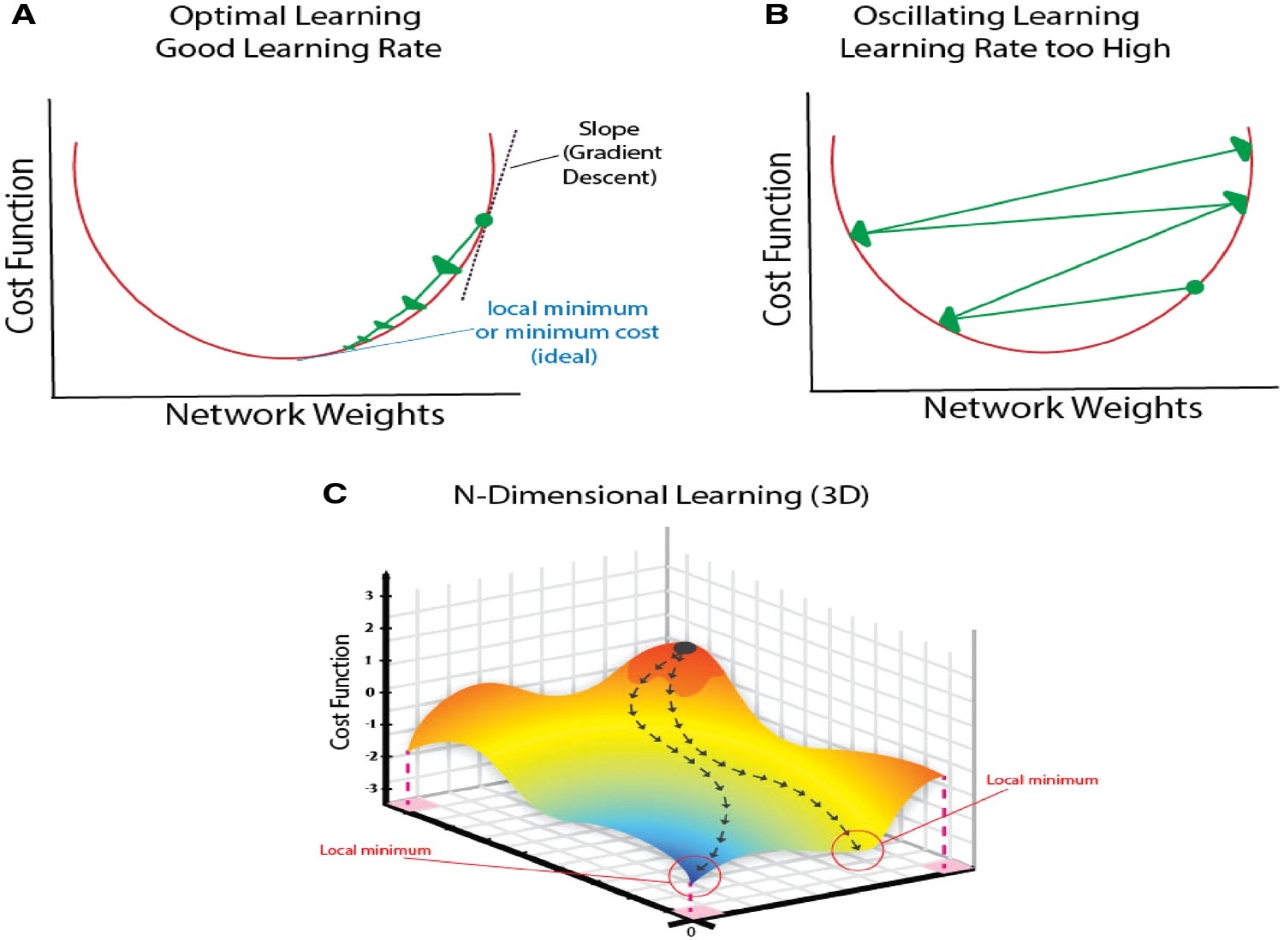


Figure 3.4**:** Impact of deep learning design on learning: effect of learning rate

*(Image Source: https://doi.org/10.1093/eurheartj/ehz056)*

1. Efficient Learning: As a function of weight, the cost function steadily lowers to reach the ideal point, known as the local minimum.
2. Cost function overshoots minimum and oscillates because learning rate is too high.
3. An analysis of the cost function using gradient descent simultaneously examines two variables.

Below problems to be overcomed:

1. Weights are initialized with a big value [ Exploading Gradient Problem]
2. Weights are all 0’s [ neuron will not learn anything]
3. Weights are all same

**Weight Updation Formula:**

Logo, company name

Description automatically generated

11|Page

So, Researchers come up with various weight initialization techniques to overcome all the above-mentioned issues. They decided that weights should be small, should be different and have different variance. By studying all the techniques, I have used He Uniform and Glorot Initialization techniques which gave me the better results.

Text, Word

Description automatically generated

3.1.6 Loss Function or Cost Function or Error Function

After getting a predicted output. We calculate loss function using Mean Squared Error or Binary Cross Entropy and It has to decreased by the Adam Optimizer. Since out problem statement is related to classification thus, we are using Loss functions as MSE and binary cross entropy.

Diagram

Description automatically generated

Figure 3.5: Cost Function

= SLOPE or GRADIENT (partial derivation, Chain rule)

12|Page

Text, letter

Description automatically generatedAt Global Minimum, Neural Network will understand that and stops the training. But, In real world scenario slope does not equal to zero because we will be having many features and dimensions.

**For LHS of Cost Function:**

For Left Hand Side of the cost function, if we consider the slope is negative and repeats the same process and updates the weight until it reaches, .

Table

Description automatically generated

**For RHS of Cost Function:**

For Right Hand Side of the cost function, if we consider the slope is positive and repeats the same process and updates the weight until it reaches, .

Table

Description automatically generated

13|Page

3.1.7 ADAM (Adaptive Moment Estimation) OPTIMIZER

Combination of both momentum and RMS PROP called as ADAM Optimizer. This was the mostly used algorithm in DL.

**Algorithm Implementation:**

Text

Description automatically generated with medium confidence

* Momentum equations

Text

Description automatically generated

* RMS PROP equations

Diagram, text

Description automatically generated

* Bias Correction:

A picture containing table

Description automatically generated

* ADAM:

Diagram, text

Description automatically generated

14|Page

3.2 Architecture Design

Diagram

Description automatically generated

Figure 3.6:Block diagram to build the model

As there is a correlation between heart disease, diabetes, and Parkinson’s disease, I have conducted experiments on these three conditions. The dataset for heart disease, diabetic disease, and Parkinson’s disease has been imported from the UCI ML repository and Kaggle competition as the first phase. After the dataset has been loaded, each inputted piece of data is shown. After pre-processing the data for visualisation, where we must look for outliers, missing values, and scale the dataset, we divide the data into training and testing. After that, I used the testing dataset to apply knowledge of the classification method and Deep Neural Network to the training dataset. We will select the algorithm with the highest accuracy for each ailment after applying our ability. After that, we created a pickle file for each ailment and combined it with the Streamlit and Flask framework so that the output of the model could be shown on the website.

3.3 SYSTEM ANALYSIS

3.3.1 Functional requirement

* The patient can forecast the disease using the system.
* The user enters the data for the specific disease, and a trained model will supply the results based on that data.

15|Page

3.3.2 Non-Functional requirement

* The website will supply wide variety of values during the disease prediction.
* The website would need to function reliably and consistently.

3.4 Data Sources and Data Information

For this project, Dataset was collected from **Kaggle competition** which is a publicly available dataset and from **UCI Machine Learning Repository**.

3.4.1 Diabetes dataset

This dataset is derived from the National Institute of Diabetes and Digestive and Kidney Diseases was used for analysis. By using diagnostic measurements, it is possible to predict a patient's risk for diabetes. These samples were chosen with certain restrictions from a wider database. The dataset consists of 8 input medical attributes namely Pregnancies, Glucose, Blood Pressure and so on, and 1 output feature called as ‘Outcome’.

**Attribute Information:**

From a larger database, these instances were selected based on several constraints. A minimum of 21-year-old female Pima Indian patients are treated here.

Table 3.1: Diabetes dataset detailed information

Graphical user interface, text, application, email

Description automatically generated

16|Page

3.4.2 Heart Disease Dataset

These 1988 data sets contain four databases: Cleveland, Budapest, Switzerland, and Long Beach V. However, all published experiments only mention the use of 14 of the 76 properties, including the one that was anticipated. The number 0 indicates that there is no disease, whereas the number 1 indicates the presence of disease.

Table 3.2: Heart disease dataset detailed information

Table

Description automatically generated

17|Page

3.4.3 Parkinson’s Disease Data Set

The Centre for Machine Learning and Intelligent Systems at UCI has retrieved the Parkinson illness vocal dataset from the repository. 23 of the 31 participants in the collection with biological voice characteristics have Parkinson's. The "status" column in the class column is set to 0 for healthy and 1 for PD. The information is in CSV file. The attribute details are as follows:

**Attribute Information:**

Table 3.3: Parkinson’s disease dataset detailed information

Table

Description automatically generated

18|Page

3.5 Deep Neural Network Workflow

The deep learning workflow has seven important components to be followed. They are

1. Data Acquiring
2. Data Pre-Processing or Feature Engineering
3. Splitting and balancing the dataset
4. Building and training the model
5. Evaluation
6. Hyperparameter Tuning
7. Deployment

Graphical user interface

Description automatically generated

Figure 3.7: DL workflow

3.5.1 Data Acquiring

The most important question in a deep learning project is usually always, "Can we collect enough labelled data?" Our model will perform better if we have more labelled data. Our solution's success depends on how well we can gather data. Obtaining data is sometimes the most difficult and crucial aspect of a deep learning endeavour. The finest data sources are often publicly accessible datasets.

19|Page

#Importing the necessary libraries

Text, letter

Description automatically generated

Numerous sizable, labelled data sets are hosted by websites like Kaggle. Working with these carefully managed datasets makes beginning a deep learning project easier. For this project, Dataset have been collected from **Kaggle competition** which is a publicly available dataset and from **UCI Machine Learning Repository**.

3.5.2 Data Pre-processing

After creating our dataset, we must pre-process it to supply characteristics that will be helpful to our deep learning models. When preparing data for neural networks, our three main objectives are to:

A picture containing text

Description automatically generated

1) clean the data

2) deal with categorical features and text

3) normalize our factual variables using normalisation or standardisation methods. Pre - processing stage is also a great chance to get to know our data better.

We will follow similar steps to the remaining heart and Parkinson’s disease dataset.

3.5.2.1 Cleaning Data

Our datasets often include irregular examples, additional characteristics, incomplete information, and outliers. Testing for outliers and removing them, removing useless features, completing incomplete information, and filtering out noisy samples are all acceptable practises.

3.5.2.2 Scaling Features

Our algorithms may face difficulty with input parameters with high values since we start neural networks with minimal weights to stabilise learning. Therefore, we often scale real-valued variables in two different ways: Features can be standardised so that they have a mean of zero and a variance of one, and they can be normalised so that they range from 0 to 1.

20|Page

*Text, letter

Description automatically generated*

3.5.2.3 Handling Categorical data and Text

The inputs for neural network models are values. This means that all category data and text must be converted to real-valued numbers. - Typically, categorical data are managed by either giving each choice a separate unique integer or by changing them to one-hot encodings. - Before encoding our words as numbers when dealing with raw text strings, we need to undertake a few more processing processes. Tokenizing our data (breaking up our text into distinct words or tokens) and padding our data are two of these stages.

3.5.3 Splitting and balancing the dataset

After data processing, it's time to divide the dataset. Typically, training and validation datasets are created from our data. In some circumstances, we also produce a third holdout dataset known as the test set. When we don't do this, we frequently confuse the phrases "test" and "validation" sets.

We use the training dataset to develop our model, then the validation dataset to assess it. After choosing our model and fine-tuning our hyperparameters, if a third holdout test set has been generated, we evaluate our model on this dataset. With the use of this third step, we can avoid selecting a collection of hyperparameters that just so happens to be effective with the data we selected for our validation set.

The size of our divides and how we will categorise our data are the two main factors to consider while dividing our dataset. We need to fix imbalances in our training set once we've divided our data.

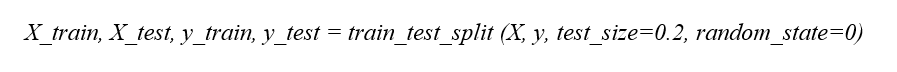
3.5.3.1 Splitting Our Data

We typically save 10 to 30 percent of our data for testing and validation. It is more crucial to assign a bigger percentage of data to the validation set when we have a smaller corpus. This makes it more likely that our validation dataset will accurately reflect the distribution of our actual data.

The **train test split function** from **Scikit-learn** divides our data into training and validation datasets and defines the amount of validation data that will be used.Graphical user interface

Description automatically generated with low confidence

21|Page



3.5.3.2 Stratified train test splits

When dividing a highly unbalanced dataset for classification, more caution must be used since it is extremely conceivable that more occurrences of our minority classes will wind up in the training set or the validation set. In the first scenario, our validation measures won't adequately reflect how well our model can categorise the minority class.

The model will overestimate the likelihood of the majority class in the second scenario. The answer is to employ a stratified split, which guarantees that the training and validation sets include an equal number of samples from each class. The train test split function would assess the percentage of every class and make ensuring that this ratio is the same for both training and validation data if we provide the stratify variable to our array of labels.

3.5.3.3 Handling Imbalanced Data

Deep learning models are challenged by imbalanced data, where certain classes occur considerably more frequently than others. When neural networks are trained on unbalanced data, the final model will be strongly biased towards predicting those classes that are in the majority. This is particularly troubling because, in the normal course of things, we give far greater attention to detecting examples of the minority classes.

Down sampling and up sampling are the two basic methods for coping with unbalanced training data. Great caution can be used when pursuing either of these two strategies, and it's recommended to consult a subject-matter expert before proceeding. By removing samples from our majority class, oversampling allows us to balance our data. We replicate instance of our minority class during oversampling to increase their frequency. Synthetic Minority Oversampling Technique is a well-liked replacement for classical oversampling (SMOTE). The SMOTE algorithm generates synthetic cases and incorporates them into our dataset that are comparable to those in our minority class.

Almost always, we merely make adjustments to our training data's imbalance and leave the validation data alone. We need to compensate for imbalance only after our train-test split to only supplement our training data. Prior to splitting the data, we rarely oversample it. If we do, replicas of our testing data might find their way into our training data. We term this an information leak.

3.5.4 Building and Training the Model

After dividing our dataset, we may select our layers and loss function. We also need to choose a suitable number of concealed units for each tier. The number of layers you use and the size of each layer that works best depend entirely on your data and architectural setup.

22|Page

It's best to begin with a couple layers (2-6).

Typically, we use between 32 and 512 hidden units for building each layer.

As we progress through the model, we also tend to reduce the size of hidden layers.

Typically, we begin by using SGD and Adam optimizers.

It's customary to set an initial learning rate by default to 0.01.

3.5.5 Evaluating Performance

We assess the performance of the model on our validation set after each training iteration. At training time, when we supply a validation set, Keras takes care of this automatically. We can predict how well our model will perform on brand-new, untested data based on how well it performed on the validation set.

It's crucial to pick the right metric when evaluating performance. The significance of accuracy (and even AUC) will be diminished if our data collection is severely unbalanced. We should probably think about metrics like recall and precision in this situation. Another helpful metric that combines recall and precision is the F1-score. A confusion matrix can be used to see which data points are and are not misclassified.

3.5.6 Tuning Hyperparameters

Almost usually, we will need to make adjustments to our initial hyperparameters. We experiment with various learning rates, batch sizes, architectures, and regularisation methods for training and assessing our model. We should check our metrics and loss as we adjust our parameters and keep an eye out for any hints as to why our model is having trouble. Unstable learning shows that we probably need to slow down and/or speed up our batch sizes. We are overfitting if there is a difference in performance between the training and assessment sets. As a result, we should either reduce the size of our model or apply regularisation (like dropout). We are underfitting if we perform poorly on both the training and test sets. We may need a larger model or a different learning rate.

Starting with a simpler model and increasing the hyperparameters until training and validation performance diverge, showing overfitting to the data, is a typical strategy. Critically, your scores will vary regardless of the hyperparameters since the weights of neural networks are randomly initialised. Running the same hyperparameter configuration a few times with different random seeds is one technique to ensure that your conclusions are accurate. We are prepared to use our model once our outcomes satisfy us. Now is the time to employ any holdout test sets we created that were independent of our validation data to validate our model. The performance of our model on omitted data is finally guaranteed by the holdout test set.

23|Page

3.5.7 Deployment

After a model has been trained, we might want to use it in the actual world. This is especially true in professional contexts where our networks will be used by clients and employees or be used internally by our tools and products. There are three main things to think about while implementing a neural network.

* Even managing traffic from numerous users requires a large amount of processing when using a neural network to assess a single input. As a result, it's crucial to host the Docker container where it can access strong processing resources when deploying a neural network model in a container. A fantastic place to start is with one of the cloud platforms like AWS, GCP, or Azure. These platforms offer adaptable hosting services for programmes that can grow to accommodate shifting demand.
* A popular method for interacting with our model via the web is Flask, a web framework built on Python. Requests may be managed, and inputs passed to our model using Flask.
* Where we host our model may have an impact on this. However, Docker Containers are a well-liked all-purpose answer to this final query. For our application to operate quickly in any computer environment, we need to package up our code and its dependencies (such as the correct version of TensorFlow) using Docker containers.
  1. **Architecture of Keras**

A full framework is provided by Keras to build any kind of neural network. Both creative and incredibly simple to understand, Keras. It supports neural network models ranging from the simplest to the largest and most complicated.

Architecture of Keras can be divided into three types: They are mainly

* + 1. Model
    2. Layers
    3. Core Modules

Each ANN in Keras is defined by a Keras Model. It is interesting to note that Keras models are made up of Keras Layers to display the various layers of an ANN layers such as input, hidden layer, output layer, convolution layer, etc. Activation, loss, and regularization functions are also available to Keras models and layers. Any ANN algorithm (CNN, RNN, etc.) may be expressed simply and effectively using the Keras model, Keras Layer, and Keras modules.

24|Page

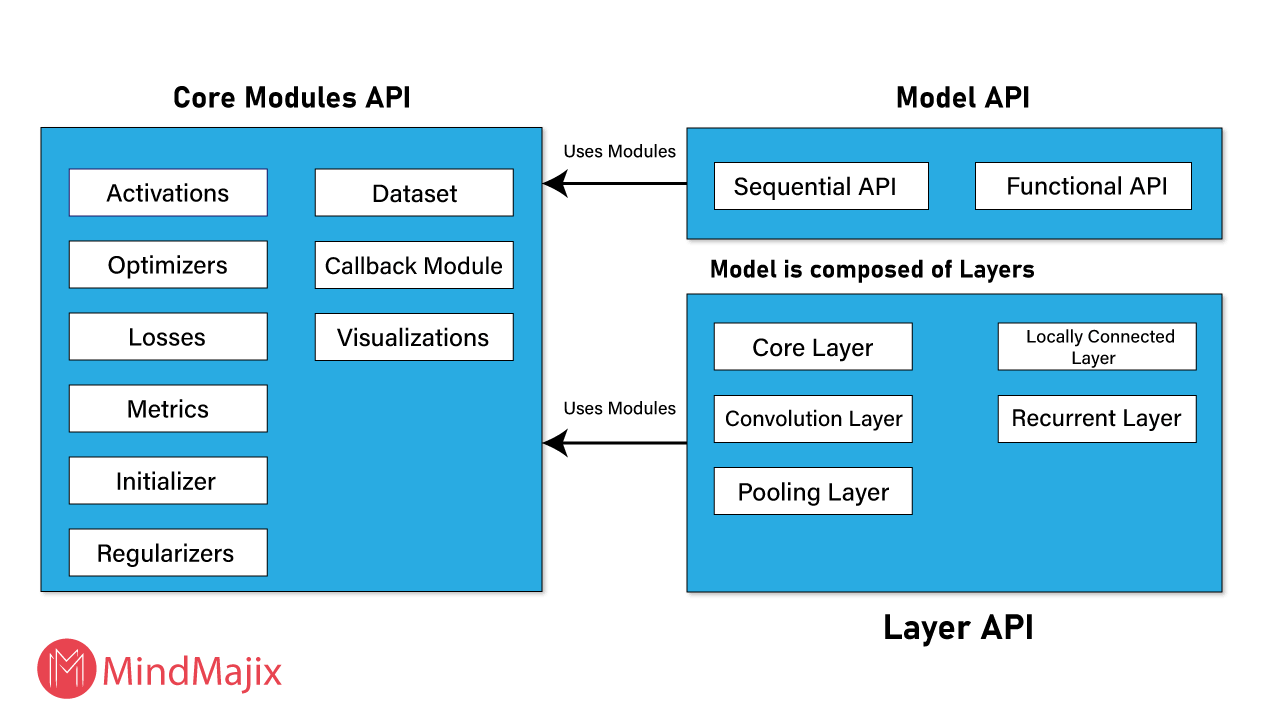


Figure 3.8: Architecture of Keras

* Keras models are mainly divided into two types. They are Sequential API and Functional API.
* Sequential API: It is a Keras Layers configuration in sequence. All of the current neural networks may be represented using the sequential model.
* Functional API helps us to create complex models.
* Ever Keras layer in the model represents a layer in the actual neural network model. The key layers used to develop neural network are mainly pooling layers, convolution layers, Core layers and Recurrent layers.
* Core Modules API: Activations, Optimizers, Regularizers and so on, are some of the neural network functions used foe building the model.
  1. **K-Fold Cross Validating Neural Networks**

The k-fold cross-validation method can be useful in evaluating the performance of a neural network if we have smaller data. By implementing "wrapping" in Keras any neural network into scikit-learn, we can take advantage of k-fold cross-validation. A compiled neural network must first be returned by a function. To make scikit-learn use the model, we wrap it in Keras Classifier. Our neural network can then be used as another scikit-learn learning algorithm. The neural network in our solution was cross validated three times using cross\_val\_score.

25|Page

Chapter-4

Implementation

4.1 Algorithm used for predicting and model evaluation

4.1.1 ANN for Diabetes Disease

The working of the ANN is as followed

**Step-1: Importing the necessary libraries and packages**

For creating the ANN, we need to import libraries like Sequential from Keras. Models, Dense, Dropout, ReLu, Sigmoid, Adam (Optimizer), Mean Squared Error(loss) etc.

I have selected ReLu as activation function in all the hidden layers because this was the most famous one which prevents **vanishing gradient** issue if we compare with sigmoid.

Adam is the best optimizer compared to other

* **Sequential** is responsible to create the neural networks.
* **Dense** is used to create hidden layers between input layer and output layer.
* An **activation function**, such as ReLu or Sigmoid, is used to determine a neural network's output. Each neuron in the network has this type of function attached, which determines whether to stimulate it or not depending on the fact that the data from each neuron is important for the forecasting model.
* **Dropout Layer-** Every hidden neuron will do computations internallythat leads to overfitting.So, to fix this problem, we use a technique called **“DROPOUT”** which is basically regularization parameter**.**

So, I just want to keep Dropout =0.2(which means 20% of the hidden layers will randomly selects) in Hidden Layer 1. Then it will deactivate that neuron (in that neurons input multiplied by weights and bias will not happen because those neurons got deactivated). That operation cannot be done in both front and backward propagation. In the second iteration (or epoch) randomly selects different neurons from Hidden Neuron layer1 and process stays same. Starting neurons which was deactivated earlier they turns to be activated.

Dropout ratio will get multiplied with weights in every layer excluding deactivated neurons, then we get output for the test data. By this way we are overfitting the problem (Low bias and Low variance).

26|Page

***# Let’s make the ANN***

***Text, letter

Description automatically generated***

**Step-2: Initializing the Artificial Neural Network**

**Text, letter

Description automatically generated**

* Initialize the **sequential** library which will be an empty neural network
* Adding the **input layer** and the first **hidden layer** using dense layer by adding parameters. I was considered 12 hidden neurons, input dimension as 8(no.of. features), activation function as relu, and **kernel initializer** as **he uniform**.
* Adding the Second hidden layer with 8 units as hidden layers, activation function as ReLu, and **kernel initializer** as **he uniform**.
* Adding the dropout layer as 20%
* Adding the third hidden layer with 4 units as hidden layer, activation function as relu, and **kernel initializer** as **he uniform**.
* Adding the output layer with 1 unit, because this model is a binary classification problem statement, so activation function taken as “Sigmoid”, and **kernel initializer** as **glorot uniform.**

27|Page

* **Compiling the ANN:**

I have used loss function as Mean Squared Error and optimizer as “Adam” which will reduce loss and metrics is Accuracy.

**Step-3: Model. Summary**

It shows us complete information about neural network. It has four layers which consists of 1 input layer with 8 attributes, 3 hidden layers with 12, 8 and 4 as hidden neurons, and 1 output layer which we can see clearly in the below figure (**i.e., ANN for Diabetes Disease).**

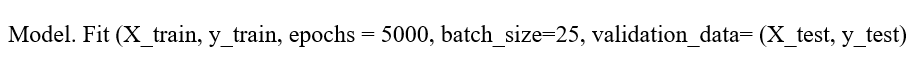
Param explains us about how many weights and biases has been initialized to the layers. 108 has been initialized for first hidden layer, 104 and 36 for 2 and 3 hidden layers, and finally 5 has been initialized for the final output layer.

Table

Description automatically generated

Figure 4.1 ANN complete information for diabetes disease

Finally, I will fit the model with the X\_train, y\_train, validation\_data to evaluate the dataset separately for the test dataset, and epochs as 5000. I have set the batch size as 25, batch size specifically used so that the computation power becomes less and we do not have to load a greater number of records at a single time, and we can be able to run this program in our device easily. After 5000 epochs, I got accuracy training was 91% and validation accuracy was 74%. We will be getting better accuracy by just selecting right kind of Initializer and by using simple neural network.



28|Page

Table

Description automatically generated

The network architecture is first dumped, allowing us to observe the various layer types employed, their output form, the number of parameters they must optimise, and their connections. The neural network is then trained and put to the test. We are aware that TensorFlow serves as the internal computing backend for Keras. For the time being, we won't go into detail about how the training works inside, but we can see that the programme runs through 5000 iterations, getting more accurate each time.

After the training is complete, we put our model to the test on the diabetes dataset and got an accuracy score of approximately 98.37%.

**Step-4: Making predictions and evaluating the model**

Once we have chosen and fit a final DL model in Keras, we can use it to for predictions on new data samples.

**Step-5: Saving the model and loading saved model**

**Graphical user interface, text, application, email

Description automatically generated**

29|Page

Diagram

Description automatically generated

Figure 4.2: ANN Architecture for Diabetes Disease

*Source: Self-Created (Using NN-SVG architecture schematics)*

4.1.2 ANN for Heart Disease

All the six steps which we followed for Diabetes disease will be followed for this heart disease as well. But there are quite a few parameters which we should change. They are mainly input dimensions, epoch number and batch size. I have unchanged remaining parameters like hidden layers, output layer, activation function, optimizer, test size and loss function because dataset we have taken is binary classification problem statement. Those unchanged parameters fit the model very well.

**Model. Summary**

It shows us complete information about neural network. It has four layers which consists of 1 input layer with 13 attributes, 3 hidden layers with12, 8 and 4 as hidden neurons, and 1 output layer which we can see clearly in the below figure (**i.e., ANN for Heart Disease).** Param explains us about how many weights and biases has been initialized to the layers. 168 has been initialized for the first hidden layer, 104 and 36 for 2 and 3 hidden layers, and finally 5 has been initialized for the final output layer.

30|Page

Table

Description automatically generated

Figure 4.3: ANN complete information for heart disease

Finally, I fit the model with the X\_train, y\_train, validation\_data to assess the dataset separately for the test dataset, and epochs as 3000. I have set the batch size as 20, batch size specifically used so that the computation power becomes less and we do not have to load a greater number of records at a single time, and we can be able to run this program in our device easily. After 3000 epochs, I got accuracy training was 99.17%. At the end, I have evaluated the model, loaded and saved the file with the pickle.

**Diagram

Description automatically generated**

Figure 4.4: ANN Architecture for Heart Disease

*Source: Self-Created (Using NN-SVG architecture schematics)*

31|Page

4.1.3 ANN for Parkinson’s Disease

All the six steps which we followed for Diabetes and Heart Disease; I have implemented similar procedure for Parkinson’s disease as well. But there are quite a few parameters which we should change. They are mainly input dimensions, epoch number and batch size. I have unchanged remaining parameters like hidden layers, output layer, activation function, optimizer, test size and loss function because dataset we have taken is binary classification problem statement. Those unchanged parameters fit the model very well.

**Model. Summary**

It shows us complete information about neural network. It has four layers which consists of 1 input layer with 22 attributes, 3 hidden layers with 12, 8 and 4 as hidden neurons, and 1 output layer which we can see clearly in the below figure (**i.e., ANN for Parkinson’s Disease).**

Param explains us about how many weights and biases has been initialized to the layers. 276 has been initialized for the first hidden layer, 104 and 36 for 2 and 3 hidden layers, and finally 5 has been initialized for the final output layer.

Table

Description automatically generated

Figure 4.5: ANN complete information for Parkinson’s disease

Finally, I fit the model with the X\_train, y\_train, validation\_data to evaluate the dataset separately for the test dataset, and epochs as 3000. I have set the batch size as 20, batch size specifically used so that the computation power becomes less and we do not have to load a greater number of records at a single time, and we can be able to run this program in our device easily. After 3000 epochs, I got accuracy training was 100%. At the end, I have evaluated the model, loaded, and saved the file with the pickle.

32|Page

A picture containing text, accessory

Description automatically generated

Figure 4.6: ANN Architecture for Parkinson’s Disease

Source: Self-Created (Using NN-SVG architecture schematics)

**4.1.4 Requirements**

* NumPy == 1.21.4
* Pickle-mixin == 1.0.2
* Streamlit == 1.2.0
* Streamlit-option-menu == 0.3.2
* Scikit-learn == 1.0.1
* Anaconda Navigator
* Spyder (to develop web app)
* Google colab or jupyter notebook (to predict the models)

33|Page

Chapter-5

Model results

Diabetes, heart disease, and Parkinson's disease prediction models all make use of deep neural networks in the system since they supplied the best accuracy. When the patient adds the disease-specific parameter, it will show whether the patient has the particular disease in concern. The parameters will display the necessary value range, and if the value is outside of that range, is invalid, or is empty, a warning message will appear, recommending that the user input a proper value. I have reported accuracy score and classification report with precision, recall and F1-score for the three diseases in the below tables (table 5.1 and table 5.2). I have also shown diabetes disease, heart disease and Parkinson’s disease prediction page where we will be giving inputs value and gets the results whether the person having the disease or not.

Table 5.1 Accuracy for the diseases

|  |  |  |  |
| --- | --- | --- | --- |
| ***S. No*** | ***Disease*** | ***Algorithm*** | ***Accuracy Score*** |
| ***1*** | **Diabetes Disease** | **DNN** | **98.37%** |
| ***2*** | **Heart Disease** | **DNN** | **99.17%** |
| ***3*** | **Parkinson’s Disease** | **DNN** | **99.36%** |
|  |  |  |  |

Table 5.2 Classification report for the diseases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***S. No*** | ***Disease*** | ***Binary*** | ***Precision*** | ***Recall*** | ***F1-Score*** |
| ***1*** | **Diabetes**  **Disease** | **0** | **0.81** | **0.79** | **0.80** |
| **1** | **0.54** | **0.57** | **0.56** |
| ***2*** | **Heart**  **Disease** | **0** | **0.79** | **0.81** | **0.80** |
| **1** | **0.85** | **0.82** | **0.84** |
| ***3*** | **Parkinson’s**  **Disease** | **0** | **0.82** | **0.90** | **0.86** |
| **1** | **0.96** | **0.93** | **0.95** |

34|Page

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5.1: Diabetes Disease Input Data

A screenshot of a computer screen

Description automatically generated with medium confidence

Figure 5.2: Diabetes Disease Output Result

35|Page

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5.3: Heart Disease Input Data

A screenshot of a computer

Description automatically generated with medium confidence

Figure 5.4: Heart Disease Output Result

36|Page

Graphical user interface

Description automatically generated

Figure 5.5: Parkinson’s Disease Input Data and displaying of Output Result

37|Page

Chapter-6

Discussion and model validation

I have clearly elucidated about the model evaluation and prediction for the three different diseases using ANN In the implementation and results section. But for building streamlit based UI we need three important libraries they are mainly Pickle pypi (pip install pickle5) to load saved models, streamlit pypi (pip install streamlit), and pip install streamlit-option-menu to create side bars. We must create a new environment in Anaconda Navigator and copy each libraries and install by pasting in the command prompt. For the development of web app, I have used Spyder using streamlit.

Steps followed to develop web app for three diseases:

**Step-1: Importing the libraries like pickle, streamlit, and option menu**

*A picture containing letter

Description automatically generated*

**Step-2: Loading the saved models**

**Text

Description automatically generated**

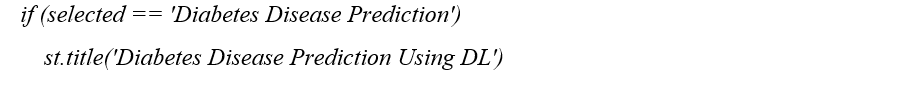
**Step-3: Creating the side bars for the navigation**

*Text, letter

Description automatically generated*

38|Page

**Step-4: Diseases prediction page and Page title**

****

**Step-5: Getting the input data from the user**

**Text

Description automatically generated with medium confidence**

**Step-6: Code and Creating a button for prediction**

**Text, letter

Description automatically generated**

39|Page

Similar procedure for heart disease and Parkinson’s disease needs to be followed from step 4 to step 6 for developing the prediction page, title, code, and button for prediction.

**Step-7: Deploying a Public ML web app by using GitHub and Streamlit cloud**

We must create an account in GitHub and Streamlit Cloud to link both for deploying the models in public. So, streamlit cloud can access files from GitHub repository. I have created a repository name as public\_ml\_web\_app in which I was loaded all files like saved models, model file for web app, requirements file in it. After doing this, we have to open a streamlit cloud and give information like GitHub repository name, Branch and Main file path for the deployment. We can clearly see in User interface design which I mentioned below in Figure 6.1.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 6.1 GitHub repository page

Graphical user interface, text, application

Description automatically generated

Figure 6.2 Deployment of a public web app using Streamlit cloud

40|Page

**6.1 User interface design:** This was the public ML web app developed by following all the steps which was mentioned earlier. We can easily access to different diseases by visiting this interface( <https://triplek182-public-ml-web-app-mdps-public-k6qkjq.streamlitapp.com/>)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 6.3: User Interface design

41|Page

Chapter 7

Conclusion and future scope

In this report, Multiple disease prediction system using deep neural network and web app deployment using flask and streamlit are presented.

This project's major objective was to develop a multiple disease prediction system that could accurately forecast diverse types of diseases at once namely diabetes, heart and Parkinson’s disease using DNN and web app deployment using streamlit and flask. The user does not have to navigate via several websites in this project which also saves time. Early diagnosis of diseases can both lengthen your life and spare you from financial hardship. To obtain the highest accuracy for this purpose, we have deployed deep neural network technology and created web app using streamlit. After applying Deep Neural Network, I got accuracies around 98.37% for diabetes disease, 99.17% for heart disease, and 99.36% for the Parkinson’s disease which we can see in the Table 5.1. By applying weight initialization techniques like he uniforms and glorot uniform to the model the accuracy was increased drastically which

7.1 Future Scope

* We will eventually be able to add new diseases to the current API.
* This project can be enhanced by giving medication recommendations to patients along with the data.
* We can include comments from medical experts who can share their thoughts and opinions on specific treatments performed by the practitioner on the patient.
* We can develop making the system user friendly live chat service in which the patient can speak with a clinician about medications for the specific result of their problems.
* This project could be used as training opportunity for new medical professionals and nurses working with these diseases.
* The patient has the option of selecting which medications to use in order to live a better life. Furthermore, if done on a big scale, it may be employed in medical places like hospitals and pharmacies, where a patient would not have to endure long lines for diagnosis if he is experiencing symptoms of any of these diseases.

42|Page

References

[1] Priyanka Sonar, Prof. K. Jaya Malini,” DIABETES PREDICTION USING DIFFERENT MACHINE LEARNING APPROACHES”, 2019 IEEE ,3rd International Conference on Computing Methodologies and Communication (ICCMC)

[2] Archana Singh, Rakesh Kumar, “Heart Disease Prediction Using Machine Learning Algorithms”, 2020 IEEE, International Conference on Electrical and Electronics Engineering (ICE3)

[3] A. Sivasangari, Baddigam Jaya Krishna Reddy, Annamareddy Kiran, P. Ajitha,” Diagnosis of Liver Disease using Machine Learning Models” 2020 Fourth International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)

[4] Using machine learning on big data of healthcare. In *2018 fourth international conference on computing communication control and automation (ICCUBEA)* (pp. 1-6). IEEE.

[5] Preethi, S., Chandan, N., Darshan, N.K. and Gowrav, P.B., 2020. Diabetes Disease Prediction Using Machine Learning. *Int. J. Mod. Trends Eng. Res*, *6*, pp.37-43.

[6] Mujumdar, A. and Vaidehi, V., 2019. Diabetes prediction using machine learning algorithms. *Procedia Computer Science*, *165*, pp.292-299.

[7] Challa, K.N.R., Pagolu, V.S., Panda, G. and Majhi, B., 2016, October. An improved approach for prediction of Parkinson's disease using machine learning techniques. In *2016 International Conference on Signal Processing, Communication, Power, and Embedded System (SCOPES)* (pp. 1446-1451). IEEE.

[8] Kohli, P.S. and Arora, S., 2018, December. Application of machine learning in disease prediction. In *2018 4th International conference on computing communication and automation (ICCCA)* (pp. 1-4). IEEE

[9] (a) Original owners: National Institute of Diabetes and Digestive and Kidney Diseases (b) Donor of database: Vincent Sigillate ([vgs@aplcen.apl.jhu.edu](mailto:vgs@aplcen.apl.jhu.edu)) Research Centre, RMI Group Leader Applied Physics Laboratory, The Johns Hopkins University, Johns Hopkins Road Laurel, MD 20707

[10]'Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection’, Little MA, Mcsharry PE, Roberts SJ, Costello DAE, Moroz IM.  
BioMedical Engineering Online 2007, 6:23 (26 June 2007)

43|Page

[11] Ordonez, C., 2004. Improving heart disease prediction using constrained association rules. In *Seminar presentation at University of Tokyo* (Vol. 4)

[12] Yılmaz, E. and Kılıkçıer, Ç., 2013. Determination of fetal state from cardiotocogram using LS-SVM with particle swarm optimization and binary decision tree. *Computational and mathematical methods in medicine*, *2013*.

[13] Gelb, D.J., Oliver, E. and Gilman, S., 1999. Diagnostic criteria for Parkinson disease. *Archives of neurology*, *56*(1), pp.33-39.

[14] Revett, K., Gorunescu, F. and Salem, A.B.M., 2009, October. Feature selection in Parkinson's disease: A rough sets approach. In *2009 International Multiconference on Computer Science and Information Technology* (pp. 425-428). IEEE.

[15] Kaladhar, D.S.V.G.K., Nageswara Rao, P.V. and Ramesh Naidu Rajana, B.L.V., 2010. Confusion matrix analysis for evaluation of speech on Parkinson disease using Weka and MatLab. *International Journal of Engineering Science and Technology*, *2*(7), pp.2734-2737.

[16] Arumugam, K., Naved, M., Shinde, P.P., Leiva-Chauca, O., Huaman-Osorio, A. and Gonzales-Yanac, T., 2021. Multiple disease prediction using Machine learning algorithms. *Materials Today: Proceedings*.

[17] Tsanas, A., Little, M.A., McSharry, P.E. and Ramig, L.O., 2010, March. Enhanced classical dysphonia measures and sparse regression for telemonitoring of Parkinson's disease progression. In *2010 IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 594-597). IEEE.

[18] Weider, D.Y., Gill, J.S., Dalal, M., Jha, P. and Shah, S., 2016, December. Big data approach in healthcare used for intelligent design—Software as a service. In *2016 IEEE International Conference on Big Data (Big Data)* (pp. 3443-3449). IEEE.

[19] Naraei, P., Abhari, A. and Sadeghian, A., 2016, December. Application of multilayer perceptron neural networks and support vector machines in classification of healthcare data. In *2016 Future Technologies Conference (FTC)* (pp. 848-852). IEEE.

[20] Roderick, O., Marko, N., Sanchez, D. and Aryasomajula, A., 2017. Data Analysis and Machine Learning Effort in Healthcare: Organization, Limitations, and Development of an Approach. *Internet of Things and Data Analytics Handbook*, pp.295-328.

[21] Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C. and Johannes, R.S., 1988, November. Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the annual symposium on computer application in medical care* (p. 261). American Medical Informatics Association.

44|Page

**Appendix**

1. **Bar graph between Accuracy score and diseases**

**Chart, bar chart

Description automatically generated**

1. **Keras**

**Text, table

Description automatically generated**

45|Page

1. **Project Gantt Chart**

**Timeline

Description automatically generated**

46|Page