Analysis of Type 1 Diabetes Verbal Autopsy Data by Machine Learning Techniques

By

THOKOZILE MANAKA

UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

SUPERVISED BY

DR DEEPAK KAR, DR JONG SOO KIM, PROF JUSTINE DAVIES AND DR ALISHA WADE

Department of Physics
UNIVERSITY OF THE WITWATERSRAND

A dissertation submitted to the University of the Witwatersrand in accordance with the requirements of the degree of MASTERS in the Faculty of Science.

FEBRUARY 2019

Word count: ten thousand and four

Abstract

Big data is a term used for data sets with large, diverse and complex structures that are often quite difficult to analyze or visualize using traditional computing methods and approaches. Machine learning (ML) techniques are effective in analyzing these types of data and extracting information from them. Health care systems generate large amounts of data from record keeping and this supports a wide range of medical decisions like population health surveillance and disease management for the overall improvement of the quality of health care delivery. In areas where there are no health registration systems a method of verbal autopsy is relied on to give information of a likely cause of death.

In this study type 1 diabetes (T1DM) verbal autopsy data from MRC/Wits Rural Public Health Transitions Research Unit (Agincourt) was used as a test case for applying modern machine learning classification techniques to ascertain the cause of death by type 1 diabetes. Machine learning techniques used for the classification task were artificial neural networks (ANNs) and random forests which are realized with a keras front end and tensor flow. Machine learning algorithms automatically learn to make accurate predictions based on past observations by learning patterns in the data for this study, they learned features present in patients with diabetes and were able to identify patients who could have died from the disease. This is the first study on type 1 diabetes verbal autopsy data by the two machine learning techniques in South Africa.

Performance metrics like precision, recall, confusion matrix were used for these classifiers because the data was incredibly skewed and the results obtained show that the random forest classifier classified the deaths by diabetes better than the artificial neural network. In particular the roc-score compares favourably with the study that was done by two clinician specialists in the disease whose study was similar.

Dedication and acknowledgements

I would like to stretch out my sincere gratitude to my advisor and supervisor Dr Deepak Kar for the utmost and continuous support to my research, his guidance and immense patience in looking for collaborators and recommendations on ways to go around the thesis, words cannot express how thankful I am.

I would like secondly to thank my co-supervisor, Dr Jong Soo Kim for making this research project possible from his encouragement, immense knowledge impartment he has done of the Machine Learning disciple and in making sure I overcome my coding phobia which is what made this research what it is today. I am very much thankful for the challenging tasks which have not greatly shaped my thinking and working skills.

My heartfelt thanks also go to Dr Alisha Wade and Prof. Justine Davies who provided me the opportunity to get insight into the medical world by allowing us access to the Agincourt Verbal Autopsy data. The guidance to understanding how to read and interpret it was profound and by also sharing their research study on the same data set, I was able to widen my horizon on the available analysis techniques available for data and have a baseline I could compare my results with.

I would like to thank my family, my father and my mother whom I am greatly indebted to for always striving for me to get the best education and supporting me in all throughout my endeavours and helping me realise my potential.
I would lastly like to thank the government of Lesotho, through the National Manpower Development Secretariat (NMDS) for supporting my research and funding me throughout the period of my study. I would like also to thank the National Research Foundation (NRF) for supporting the research project through the Grantholder linked bursary.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED:..T...Manaka DATE: 14 AUGUST 2019

TABLE OF CONTENTS

List of Tables

Table **Page**

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Literature review

1.1.1 Pathophysiology of T1DM

Insulin is a hormone that allows glucose in the blood to enter body cells and fuel it with energy. The immune system of a person with type 1 diabetes attacks insulin producing cells in the pancreas so that insulin cannot be produced. When glucose from foods enters the blood stream but is not able to enter body cells due to the absence of insulin, a build up of glucose in the blood occurs. The initial mechanism that the body uses in trying to get rid of it this glucose is through urination in the kidneys and this leads to extreme thirst, another symptom of diabetes. Other symptoms include fatigue, which is due to lack of energy and as an alternative for this energy, the body breaks down its fat stores which leads to weight loss. Over long periods of time high glucose levels in the blood cause complications like diabetic retinopathy (DR) which affects the eyes. Thirst, excessive urination and weight loss could then be suggestive of a diagnosis of T1DM.

1.1.1.1 Machine Learning in type 1 diabetes

To date, machine learning techniques have been used in analyzing various aspects of diabetes. The first attempt in predicting DR using the machine learning technique of shrinkage and selection operator was reported by E.Oh et al. [1] where they used health record data. Ibrahim et al. [2] used an adaptive neuro fuzzy interference classifier to make a prediction of diabetes maculopathy while Roychowhury et al. [3] conducted a study on the severity in DR using diabetic retinopathy analysis using machine learning (DREAM), the famous screening system that does an analysis in fundus retinal images with varying illuminations and fields of view by machine learning methods. Trorok et al. [4] developed a method where different types of medical data were used as input in

the machine learning technique of Gradient Boosting Machine for screening DR. Kotsiliti et al. [5] built a classification model for predicting diabetic retinopathy. The model was based on patient characteristics and biochemical measures and an analysis of the model's performance showed that the model was good at classifying the condition and that utilization of patient information and routine biochemical measures can be used in identifying patients at risk of developing retinopathy.

Hypoglycaemia is a condition indicating low blood sugar levels and is one other feature of utmost importance in the prediction and diagnosis of diabetes [6]. Sudarshan B et al .[7] used k-nearest neighbour, naïve Bayes, support vector machines and random forests to make predictions using this symptom in patients with diabetes. Random forests made better predictions using this symptom and this is owned by its feature importance criteria which is able to select specific features in the target as the ones having the largest contribution in affecting the outcome of the classifier. Thomas et al. [8] showed that the bootstrapping and ensemble schemes are characteristics that make random forests overcome over fitting problems, are the more effective and non-parametric for a wider range of different datasets.

High levels of glucose in the blood can injure nerves in the body. Diabetic neuropathy occurs when nerves are damaged by elevated glucose levels and can be classified as peripheral neuropathy which leads to foot ulcers and autonomic neuropathy which leads to gastrointestinal disorders. Huang et al. [9] used decision trees, random forests, naive Bayes and support vector machines as risk prediction tools for predicting and implementing early treatment of these complications. They showed that these machine learning techniques have potential in facilitating the early identification of the complications and in the development of more efficient treatments. DuBrava et al. [10]employed random forest's feature importance technique on a database of the diabetes screening research initiative (DiscRi) in the prediction of cardiovascular autonomic neuropathy (CAN).

1.1.1.2 Verbal Autopsy

A verbal autopsy is a record of an interview about the circumstances of an uncertified death. It is conducted through a set of standardized questions which aim to guide and lead to symptoms of a cause of death. A non-clinician field worker interviews a relative of the deceased about the circumstances leading to death [11]. It was recommended by the World Health Organisation (WHO), a specialized agency of the United Nations concerned with public health upon the realization that in most developing countries many deaths go unaccounted for. These are countries that do not have laws, enough resources or necessary reinforcements to make this registration of deaths possible. Sometimes this is due to deaths occurring outside of health facilities.

In 2014, WHO reported that two-thirds (38 million) of the 56 million global deaths each year are

unregistered. Of the 192 countries in the world, only 23 are shown to have high quality death registration systems while 75 have no cause-specific mortality at all. Most of these countries without complete vital registration systems and therefore far less reliable data are among the poorest, least developed and could in turn be the ones that would highly benefit from accurate reporting systems of causes of deaths. To combat this, WHO showed that verbal autopsies (VA) provide a way to diagnose cause of death in these countries.

Traditionally, the interview responses were reviewed by one or more physicians to ascertain a likely cause of death, but recent and emerging works have explored the use of expert algorithms and data-driven methods such as statistical models and machine learning algorithms. Although various VA methods do not predict causes of death of diseases with vague symptoms as accurately as laboratory diagnostics can, the World Health Organisation shows that verbal autopsies have proved to do predictions on likely causes of death with some degree of accuracy. Todd et al. [12] in a study of the validity of verbal autopsies for assessing causes of death revealed that verbal autopsies were highly specific (98% specificity for all causes of death) and yet not very sensitive (< or = 60% sensitivity for all causes).

A verbal autopsy diagnosis is a semi-supervised machine learning problem in which the responses to the questions in the questionnaire form attributes which can either be binary, categorical or continuous and the classification of the disease is the categorical response. The “true” or ‘’positive” labels for the response are the actual disease classification for a verbal autopsy. For a verbal autopsy classifier to be useful for classifying the death of an individual, it should be able to classify a death due to a disease with a sensitivity (true positive rate) near 90%.

The data collected is analyzed to assign an underlying cause of death in one of three ways;

- Physician Review

  The assignment of cause of death by physician review relies to some extent on subjective judgment and results can vary considerably between physicians in the same settings, between settings and even between the same physicians in the same settings. Predefined expert algorithms for each disease of interest have sometimes been used to assign a cause of death (alone) in conjunction with physician review [13].

- Expert Algorithms

  An advantage of using algorithms alone is that the derivation of causes of death in different settings can effectively be standardized. Uncertainties of whether algorithms can really make diagnoses with validities as high as those obtained by physician reviews are still arguable. D. Chandramohan et al.[13] reported that physician reviews produced more accurate results than the application of expert algorithms in a validation study he did

on adult verbal autopsy. In a comparison of the performance of expert algorithms with algorithms derived from VA interview data for different diseases M. Quingley et al. [14] showed that for most causes of death, the data-derived and the expert algorithms yielded the same levels of diagnostic accuracy, and in the case of malaria diseases, the data-derived algorithm performed significantly better with an estimated sensitivity of 78% compared to 47% for the expert algorithm.

- Statistical Models

  Statistical models work by finding an almost simple formulation of a boundary in classification model problems. It does this by finding the relationship between variables to predict an outcome. They as a result encompass a larger set of assumptions about the data than most expert algorithms such as that the relationship between dependent and independent variables be linear and that the errors should be normally distributed for all values of the dependent variable. One disadvantage that these models have is that they are applicable on smaller sets of data with less observations and less variables in comparison to machine learning or expert algorithms and may thus be limited in that regime [15].

  This distinction is highly related to the prediction-explanation distinction. Prediction is obviously forward looking. We train machine learning models on past data to make predictions on current and future data. In contrast, we train statistical models to quantify the relationship between variables, and that relationship only exists in data that was generated in the past. Statistical models are more appropriate if your primary purpose is explanation rather than prediction, a statistical model may be more appropriate.

1.1.1.3 Verbal Autopsy in type 1 diabetes

Diabetic ketoacidosis is a complication of diabetes of severe insulin deficiency. Zabiullah Ali looked at 20 406 verbal autopsies of people in Maryland who died over a six-year period and found that 107 people died from diabetic ketoacidosis, 60 of whom were previously diagnosed with diabetes while 32 were not, and that nearly half who had died with no history of diabetes were in their 40's [16].

Twenty six thousand, six hundred and eighty two autospy reports done at the department of Forensic Medicine in Sydney were studied of which 1914 were of individuals who had diabetes. Of these deaths by type 1 diabetes cardiovascular diseases accounted for 51% of deaths, acute complications of diabetes 27%, unnatural deaths 28% and sudden unexpected deaths 22%. Further, unexpected deaths were much more common in those with type1 diabetes compared with a control population of the same age group [16].

1.2 Aims

1. To compare machine learning techniques and clinician specialist review in using verbal autopsy data to classify deaths potentially due to type 1 diabetes.

2. To determine the performance of two machine learning techniques; artificial neural networks and random forests on the verbal autopsy data with regard to the positive class of the data.

1.3 Research Questions

1.3.1 Why type 1 diabetes?

Diabetes is a global burden, in 2013 the global prevalence of diabetes was estimated at 382 million and is expected to reach 592 as estimated by the International Diabetes Federation. With 175 million of cases undiagnosed currently, a lot of people with diabetes progress towards more severe complications unaware and a great majority of the number of people are diabetic live in low-and middle-income countries where the epidemic is gathering at alarming rate.

1.3.2 Why use the verbal autopsy data?

Verbal autopsy data is important because it provides cause of death data that would not otherwise be available. The Agincourt health and socio-demographic surveillance system (HSDSS) is the research platform of the MRC/Wits rural public health and health transitions research unit. One of its primary works is in strengthening and extending high functioning health and socio-demographic surveillance systems. It serves to decentralize health systems development through it's goals to "better understand health, population and social transitions dynamics in rural (and Southern Africa) [17]. This data thus provides a means to see whether machine learning can identify individuals with type 1 diabetes.

1.3.3 Why use Machine Learning Methods?

Machine learning algorithms, in comparison to statistical models, are more accurate, automatic, faster, customizable and more scalable and thus help minimize if not eliminate misclassifications or in the case of disease diagnosis any form of mis-diagnosis that can be encountered. Machine learning algorithms can therefore be an indispensable tool for predicting and diagnosing and diabetes. Machine learning techniques like stochastic gradient descent, streaming of data over networks and processing of it in mini batches increase the capacity concept further. Computational speeds, which are enhanced by moving data computations from central processing units to faster graphics processing units are other benefits of using machine learning. Machine learning techniques can therefore prove to be an indispensable tool for analysing verbal autopsy data.

CHAPTER 2

DEEP LEARNING

According to IBM's 2013 annual report, in 2012 alone 2.5 billion gigabytes of data was generated per day. The Large Hadron Collider (LHC) experiment at Cern has produced over 49 petabytes of data in 2016 with over 250 petabytes of disk storage and over one hundred and seventy five thousand computing cores which have shown to also have increased significantly over time. Over 6 million people own cellular phones in South Africa, medical health records increase daily and the growing amounts of sensor-controlled devices in industries means there are large sets of data being generated everyday. Even though data has gotten bigger in terms of both its access and acquisition, its value, or the information we can get from it, how truthful it is and how much it can be relied upon makes it a need for tools. This makes a demand for ways of analyzing it an indispensable tool. Machine learning is this tool and works by developing computer programs that can access data, extract information from it and use it to learn for themselves [18].

2.1 Machine Learning

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on models and inference instead [19]. The process of learning begins with observations, data or instructions in which to look for patterns and then make decisions based on new examples of data. When the target variable consists of distinct categories, the problem or the learning task is called classification and when the target variable is a continuously varying values the task is called regression. There are essentially four kinds of machine learning algorithms, namely supervised, semi-supervised, unsupervised and reinforcement machine learning.

2.1.1 Supervised Machine Learning Algorithms

These are algorithms that can apply what has been learned in the past to new data. They use labels to predict future events by first analysing a known set of training data, the learning algorithm then produces an inferred function to make predictions of the output values. After sufficient training the system is able to provide predictions for new sets of input. Supervised learning can work to serve either two purposes:

1. To automate time-consuming or expensive manual tasks, e.g. doctor's diagnosis.

2. To make predictions, number recognition e.g. portal codes.

The two tasks need labelled data and examples of various ways to get data examples include history which already has these labels, experiments and labelled data can also be crowd sourced from people as in the case of verbal autopsies. Supervised learning includes regression and classification.

2.1.1.1 Regression

For this task, a machine learning algorithm predicts a numerical value given some input, it does this by giving as output a function f where x is the training set and y is a real/continuous number.

$f : x \rightarrow y$

An insurance company may want to predict an expected premium amount that an insured person can afford to pay. A security company may also wish to make predictions of future prices of security utilities for it's clients. These kinds of predictions are regression tasks.

2.1.1.2 Classification

In classification, a machine learning algorithm specifies to which of the category some input belongs to. The learning algorithm outputs a function $f : x \rightarrow y$ whose outcome is limited to a discrete set of values. Classification tasks include object recognition, where an input can be an image and the output a categorical variable identifying the object in the image.

2.1.2 Unsupervised Learning

Unsupervised learning is when we have input data and no corresponding output variables. For these kinds of cases, a machine learning algorithm can work on its own to find information that is not obvious to humans. This way machine learning algorithms are drawing conclusions from unlabelled data. They can do this through various ways like clustering. Clustering is the process of finding groups of objects that are similar. One particularly used clustering example is $k - means$ clustering where the clustering algorithm divides the training set into k different

clusters of examples that are near each other. This way the algorithm is said to be one-hot encoding. One hot encoding is converting categorical variables into a form that machine learning algorithms can use to make the prediction job easier. As an example, an input x is represented by a k-dimensional one-hot code vector h; if x belongs to cluster 1, then $h_i = 1$ while all the other entries in the representation are zero.

Challenges of unsupervised learning are more pronounced in techniques like clustering, the clustering problem is ill posed in the sense that there is no single criterion that measures how well a clustering of the data corresponds to the real world. Another challenge is that there are many different clusterings that all correspond well to some property of the real world. One can end up with a clustering that relates to one feature but obtain a different, equally valid clustering that is not relevant to our task. Another challenge of unsupervised learning methods is in evaluating the performance of the model. As opposed to supervised learning where for performance measures we can compare the real labels to the predicted labels and note any differences that may be there, there is no data to compare made predictions with in unsupervised learning.

2.1.3 Semi-Supervised Learning

Some cases are a combination of labelled and unlabelled observations and the unlabelled data is more than the labelled ones. For such cases groups of similar observations are formed using clustering. The clustering information and classes of labelled can the be used to assign a class to unlabelled observations. A special form of semi-supervised learning is called active learning. Here the algorithm learns from existing data in a supervised way and generates new inputs and their labels. In the active part of the learning, the algorithm queries an information source that will give hints of the potentially useful unlabelled data. This unlabelled data is the one considered for the learning process after which another query for unlabelled data will lead to further iterations.

2.1.4 Reinforcement Learning

This type of learning uses the concept of "reward for an action." It is involved in objects that function on their own accord like human beings, robots or even artificial neural networks. An example is in creating a model for a tennis match. Actions like serves, returns, and volumes are what to aim at to change the state of the game, current sets and leading players. Each of these actions change the game's current state which will in turn affect the next state, and the following state after that one. All the rewards per states are taken into account to the final state.

2.2 Loss Functions

A loss function is a function that maps an event or values of one or more variables onto a real number and represents a "cost" associated with such an event. It represents a price we are willing

to pay by predicting a value $\hat{y} = f(x)$ in place of a label y for a given input variable x.

To be called a loss function, a function has to be a function of one variable x, and

$$L(y, \hat{y}) = y - \hat{y} \tag{2.1}$$

for a regression problem and

$$L(y, \hat{y}) = y\hat{y} \tag{2.2}$$

for a classification problem.

A loss function is also a Lipschitz function that is

$$|f(x) - f(y)| <= C|x - y| \tag{2.3}$$

for all x and y, where C is a constant independent of x and y [20].
The biggest challenge of statistical learning theory has been in finding not only the necessary but sufficient conditions for the consistency of the Empirical Risk Minimization principle. The choice of which loss function to use for which problem has been a computational issue for a long time [21]. Loss functions for machine learning can be put into two categories, regression loss functions and classification loss functions.

2.2.1 Regression Loss Functions

These are used in regressive problems where the target variables are continuous. In regression input spaces are real numbers and outcome spaces are also real numbers i.e if $f(x)$ or $\hat{y}$ is the predicted (action) and y is the actual observed value (the outcome). The loss function takes the form below and produces a real number.

$$L(\hat{y}, y) \longrightarrow (y - \hat{y}) \in \mathbb{R} \tag{2.4}$$

Types of regression loss functions are:

2.2.1.1 The Square Loss

The square loss function is given by

$$L(\hat{y}, y) = (y - \hat{y})^2 \tag{2.5}$$

where $\hat{y} = f(x)$
The output space y of this function is {-1, 1}. The individual differences between the actual and predicted value is squared, a summation of these then taken over the individual squares and an average taken. The resulting value is the mean square. This is the standard loss function for many problems in which larger values are in fact more important like if we want to calculate an integral over predicted values. The square loss function is depicted in Figure 2.1

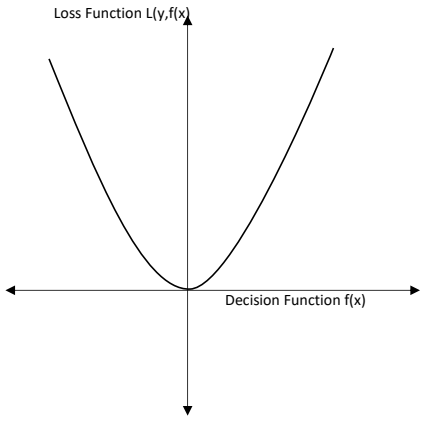

Figure 2.1: Square Loss

2.2.1.2 Absolute or Laplace or Ll Loss

This loss is the absolute value of the differences between the actual data labels and the predicted values.

$$L(y, \hat{y}) = |y - \hat{y}| \tag{2.6}$$

This loss function has the property of not being differentiable and is not affected by outliers as much as the square loss is. Outliers are extreme values that fall a long way outside of the other observations.

2.2.2 Classification Loss Functions

For an input x, the output variable of classification problems is usually a probability value $\hat{y} = f(x)$, called the score for the input x and the target variable is a binary variable, 0 for false and 1 for true. Classification loss functions have the following properties:
The outcome space y of classification loss functions is

$$y = \{-1, 1\} \tag{2.7}$$

If

$\hat{y} > 0$ we predict 1

and if

$\hat{y} < 0$ we predict -1.

2.2.2.1 Zero-One Loss (0-1)

It is the most commonly used classification loss function and it assigns 1 to loss for a correct classification and 0 for an incorrect classification and can be shown graphically in Figure 2.2.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{\infty} (y - \hat{y}) \tag{2.8}$$

Where $\hat{y} = f(x)$, i the individual entry value, N is the number of training examples and $L(y, \hat{y})$ is non-convex, non-differentiable and discontinuous. This therefore makes this loss function's emperical risk minimization not computationally feasible.

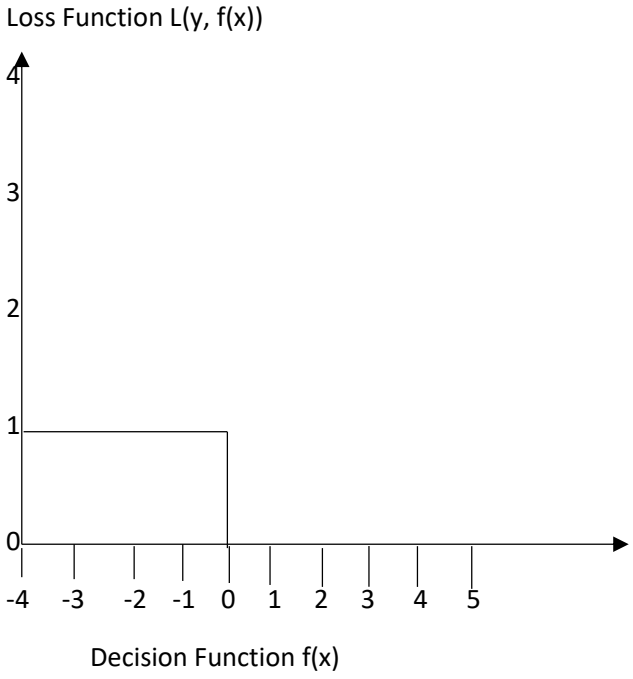

Figure 2.2: Zero One Loss

2.2.2.2 Hinge Loss

For an intended output space

(2.9) $$y = \{-1, 1\}$$

the hinge loss of the prediction $\hat{y}$ is given by

(2.10) $$L(y, \hat{y}) = max(0, 1 - y\hat{y}) = |1 - y\hat{y}|$$

As depicted by Figure 2.3 the error occurs for values of the loss line less than 1.

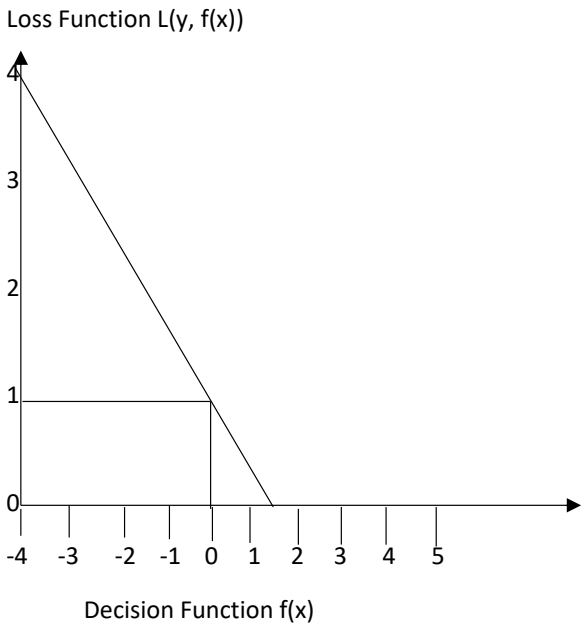

Figure 2.3: Hinge Loss

This loss line is an upper bound of the zero-one loss.

2.2.2.3 Logistic/Log Loss

The logistic loss, also known as the cross entropy loss is defined as

$$L(y, \hat{y}) = -\sum_{i}^{N} y_i \log \hat{y}_i \tag{2.11}$$

where y_i is the actual output and $\hat{y}_i$ is the predicted output by the model and N the number of classes. Figure 2.4 shows the logistic loss graphically.

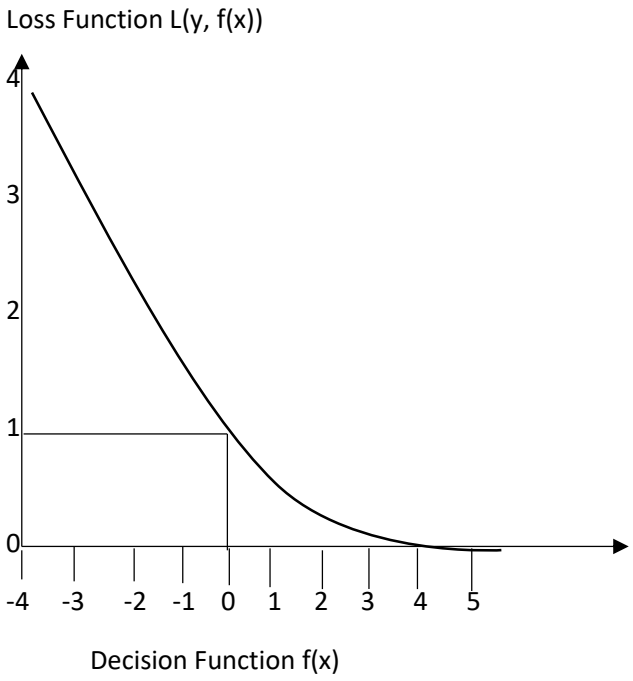

Figure 2.4: Logistic Loss

There are two types of the cross entropy loss, the binary cross entropy and the categorical cross entropy.

1. Binary Cross-Entropy

Classification tasks that involve two classes,a yes or a no or positive and negative or 0 and 1,it is defined by

$$L(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{N=2} y_i \log \hat{y}_i \tag{2.12}$$

Where $y_i\{0,1\}$ is the actual score, i denotes the training input value, $\hat{y}_i$ is the predicted score and N is the number of classes.

2. Categorical Cross-Entropy

   The categorical cross entropy is used when the number of classifications per problem are arbitrary and for such problems the cross entropy loss function can be generalized for a random number of classes as

$$L(y,\hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{J} y_i{}^j \log \hat{y}_i{}^j \tag{2.13}$$

   Where i denotes the training example value and N is the total number of these training examples. While j represents the entry number of the training example J is the total number of entries. Categorical means that the class assignment for sample i is in the form of a vector $\vec{y}_i = (y_1, y_2, ...y_J)^T$ and each component is binary.

2.2.2.4 The Mean Squared Error

The individual differences between the actual and the predicted value is squared, a summation of these taken over the individual squares and an average taken. The resulting value is the mean square. This is the standard loss function for many problems in which larger values are in fact more important like if we want to calculate an integral over predicted values.
Qualitatively the behaviour of the square loss does not change moving from regression to classification.

$$L(y,\hat{y}) = \frac{1}{N}\sum_{i}^{N}(y-\hat{y})^2 \tag{2.14}$$

where y is the actual output, $\hat{y}$ the predicted output and N the total number of observations.

2.3 Optimization Algorithms

The functionality of neural networks lies in a model's ability to minimize a loss and the process by which a neural network does this is called optimization. It does this through iterative techniques

that start at a certain solution and gradually work to improve the performance of the neural network by updating the model's parameters like weights and bias. These parameters are called learnable because the model learns and updates them in the direction of optimal solution.

Optimization algorithms first do a random search which takes a bunch of weights $w's$ thrown randomly into a loss function to see how well they do in minimizing it. Figure 2.5 shows the initialization on weights and the search for the one that gives a minimum error. The updates are done in either a positive or negative direction depending on how far off the error is from the correct value.

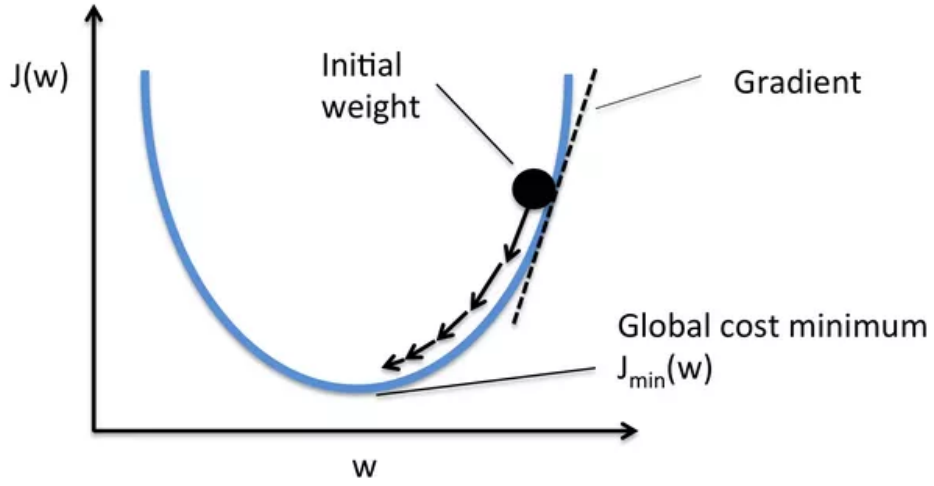

Figure 2.5: Gradient Descent

https://towardsdatascience.com/

In one dimension, the derivative or slope of a function can be computed at any point by imagining that if a small step h is taken in a certain direction, compare the difference in the function value over that step and then drive the step size to zero. The result we get from this is the derivative of the function at that point.

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2.15}$$

In cases where x is a vector, equation(14) generalizes to a gradient which is a vector of partial derivatives in all directions. The slope in any direction is the dot product of the direction with the gradient and the direction of the steepest descent is the negative gradient. The gradient has a property that it points in the direction of greatest increase of the function the opposite direction of which is the direction of greatest decrease of the function. The dot product of the gradient with the unit vector describing the direction gives a slope of a function in any direction.
A gradient gives a linear first order approximation to a function at a point of interest. The

learning part of every machine learning algorithm consists of computing gradients and using these gradients to update learnable parameter vectors like weights. The loss is the function of weights w.

The loss function L takes the form of an average over all individual losses L_i of examples from 1 to N, where N is the number of samples in training data.

$$L(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i) \tag{2.16}$$

The analytical gradient is exact, faster but error-prone. A computed gradient leads to an algorithm that is called gradient descent. Optimization algorithms fall into two major categories; first order optimization algorithms and second order optimization algorithms.

2.3.1 First Order Optimization Algorithms

These algorithms minimize or maximize a loss function by using its gradient with respect to the learning parameters. A first order derivatives tell us whether a function is increasing or decreasing at a point. A fundamental first order optimization algorithm is gradient descent.

2.3.1.1 Gradient Descent

A gradient is represented by a Jacobian matrix J. It is a matrix consisting of first order partial derivatives (gradients). It finds the minima, controls the variance, updates the model's parameters and then lead a model to convergence by using these weight updates. The updating and tuning of the model's parameters is in a direction that can minimize the loss function. A formula of parameter updates is

$$\theta = \theta - \eta \cdot \nabla J(\theta) \tag{2.17}$$

where θ is the learning parameter, η the learning rate, determines how big or small of an update to make $J(\theta)$ the loss function with respect to the parameter and $\nabla J(\theta)$ is the gradient of the loss function.

A back propagation in the network carries error terms and updates weight labels using gradient descent. Gradient descent calculates the gradient of the cost function with respect to the parameters (weights and bias) and updates the parameters in the direction opposite to the gradient of the loss function. The loss function for back propagation becomes

$$L(w) = \frac{1}{2}(y - f(\sum w_i x_i))^2 \tag{2.18}$$

Variants of gradient descent are stochastic gradient descent, mini-batch gradient descent. One stems from the fact that the traditional batch gradient descent calculates the gradients of the whole data set but performs only one update, which can be very large and not fit in the memory. How big or small of and update to do is determined by the learning rate. Another pit fall of using the standard gradient descent is that for large data sets it computes redundant updates. The above-mentioned problems of the standard gradient descent are rectified by a stochastic gradient descent.

2.3.1.2 Stochastic Gradient Descent

Stochastic gradient descent performs parameter updates for each training example, one update at a time which is a great advantage because training patterns are then easily seen and for this it is fairly faster. Parameter updates for each training example is given by

$$\theta = \theta - \eta \cdot \nabla J(\theta; x_i; y_i) \tag{2.19}$$

where $\{x_i, y_i\}$ represents training examples.
The effect of these frequent updates is that parameter updates have a high variance which causes the loss function to fluctuate to different intensities. This helps discover new and possibly better local minima whereas the standard gradient descent only converges to the minimum of the curve. A slight glitch with stochastic gradient descent is that due to the frequent updates and fluctuations, the high variance parameter updates for each training example cause the loss function to fluctuate heavily and it becomes hard to get the minimum value of parameter due to which we might not get the minimum value of parameter which gives the least loss value. The frequent updates and fluctuations complicate the convergence to the exact minimum and thus will keep the model overshooting [21].

2.3.1.3 Mini-Batch Gradient Descent

To overcome problems that come with stochastic gradient descent and standard gradient descent a technique of mini-batch gradient descent is used. It takes the best of both techniques and performs an update for every update for each batch with n training examples in each batch. The greatest advantage it offers is that it reduces the variance in the parameter updates, leading to a much more stable convergence. Another advantage is that it makes use of highly optimized matrix optimizations found in deep learning libraries that make computing the gradient with respect to a mini-batch very efficient. The number of examples in a batch (batch size) ranges from 50 to 256 and can vary as per the application and problem being solved [22].

Gradient descent methods with its variants have challenges which include making a right choice of the learning rate. A learning rate that is too large can greatly hinder convergence and give

more advantage to convergence and cause the loss function to fluctuate around the minimum. At the same time a learning rate that is too small leads to a very slow convergence. The same learning rate problem applies to all parameter updates. A. Cotter et al. [23] showed that if data is sparse and features of data have very different frequencies, it might not be necessary to update all of them to the same extent but rather but perform larger updates for rarely occurring features.

2.3.2 Gradient Descent Optimization

2.3.2.1 Momentum

In physics momentum is a quantity expressing the motion of a body and is equal to the product of the mass of a body and its velocity. In machine learning it refers to adding up a decaying sum of past gradients with decay constant γ in a momentum vector m. The new gradient replaces the one in the gradient descent method. The high variance oscillations in stochastic gradient descent that make it hard to reach convergence are overcome by a technique called momentum.

It does this by adding a fraction "γ " of the updated vector of the past step $t-1$ to the current update vector at time t. As in physics, when an object gathers momentum and it's velocity keeps on increasing, the same thing happens with machine learning parameter updates. The momentum term leads to faster and stable convergence.

Moving the parameters θ by the momentum term $\gamma V(t-1)$ gives an approximation of the next position of the parameter and thus computing the gradient with respect to the future position of the parameters

$$V(t) = \gamma V(t-1) + \eta \nabla J(\theta) \tag{2.20}$$

The final parameters updates are given by

$$\theta = \theta - V(t) \tag{2.21}$$

These parameter updates are what lead to a faster and more stable convergence and an overall reduced oscillation.

2.3.3 Adagrad and AdaDelta

Adagrad makes large updates on parameters that are not very frequent and smaller updates for parameters that are frequent. This makes it well suited for conditions in which data is sparse. Adagrad uses a different learning rate for every parameter θ_i for each time step t and updates

each parameter respectively.

An advantage of Adagrad is that learning rates need not be manually tuned however the learning rates are found to always be decreasing and decaying. A disadvantage of Adagrad is that the learning rate is always decreasing and decaying and the model stops learning [24]. This happens because of an accumulation of each squared gradients in the denominator. This problem can however be rectified by AdaDelta which removes the decaying learning rate. It does this by limiting the accumulated past gradients to a fixed size; it defines the new gradient as the decaying mean of of past gradients [25].

2.3.3.1 Adam

The Adaptive moment estimation combines the momentum advantages with the norm based methods and from this gives off better results. In addition to computing adaptive learning rates for each parameter and storing the decaying average of past squared gradients like AdaDelta, Adam also stores the decaying average of past gradients $m(t)$ like momentum.

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{2.22}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \tag{2.23}$$

m_t and vt are values of the first moment which is the mean and the second moment which is uncentered variance of the gradients respectively. The final formula for the parameter update is

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \cdot \hat{m}_t \tag{2.24}$$

Adam works well in practice and compares favourably to other adaptive learning- method algorithms as it converges very fast and the learning speed of the model is quite fast and efficient and it rectifies every problem that is faced in other optimization techniques such as vanishing learning rate, slow convergence or high variance in the parameter updates which leads to fluctuating loss function [26].

2.3.4 Second Order Optimization Algorithms

Second Order derivatives, also called Hessian (the matrix of second order partial derivatives) are usually faster than the first order derivatives when the second order derivative is known, and they have the advantage that they don't ignore the curvature of a surface. They have the disadvantage of being rather costly to compute in terms of complexity and time and thus are not

used much. The second derivative tells us whether the first derivative is increasing or decreasing and hints at the curvature of the by providing us with a quadratic surface which touches the curvature of the error surface.

CHAPTER 3

NEURAL NETWORKS

A traditional computer cannot solve any problem unless specific steps that a computer needs to learn are known and fed to it through programming. A large team of experts who know a way around solutions to a certain problem is always needed and only then can a problem be solved. Real life problems however do not often have people who are ever ready to solve them.

Neural networks process information the same way the human brain does through a procedure known as learning. They learn from a set of examples they have had experience with and need not be programmed with a set of instructions. The first step of learning is training the neural network with a set of data such that with experience it can learn to solve a task with a new set of data. The first neural network was modelled using an electrical circuit in 1943 by Warren McCulloch, a neurophysiologist and Walter Pitts, a mathematician, who later wrote a paper on how they work [27].

Later works after the advancements of computers a variety of neural network architectures, learning methods and optimization algorithms come forth like one by Nathanial Rochester who simulated the theoretical neural network but failed. There were not enough datasets available to do trainings on as compared to today when data is generated at rapid amounts through numerous mediums like the internet, social media and global stock markets. There were equally no computational powers like today in the era of graphical user interfaces (GPUs) and vast amounts of data can be processed in very short time intervals. These two factors contributed vastly to the development of neural networks. Today neural networks are used in almost every industry including medicine for forecasting, voice and image recognition and have somewhat

become part of every day life [28].

Neural networks are inspired by neurons which are brain cells that receive a number of inputs from a stimuli, combine the inputs, do a non- linear transformation on the sum and then give an output. A set of neurons forming a neural network is shown in Figure 3.1. An artificial neuron is called a perceptron and is depicted in Figure 3.2.
It receives a finite set $X = x1, x2, \ldots, xn$ of discrete random variables, where each variable x_i takes values from a finite set.

The neuron discovers which variable from this set has a higher effect on the outcome, gives more importance to it, gives less importance to others through weights and then gives an output Y which is also a finite set of different values $Y = y_1, y_2, y_3, ...y_n$. The weighted sum of these inputs is then passed through an activation function σ which provides a threshold value of when to allow a neuron to fire for an output.

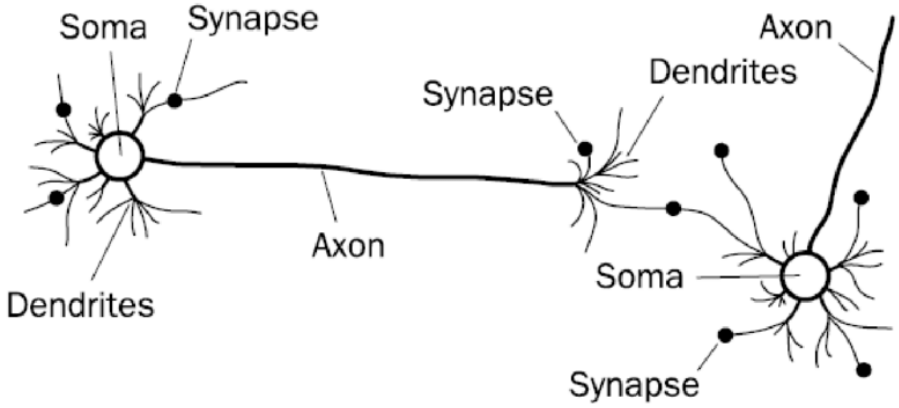

Figure 3.1: Biological Neural Network

https://www.teco.edu/ albrecht/neuro/html/img1.gif

A backward propagation is done on the weights based on the error (actual output - predicted output) to see if there should be a decrease or increase to it. From this loss value the neural network is then able to make a decision that at these certain weights, an error increases or decreases. It is at these error values that the neural network knows whether or not it has reached the final weight value.

Of the various applications of artificial neural networks, those involving medicine are receiving more attention. A lot of current research is on modelling of parts of the human body and recognizing diseases from various scans. Modelling and diagnosis of diseases have also become great areas of research interest and studies show that diagnosis is achieved by building systems

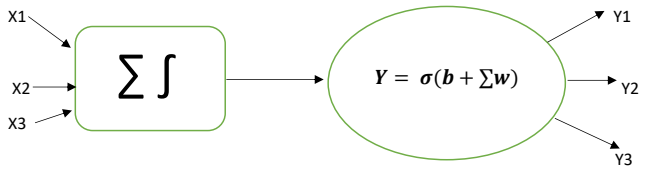

Figure 3.2: Artificial Neural Network

representative of certain diseases of individuals and comparing them with real-time physiology measurements taken from the patient. By carrying out these routines regularly, potentially harmful medical conditions can be detected at an early stage and thus making the process of combating disease much easier. Artificial neural networks are also used highly in business sections of banking and marketing.

3.1 Types of Neural Networks

3.1.1 A perceptron

A percepton is a basic unit of a neural network, also called a single neuron. The output of a perceptron is the compressed weighted sum of the inputs. Logistic regression, or a logistic classifier is one neuron. A collection of these put together form a network and we can have various types of those.

3.1.2 Feedforward Neural Networks

A neural network is called feedforward when a perceptron (neuron) has no backward link to the neurons in the previous layer as input to one of the neurons of that layer. Information flows through the function being evaluated from x, through the translational computations used to define the function f and out through the output y as shown in Figure 3.3.

The first layer is the input layer, the last the output and those in the middle are called hidden layers.

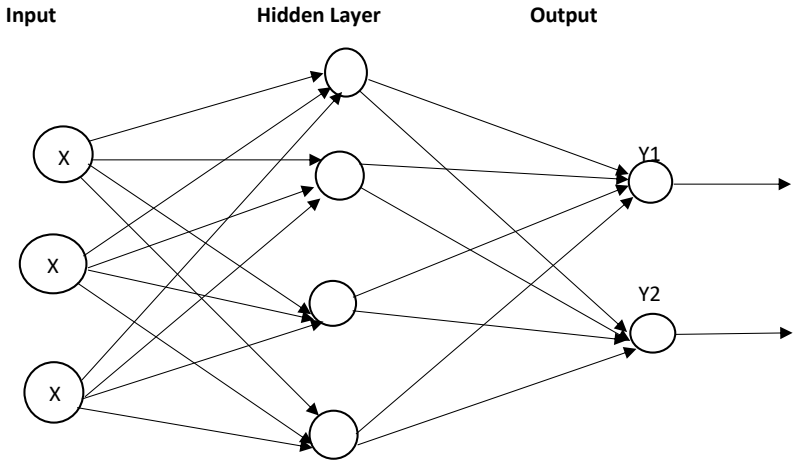

Figure 3.3: Feed Forward Network

3.1.3 Convolution Neural Networks

A convolutional neural network (CNN) is an artificial neural network that has the highest popularity for analyzing images because of it's ability to pick out patterns. The basis of CNNs is the hidden layers called convolutional layers. Like other layers, a convolutional layer receives an input, transforms the input in some way and then outputs the value to the next layer. The transformation on the inputs is a convolutional operation hence the name of the network.

To detect patterns in images convolutional layers use filters. Patterns in images can include image edges, shapes, textures and different filters are for detecting different patterns. One filter can detect a pattern of edges in an image, other filters corners, others shapes like circles or squares. The depth of a neural network determines the sophistication of its filters. This means that in further layers, rather than edges the filters detect fuller objects like eyes, ears, hairs or fur, feathers, scales and beaks if say an image is an animal. In even deeper layers, the filters detect complete images like a full dog, lizards and birds.

3.1.3.1 Convolutional Neural Networks in image classification

A convolutional layer with its respective filters accepts images of handwritten digits and the network classifies them into their respective categories of whether they are images of a one, a two, a three etc. A filter is a relatively small matrix for which the number of rows and the number of columns are initialized with random numbers. The first convolutional layer can specify one filter to be a matrix of size 3x3. When this convolutional layer receives input, the filter slides over each 3x3 set of pixels from the input image until it has slid over every block of pixels making up the entire image. The process of sliding across each and every block of pixels in an image is called convolution.

The filter convolves the entire input and gives a new representation of the input, which is made up of the entire matrix of the stored dot products we got from the filter. This matrix of dot products is the output of this layer which will be passed to the next layer as input. The same process that happened with the filter will happen to the new output with the next layer's filters. For a set of four filters in a convolutional layer, the filters are filled with values randomized as follows;
An image of a handwritten seven can be detected by the four 3x3 filters where in the matrices $-1s$ represent the dark colour black , $0s$ represent grey and $1s$ represent white. The brightest colour (white) is the edge that the filter has predicted and Figures 3.4 and 3.5 show what the four filters had individually predicted of the number seven.

The first filter detects top horizontal edges of the seven, as indicated by the brightest pixels. The second detects the left vertical edges, and the third detects the bottom and the fourth detects right vertical edges which together form an image of a seven.

3.1.4 Recurrent Neural Networks (RNNs)

Recurrent neural networks are the most flexible types of neural networks with regard to the types of data that they can process. While other neural networks take a fixed (single) input and feed it through hidden layers to produce a fixed (single) output classification, recurrent neural networks provide the flexibility in the type of input and output data that a model can process. They offer a variety of model architectures like one to one, one to many, many to one and many to many models which are shown below.

3.1.4.1 One to one model

As depicted in Figure 3.6, the layers of this model receive input which is fed through some set of hidden layers and produce a single output like a set of classification scores or categories.

-1	-1	-1
1	1	1
0	0	0

-1	1	0
-1	1	0
-1	1	0

0	0	0
1	1	1
-1	-1	-1

0	1	-1
0	1	-1
0	1	-1

Figure 3.4: Filters

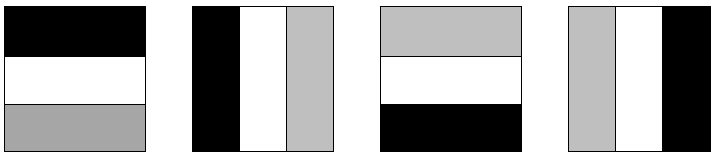

Figure 3.5: Filters as Visuals

3.1.4.2 One to many model

In this model, an input is of one size and the output of variable length. A one to many model is shown by Figure 3.7. In image classification, an image of a fixed size can be input while a sequence of variable length in a form of a caption for that image be the output. Different captions can have a differing number of words and hence the need the output to be of variable length.

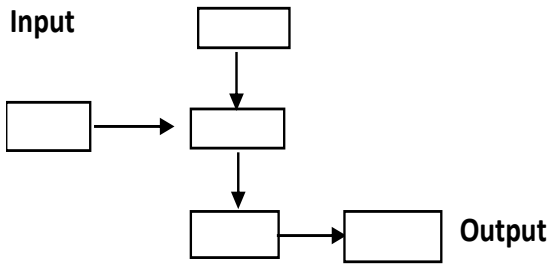

Figure 3.6: One to One Model

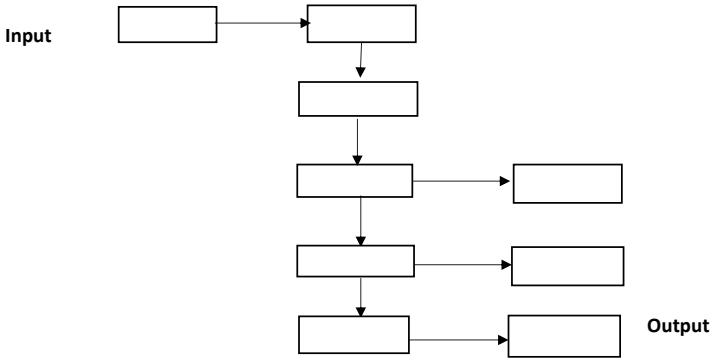

Figure 3.7: One to Many Model

3.1.4.3 Many to one model

For this model the input is of variable length and the output is of a single fixed size. If we have a piece of text and want to find the sentiment of the text; whether the text of a negative or positive sentiment, as input and a video that has a variable number of frames and as output make a classification decision about what may be the type of activity going on in that video. Figure3.8

shows a many to one model configuration.

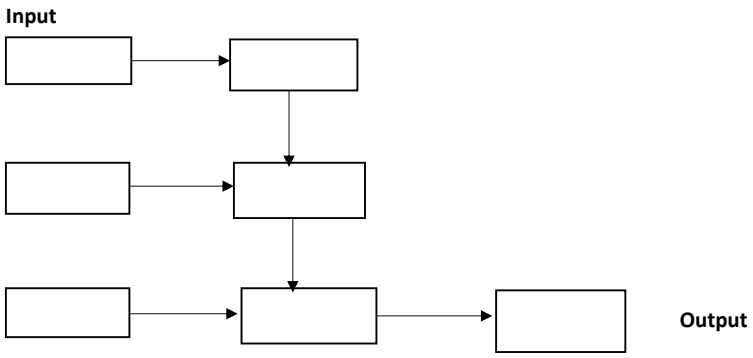

Figure 3.8: Many to One Model

3.1.4.4 Many to many model

For this model both the input and output are of variable length as shown by Figure 3.9. This is mostly seen in machine translations where a sequence of words serve as input and another sequence of words as output. We can input an English sentence and have it translated to French as our output. These models therefore have the capacity to handle both variable sequences in input and output data.

For a many to one architecture as in the sentimental analysis, the last hidden state summarizes the whole sequence. A decision is thus made based on the final hidden layer of this network as the final hidden state summarizes the content from the entire sequence. For a one to many situation, where classification is of a fixed size input to produce a variably sized output, that fixed size input is to initialize the hidden state of the model and the recurrent network will tick for each cell in the output to produce a variably sized output.

Recurrent neural networks are mostly used in language modelling problems to read sequences of words. This can happens at word level where a network produces words at a time or at character level where a model produces a character at a time. Character level language models read sequences of characters and then predict what will the next character be in a stream of text.

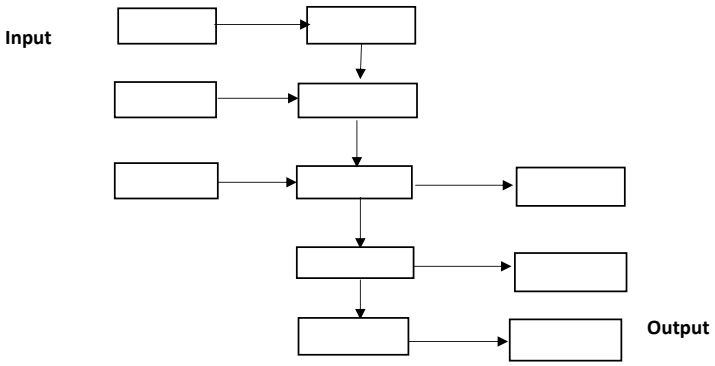

Figure 3.9: Many to Many Model

3.1.5 Recurrent Neural Networks in language modelling

Given a set of letters h, e, l, o and a training sequence “hello”, the training process of a language model starts with feeding the letters of the training sequence as inputs into the model. As with every neural network the inputs are represented in a certain way, in this case letters are represented as four-element vectors. The letter "h" can be represented by a four-element vector with a 1 in the first slot and 0s in the other three slots and the same pattern is used to represent the other letters as depicted in Figure 3.10.

At the first time step during the forward pass, the neural network receives an input letter h that will go into the first RNN cell and produce an output y_t which is the network’s prediction of what letter it thinks is most likely to come next. If the predicted letter is wrong, an activation function like SoftMax or Sigmoid can be used to quantify this loss. At the next time step the second letter in the training sequence is fed into the second RNN cell and the process repeated. The input vector together with the previous hidden state produce a new hidden state that is used together with the second hidden state to make the prediction of the next letter in the training sequence. Training the model with a different set of sequences eventually allows the model to learn how to predict the next character in a sequence, based on the context of all the previous characters that its seen before. Figure 3.11 shows sequentially this process of training carried out by a recurrent neural network.

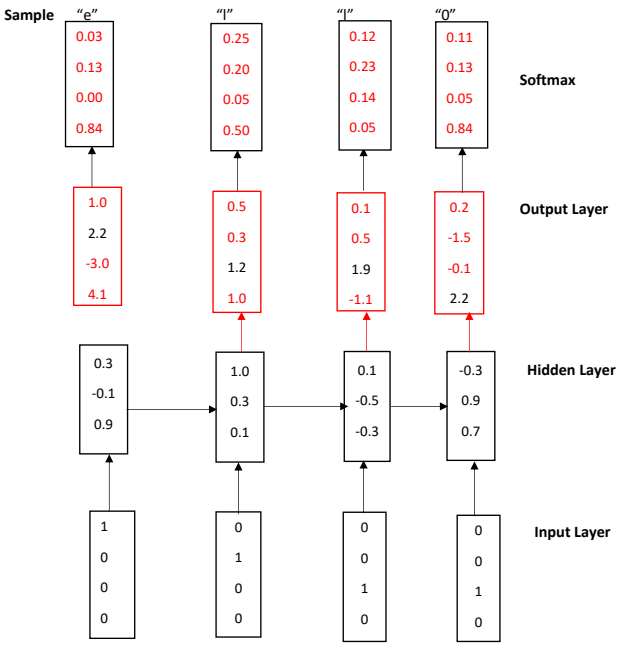

Figure 3.10: Recurrent Neural Network in language modelling

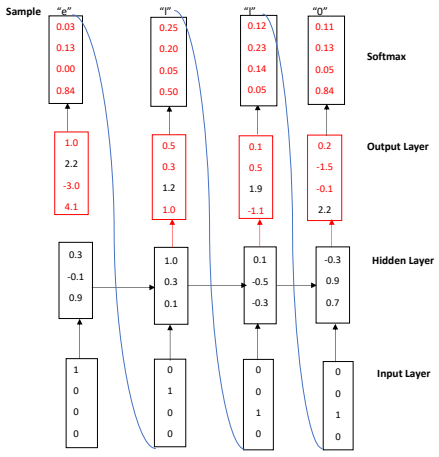

Figure 3.11: Recurrent Neural Network Working

3.2 Activation Functions

An activation function is inspired by the biological activity of the brain where different stimuli trigger different neurons and cause them to fire. The firing of a neuron is denoted by "1" and not firing by "0" and it refers to the transformation of a stimuli or an input in the case of artificial neurons. This transformation is carried out by an activation function. Activation functions do not always make transformations on inputs to be 0 or 1 there are other kinds of activation functions which are a bit more flexible like the Rectified Linear Unit (ReLU) and the Sigmoid activation function where the neuron can be a value between 0 and 1.

The transformation is done on the sum of inputs and weights to give them an upper and a lower limit. Different values of weights w_s give different scores and a quantitative measure of how good a model is can be found from a search with different values of weights. The process of searching through possible values of weights to find the correct one is called optimization.

Activation functions introduce non-linear properties to inputs. A linear function is a polynomial of only one degree which always forms straight lines in graphs and form planes and hyper planes with no curves for functions of higher degrees. Polynomials of higher degrees forming curves when plotted on graphs implies that when non-linear activations are used a neural network with many layers would behave like a neural network with a single layer as a summation of these layers would give a non-linear function. This becomes a problem when modelling neural networks with numerous kinds of data which need to be non-linear and therefore differentiable for the back-propagation optimization strategy to be employed to find a non-linear error gradients and therefore equip the neural network to learn.

3.2.1 ReLU

The rectified linear unit is defined as

$$f(x) = max(0, x) \tag{3.1}$$

and its derivative is

$$f(x) = \begin{cases} 0 \text{ if } x < 0 \\ 1 \text{ if } x > 0 \end{cases} \tag{3.2}$$

and is undefined at $x = 0$

As shown in Figure 3.12, the ReLU activation function transforms an input x to the maximum of either 0 or itself. When an input x is less than or equal to zero the ReLU activation function

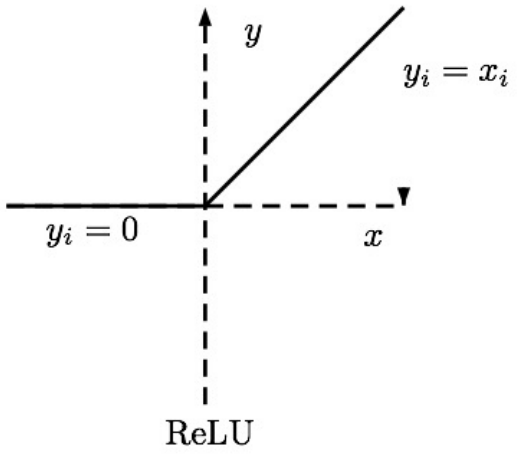

Figure 3.12: ReLU Activation Function

https://www.researchgate.net/ReLU-activation-function.png

transforms the input to 0 and if the input is greater than 0, ReLU gives as output the input value. The more positive an input value is, the highly activated it is.

ReLU has become the standard activation function ever since it has shown to improve the performance of Boltzman machines and were popularized by convolutional neural networks [29]. It has adequate non-linear properties, is not prone to vanishing gradients and greatly accelerates computation time for networks.

One of the pitfalls of ReLU is that it only activates during back propagation when the values fed into it are positive and does not activate for negative values. This becomes a problem because if the neurons are not initially activated, they stay off as zero gradients pass through them. These neurons are termed dead [30]. The mean activation is larger than zero thus the subsequent layers that follow have a positive bias and this can only be overcome by applying a negative to the network.

3.2.2 Leaky ReLU

The leaky ReLU tries to overcome the issue of dead neurons that comes with ReLU; the saturating for negative input values. It does this by forcing the function to produce a constant that multiplies the negative input values and instead of the function being zero, it will have a small negative slope and this means that it allows a small positive when the unit is not active. The leaky ReLU is given by

$$f(x) = max(x, Ax) \tag{3.3}$$

The constant multiplier A is equal to 0.1. The Leaky ReLU produces a maximum value of x and Ax as depicted by the graph in Figure3.13.

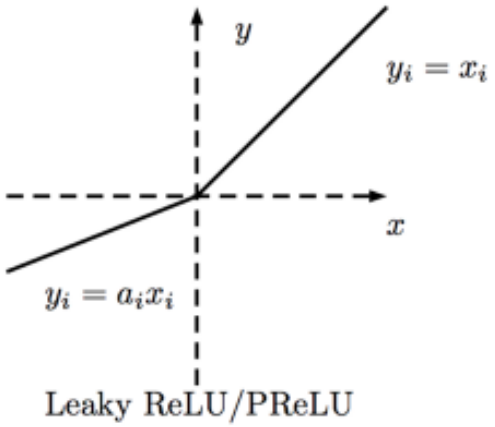

Figure 3.13: Leaky ReLU

https://cdn-images-1.medium.com/max/1600/1*W6BURQnUE62qyxJxMDpdnA.png

3.2.3 ELU

The exponential linear units overcome the problem of ReLU activation functions which have mean activations that are larger than zero and therefore cause layers that follow to have a positive bias. Figure 3.14 shows that exponential linear unit function brings down the mean activations to intervals closer to zero in turn speeding up the learning process and thus achieving higher accuracies in classification [31].

$$f(x) = \begin{cases} x \text{ if } x \geq 0 \\ A(e^x - 1) \text{ otherwise} \end{cases} \tag{3.4}$$

The constant or parameter A is constrained to $\geqslant 1$

3.2.4 Sigmoid

Sigmoid activation functions are curved at their ends. This causes the gradients at those points to be quite low because even when the value prior to applying sigmoid activation function increases considerably, the value after applying does not the gradient also does not increase much and this

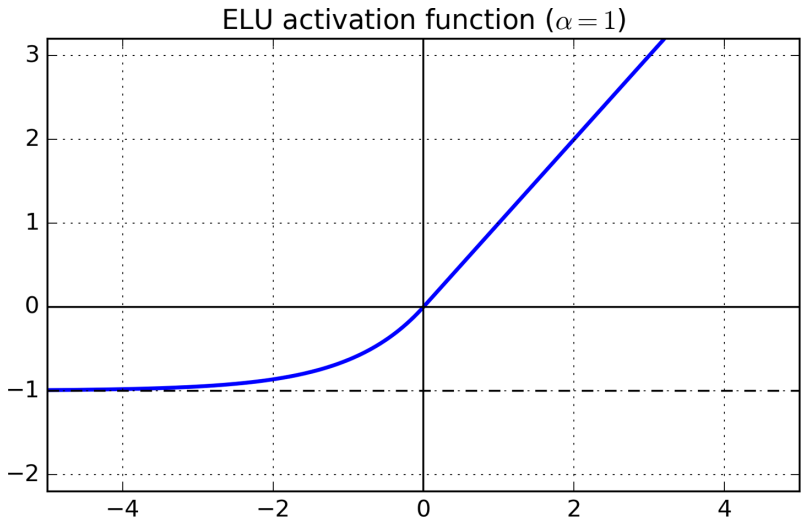

Figure 3.14: Elu activation function

https://blog.paperspace.com/content/images/2018/06/ELU.png

impacts learning. They can be good choices for shallow networks which have a maximum of two layers.

A sigmoid activation function is defined by

$$f(x) = \frac{e^x}{e^x + 1} \tag{3.5}$$

and its derivative by

$$\frac{df(x)}{dx} = f(x).(1 - f(x)) \tag{3.6}$$

The sigmoid activation function's shape as shown in Figure3.15 makes it to be very good for classification (smoothly going from 0 to 1). For highly negative input values the sigmoid activation function transforms the value to a number close to 1, for highly positive input values the sigmoid activation transforms the input into a number very close to 1 and if the input is a value close to 0, sigmoid transforms it to a number between 0 and 1. This says that 0 is the lower limit and 1 is the upper limit for a sigmoid function.

One of the pitfalls of sigmoid activation function is seen in cases where the activation of the neuron saturates to either of its limits. The gradients at these two limits is almost zero or vanishing and during back propagation no signal flows in neurons to update the weights. Another problem with a sigmoid activation function is that its output starts at zero and ends at one. This means the values after the function will be positive and makes the gradient of the weights become either all positive or all negative. This makes the gradient updates go too far in directions and makes

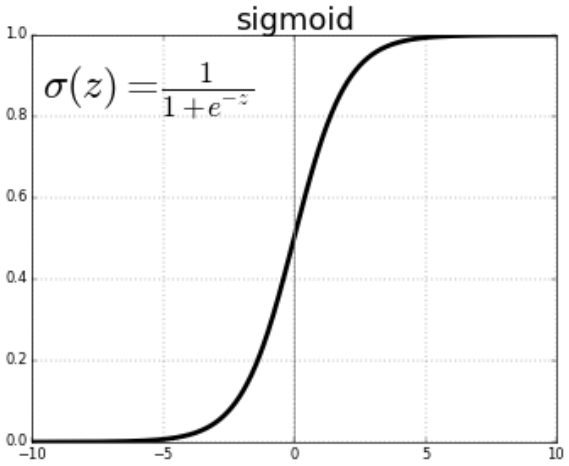

Figure 3.15: Sigmoid activation function

https://towardsdatascience.com/activation-functions-neural-networks

makes optimization harder.

3.2.5 Tanh

The hyperbolic tangent function compresses a real number into a range between -1 and 1 (its range) instead of 0 and 1, so its output is centred at zero. This makes it the most preferred activation function in practice as optimization across this range is easier. It does however like the sigmoid suffer from a vanishing gradient problem.

It is defined by

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.7}$$

and its derivative by

$$\frac{d(f(x)}{dx} = 1 - (f(x))^2 \tag{3.8}$$

The derivative of tanh for back propagation has a simple form and that is what makes it very popular in neural networks. Figure 3.16 shows the hyperbolic tangent function.

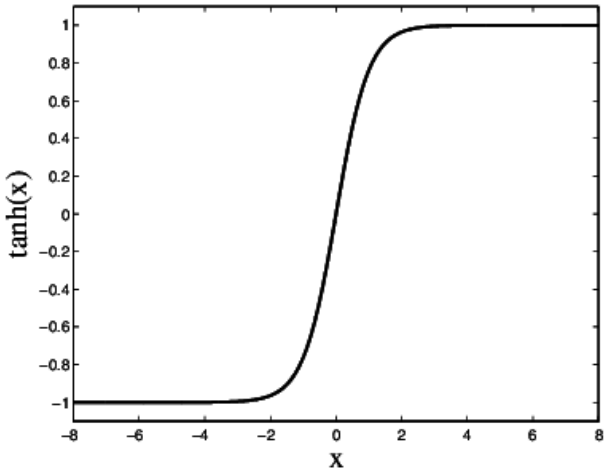

Figure 3.16: Hyperbolic Tangent

https://commons.wikimedia.org/wiki/File:HyperbolicTangent.svg

CHAPTER 4

RANDOM FORESTS

A random forest or a random decision forest is a supervised machine learning technique that works by constructing a number of decision trees during its training phase. Individual decision trees give an output and the result that a classification or a regression random forest makes is that made by most decision trees. Random forests are famous for reducing variance through their use of an estimated prediction function called bootstrap aggregation [32].

4.1 Decision Tree

A decision tree is a tree-like model of decisions. It is a machine learning algorithm that can be used for both regression and classification tasks and thus we can have classification and regression trees respectively. As the name suggests, they are represented by tree diagrams drawn upside-down with the root at the top. The root or top node has branches which also branch out in to other nodes and thereby forming a tree like structure.

Nodes of a decision tree are of three types as shown by Figure 4.1

1. A chance node, which holds probabilities that a certain event has of occurring and is annotated by the shape of a circle.

2. The second type of node is a decision node, the top most of which is the root node. It holds the decision to be made and is annotated by the shape of a square.

3. The last node on a decision tree is the end-point node or the leaf node, there are no splitting that can happen in it and it gives the outcome of a tree.

Decision trees are most efficient in their simplest form, however due to the presence of large amounts of data simple decision trees are near impossible to create and use. The best way to go around this is choosing important features or attributes to split when building our tree. This brings us to the concept entropy, a measure of randomness or order of a system.

For a given dataset (S) with possible values $\{x_1, x_2, ..., x_n\}$, entropy $H(S)$ is given by:

$$H(S) = \sum_{i=1}^{N} -p(x_i)\log_2 p(x_i) \tag{4.1}$$

Where x_i is one possible event, $p(x_i)$ it's probability, 2 the base of the logarithm used and the equation gives its certainty, the negative summation of probability times the log of probability of event x_i.

When $H(S) = 0$ there is no randomness in the data and the probability of getting a certain outcome of x_i is 100 per cent. Upon an introduction of another attribute to an event, there will be a change in the entropy and a metric which measures this change is called information gain (IG). It measures the decrease in entropy after the data is split and is given by:

Information Gain = Entropy (x) - ([weighted average] * entropy (children for the feature)) Given mathematically as

$$IG(S) = H(S) - \sum_{i=1}^{N} p_i(x)H_i(x) \tag{4.2}$$

Another important metric is the gini impurity. It measures how much an i-th attribute class can be mistaken for being in the wrong class and therefore used to split data within the decision tree. This way it ensures that when a node is split in a tree, the data it splits does not contain a wrong attribute (impurity in the data set). It is given by

$$\text{Gini(S)} = 1 - \sum_{i=1}^{N} p_i(x)^2 \tag{4.3}$$

Gini impurity and information gain both split a given data set's root node initially and then the technique is applied iteratively on the produced leaves of the tree. The depth of the algorithm in splitting the nodes depends on the type of problem at hand. Feature or attribute importance is also important in decision tree nodes, branches which show not to be very useful in the tree are cut off from the tree with a technique called pruning.
At the top of the tree is the root node, in it is a lot of randomness in data as that's where all the input data is first fed in. Entropy is very high in the root node and gradually decreases in split datasets as the tree keeps branching out. A tree is grown in the following steps

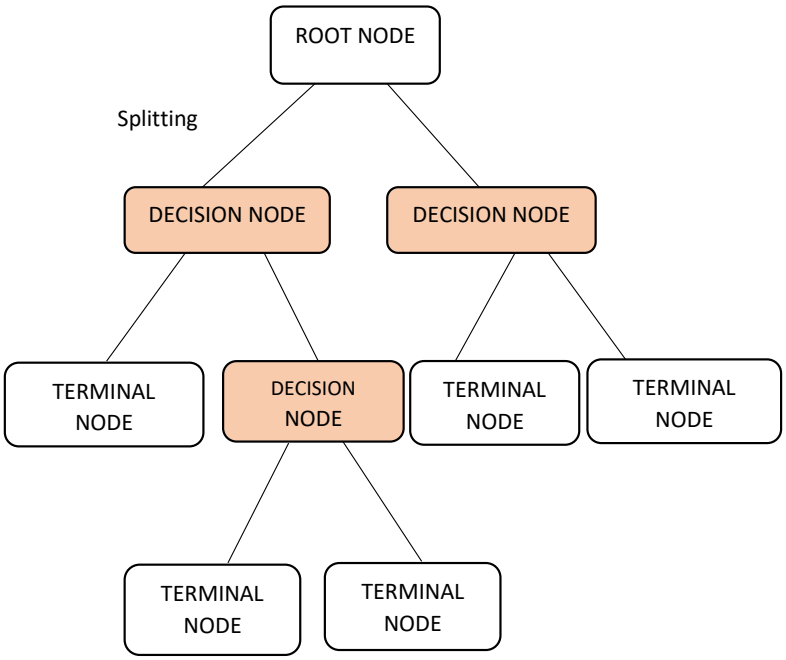

Figure 4.1: Decision Tree

1. The first assumption is that several cases in the training set is N and a sample of these N cases is taken at random with replacement. This is the sample for growing a tree.

2. Then a supposition of the number of input variables or features is done, M input variables or features. A number $m < M$ is also specified such that at each node, m variables are selected at random out of the M. The best split on these m input variables is then used to split the node. This value of m is held constant while we grow the tree.

3. Each tree is grown as far deep down to the largest extent possible.

4. The prediction on new data is then done by wholly looking at the predictions of the n trees, the majority.

Regression trees are used when the dependent variable is continuous and the value obtained by the terminal node in the training data is the mean or average response of the observation. Classification trees are used when the dependent variable is categorical and the value or class

obtained by the terminal node in the training data is the mode of observation.

Techniques are employed to improve a decision tree's characteristics for better performance speeds and more accuracies. These are bootstrapping, random forest ensemble and boosting.

4.2 Bootstrap Method

It is a statistical method for estimating a quantity such as a mean or a standard deviation from a data set. Rather than obtaining the variability, and therefore the error from independent data sets of a population, one set is instead used from the population and continuously sampled. A comparison of the different outcome variables like the mean obtained from the different sampling, can piece together how much variance a dataset has and thereby achieve better accuracies can be achieved in estimating the answer. James. G [33] showed that a bootstrap method works in the following way;
1. An assumption of a training sample set of values is made on which we aim to get an estimate of its mean.
2. An assumption that several random sub-samples of the data are taken with a replacement in each sample.
3. The mean of each sample is calculated.
4. An average is taken over all the collected means and is used as the estimated mean data.
An application of this method to bagging gives result to another technique called bootstrap aggregation.

4.3 Bootstrap Aggregation (Bagging)

This type of ensemble method tries to form more accurate predictions by using a special technique of combining predictions from other machine learning algorithms. This technique is used in models with a high variance like decision trees. A model has a high variance when predictions made by the same tree trained on completely different subsets of data can produce different predictions. The bagging algorithm aims to reduce this variance in models in the following way;
1. It creates as many random sub-samples of data as possible with replacement at every step. This means data is split up data into smaller sub-samples where the same attribute can be selected a multiple number of times.
2. The classification or regression tree models are then trained on each sample of the model. That is a decision tree algorithm is applied to each sample.
3. With the tree formed, a new dataset is applied to each tree. This way several outcome variables are retrieved and results averaged to obtain the best estimate for the outcome variable.
4. Decision trees are known to produce high variability, an error estimation of bagging is usually calculated using the MSE, like in linear regression.

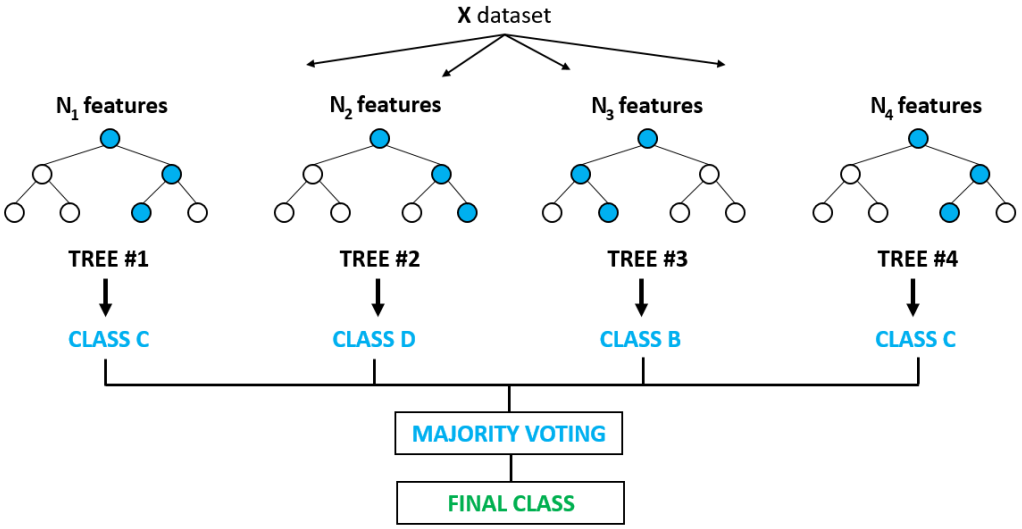

Figure 4.2: Random Forest

http://www.globalsoftwaresupport.com

4.4 Random Forest Ensemble

Often when the algorithm is applied to different sets of similar data, the trees may be like each other. This portrays the greediness of decisions tree algorithms in trying to minimize the cost and maximize information gain. The same classifiers are chosen at each node and it becomes an issue as large variance affects the outcome variable with many homogeneous trees [33].

The random forest ensemble technique improves the bagging method by finely adjusting the learning algorithm. The algorithm selects n random subsets from the training set and then trains n decision trees at each node. It continues the search of the best classifier for this sample using either information gain or gini impurity. This process ensures that the individual trees created will not be the same, thus decorrelating the outcomes of each tree. An application of the bagging technique on these trees results in a better average result.

In a random forest therefore, a subset of features of data is randomly selected and the best split feature from the subset is used to split each node on a tree in which the training is done on. A different subset's features are trained on another tree and the best split feature from this subset also chosen to split respective node. The majority outcome that is cast by different trees is taken as the final result for the random forest. A random ensemble is shown in Figure4.2

- For a regression task, the number of features a tree must search over to find the best feature is $1/3 * p$ where p is the number of predictors and the value obtained by the end-point node

is the mean of the responses gotten by all the other trees.

- For a classification task, the number of features a tree must search over to find the best feature is $\sqrt{p}$ and the value obtained by the end-point node is the mode of the responses gotten by all the other tree in the forests.

Applications of random forests include financial sectors to sieve through loyal and fraud customers and in medicine for identifying and diagnosing diseases. Random forests are also used in search or recommendation engines to identify what consumers or are up to date with and what they are likely to buy, watch or listen to. They are also used for image and voice classification tasks in games like X-box. The three most popular random forest applications are in remote sensing, object detection and the game of Kinect.

4.5 Out of Bag Samples

An important feature of random forests is its use of out of bag (OOB) samples techniques. For each training sample x_i a random forest constructs a random predictor by averaging only those trees corresponding to bootstrap samples in which x_i did not appear. An OOB error estimate is almost identical to that obtained by N-fold cross validation, hence unlike many other non-linear estimators, random forests can be fit in one sequence, with cross-validation being performed along the way. The training phase is terminated once the OOB error stabilizes. An example is a case where say a 1000 trees were used and only 100 would have been enough.

4.6 Boosting

Boosting is a technique that combines many weak classifiers to form one stronger classifier. Rather than training many different classifiers and combining the results, model is built from the training data, then a second model is created that attempts to correct the errors from the first model. More and more models are created until the training set is predicted perfectly for a maximum number of models added. A successful boosting algorithm is AdaBoost and is best used to boost performance of decision trees in binary classification problems.

Models that achieve accuracy just above random chance on classification problems are called weak learners examples of which are decision trees. A weak classifier is prepared on the training data using the weighted sample. Each instance in the training data set is weighted where the initial weight is set to

$$Weight(x_i) = \frac{1}{N} \tag{4.4}$$

where x_i is the i_{th} training instance and n the number of training instances. The process continues until a preset number of weak learners have been created or no further improvement can be made on the training data. These are then sequentially trained using the weighted training data [34].

CHAPTER 5

METHODS AND RESULTS

5.1 Data Sets

To build a neural network from scratch, the MNIST dataset of handwritten numbers was used and as a way to simulate the classification and prediction capabilities of an artificial neural network, the Titanic dataset was used to classify survival probabilities of passengers on the boat. From this we would be able to have a prototype artificial neural network architecture for the analysis of type 1 diabetes Agincourt verbal autopsy data.

5.1.1 Agincourt Verbal Autopsy

The verbal autopsy dataset analyzed by neural networks and random forest classifiers is from the Agincourt Health and Socio-demographic Surveillance System (HDSS). HDSS was established in 1922 to support district health systems development that were led by the post- apartheid ministry of health. It is located in the rural parts of the north east South Africa near the Mozambique border. It provides the foundation for the Rural Public Health Transitions Research Unit of the Medical Research Council and University of the Witwatersrand South Africa (the MRC/Wits-Agincourt Unit). Agincourt supports a number of investigations into causes and impacts of diseases to populations and social transitions [17].

The verbal autopsy dataset used was that collected between 1992 and 2015 and the following selection criteria was made in selecting it.

- The World Health Organization 2012 age selection standards were used and a range of 1-49 years was the scope of focus.

- A reviewing physician upon agreement with clinician colleagues who specialize in adult and internal medicine, (Dr Alisha Wade, Prof Justine Davies and Prof Miles Witham) made a selection of features in regard to their likeliness to be a symptom of diabetes, hyperglycemia or diabetic ketoacidosis.

- The classes are positive cases, where diabetes is agreed to be the main cause of death and the negative cases which are cases not displaying any of these features and therefore show death to be due to other causes.

- A total of ten features make up the final dataset for analysis and table 5.1 shows the two classes with the features.

Table 5.1: Agincourt diabetes features

Feature	Feature Type	Description
Female	Binary	The sex of the deceased
Tuber	Binary	The deceased had tuberculosis
Diabetes	Binary	The deceased had diabetes
Men-con	Binary	The deceased had signs of mental conditions
Cough	Binary	The deceased used to cough
Ch-Cough	Binary	The deceased coughed over a long period of time
Diarr	Binary	The deceased had diarrhoea
Exc-Urine	Binary	The deceased urinated a lot
Exc-Drink	Binary	The deceased was excessively thirsty
Age	Numeric	The age of the deceased when they died
Class	Binary	Whether the deceased died of diabetes or not

5.1.2 Titanic Dataset

The Titanic dataset is introduced to simulate the binary classification capabilities of a neural network and build a neural network prototype that will be used as a baseline for building one for Agincourt dataset.
Kaggle is an online community of data scientists and machine learners, owned by Google, Inc. It was founded in April 2010 and it allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers [35]. The titanic data set is one such file from Kaggle and consists of 891 entries of training data and 418 entries on test data with 10 columns. The features are: Passenger ID, Survived, PClass, Parch, Siblings/Spouse which are integer type of format. Age and Fare which are the float type. Cabin, Embarked, Name, Sex and Ticket which are object type. Table 5.2 gives a descriptive summary of these features.

Table 5.2: Titanic dataset features

Feature	Feature Type	Description
Passenger	Integer	The identity number of the passenger
Survived	Binary	Whether the passenger survived or not
Pclass	Integer	The class the passenger is in
Name	Object	The name of the passenger
Sex	Object	The sex of the passenger
Age	Float	The of the passenger
SibSp	Integer	Number of Siblings and spouse of the passenger abroad
Parch	Integer	The deceased had diabetes
Ticket	Object	The ticket number of the passenger
Fare	Float	The amount the passenger paid
Cabin	Object	The deceased coughed over a long period of time
Embarked	Object	Where the passenger embarked

5.1.3 Hand Written Digits Data Set

The MNIST is a database of handwritten digits that was constructed from NIST's Special Database 3 and Special Database 1 which contain binary images of handwritten digits. It consists of 60 000 training images and 10 000 testing images each of sizes 28 X 28 [36].

5.2 Methods

5.2.1 Building a neural network from scratch

As the first step to mimicking the thinking process done by the brain's neurons connected by synapses, a neural network was built through matrices. Matrices are a mathematical technique that use a grid of numbers. This neural network was aimed to classify handwritten digits from the MNIST dataset's training images and testing images.

5.2.1.1 Libraries

To build a neural network from scratch, the NumPY [37] library was used.

5.2.1.2 Data Preprocessing

The first data preprocessing step was breaking each of the images containing many digits into a sequence of images containing just a single digit to enable the classification of a single image separately.

Next, both the training and testing data sets were split into input and labels values, the labels were further transformed into a one-hot-encoding; a vector of binary values to represent numeric values. The input values were reshaped into the handwritten images are 28 X 28 pixels and were flattened out into a one-dimensional vector of pixel size 784.

5.2.1.3 Neural Network

A feedfoward neural network was chosen whose training process would take place like as in Figure 5.1. The first step was choosing the number of layers the neural network was to have. For a simple network we intended to build this was three: the input layer, the hidden layer and the output layer which were created in the following way;

A dot product of input values, the $28x28 = 784$ dimensional vector and the first set of weights is added to a bias, the sum is then passed through an activation function which does a non-linear transformation on the sum to enable learning. This is the output of the first layer which forms an input of the hidden layer which repeats the process of taking the dot product of the input with the weights of that respective layer and adding the product to the bias of that layer and passing the sum though another activation function. The number of neurons in which these dot products, summations and activations are to take place were arbitrarily chosen for the selected for the hidden layer.

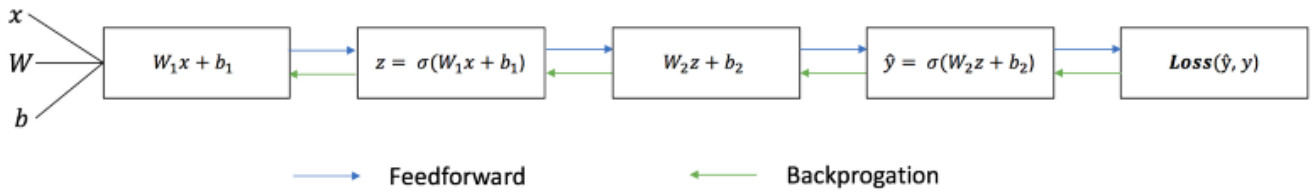

Figure 5.1: Training of Neural Network

The next step was choosing the number of layers the neural network was to have.

- The input layer takes the input values which were reshaped into the handwritten images of 28 X 28 pixels and flattened out into a one-dimensional vector of pixel size 784.
- The number of neurons in the hidden layer was arbitrarily chosen and experimentation with different values done.
- The total number of classifications is 10 digits, the output layer was therefore built with 10 neurons.

5.2.1.4 Parameter Initialization

The neural network class was created by initialization of a set of weights w that will be between the input and hidden layer and the ones between the hidden layer and the output layer.

5.2.1.5 Training

The training process as depicted in Figure follows in these steps:

1. A forward propagation was passed which is multiplying input x with weight w, adding a bias b and then applying a sigmoid activation function σ because the classification is binary.

2. The output from the first layer is multiplied by the weight w_2 of the next layer and the bias b_2 added to give a sum which is also multiplied by an activation function to give an output.

3. The loss is computed between the actual value y and the predicted $\hat{y}$ value and the slope of the cost function (the one we're trying to minimize) is computed.

4. The weights are updated and training is continued until the cost function stops converging. Convergence of the cost function is checked by comparing the weights of the previous run to those of the current run.

5. If their difference is smaller than a very small (arbitrarily set number number 10^-8) then convergence is happening.

Prediction was defined as the maximum of the values that come as a result of the forward pass i.e the node which had the highest value in the range 0 to 1.

To evaluate the performance of the model, values predicted by the neural network that matched the actual target values were divided by the total number of data points to get an accuracy. At an interruption after 10482 runs, the model's accuracy was 87%. This is the number of test samples that the model classified correctly out of all the classifications made.

The sigmoid activation function was chosen because it exists between 0 and 1. It is especially used for models where prediction is a probability, since probability of anything exists only between the range of 0 and 1. The sigmoid activation function is given by

$$\sigma = \frac{1}{1 + e^{-\sum w_j x_j - b}} \tag{5.1}$$

Where x_j is the j_{th} input variable and w_j the j_{th} weight the sum of which subtracts the bias b. The individual neurons making up layers of the neural network's outputs $\hat{y}$ are determined by

whether the sum of the product of the weights and inputs $\sum w_j x_j$ is above or below a threshold value; which is a real number.

$$\text{output} = \begin{cases} 0 \text{ if } \sum w_j x_j \leqslant \text{threshold} \\ 1 \text{ if } \sum w_j x_j > \text{threshold} \end{cases} \tag{5.2}$$

A perceptron's single output is defined as $\sum w_j x_j = \mathbf{w} \cdot \mathbf{x}$ where $\mathbf{w}$ and $\mathbf{x}$ are vectors whose components are weights and inputs respectively.
The threshold value can also be moved to the other side of the inequality and replaced by a perceptron's bias $\mathbf{b} = -\text{threshold}$. Bias is a measure of how easy it is to get a perceptron to output a value of 1 or to fire and using the bias instead of the threshold. The perceptron's output is given as

$$\text{output} = \begin{cases} 0 \text{ if } \mathbf{w} \cdot \mathbf{x} + \mathbf{b} \leqslant 0 \\ 1 \text{ if } \mathbf{w} \cdot \mathbf{x} + \mathbf{b} > 0 \end{cases} \tag{5.3}$$

To quantify how good the neural network is at doing the classification, the error function/cost function which defines the error between the desired output y and the computed output of the network on input x used was given by

$$L(y, \hat{y}) = \frac{1}{2N} \sum_{i=1}^{N} (y - \hat{y})^2 \tag{5.4}$$

where $\hat{y}$ is the output value from the model and y the target value and N is the number of training examples.

5.2.2 Neural Network Classifier for Titanic Dataset

To identify the classification and prediction capabilities of our artificial neural network, the Titanic dataset was used to classify survival probabilities of passengers on the boat.

5.2.2.1 Libraries and Modules

Anaconda 4.3.27 The Anaconda Navigator is a graphical user interface that was used to launch applications, manage the library packages and environments and the neural network was created in its Jupyter Notebook.
Python 3.6 Keras 2.0.8 with a TensorFlow backend NumPY Pandas Scikit-learn

5.2.2.2 Data Preprocessing

- Missing values are shown to affect the overall performance of artificial neural networks because artificial neural networks cannot interprete them. As the first preprocessing step, the structure of the dataset was checked to identify any missing values. Categorical values were then converted to numeric ones that machine learning algorithms could process. This was done in the following way;
- Age, Embarked and Cabin columns were identified to have missing values and the Cabin column was dropped as more than half of its values were missing.
- For the Age column an array of random numbers was created based on the mean age value as filled in for the missing values.
- The Sex column was converted to numeric ones where '0' was filled for males and '1' for females.
- The second step of preprocessing the data was to split it into training and testing sets, then input and target features where the survival column is the target and the other columns input features.
- Certain features hold more meaning than others and therefore contribute more to the classification task more than others and lead to the first step of the preprocessing; dropping unnecessary fields. This included the columns; "Fare", "Ticket", "Name", "Cabin", "Age", and "Embarked". These features bear very little or no effect on the survival of individuals.
- The remaining columns after dropping some features were set as the input variables and the surviving column as the target label. To make the dataset presentable, that is to start with "Passenger Id", "Sex", "Sibling and spouse (Sibsp)",..., sorting of the columns was done. Shuffling or randomizing of the data was also done so that the order of our data does not affect the learning process. Peralta et al. [38] has shown that it is extremely important to shuffle the training data to avoid obtaining mini batches of highly correlated values. He further showed that different random orderings will have slightly different performances from each other, but this is by very insignificant factor.
- Some features hold a higher weight in making the prediction and the classification more than others, these features were also identified and optimized in the neural network building.

5.2.2.3 Choosing the Model

Different models suit different datasets; while some are better suited for images, some are well suited for sequential data like words, sentences or music and others are well suited for numeric

data. The titanic dataset, is mainly numeric after the preprocessing step, six features were picked as input values and a linear model chosen. A linear model makes predictions and or classifications by using a linear function of the input features.

A sequential model, which is a linear stack of layers called forth sequentially in the neural network build up was chosen and layers were added in a series. The input size and activation function to be used and for the first layer the input shape was specified an inference from which the layers that followed would make. Different sizes (number of nodes in each layer) and architectures (number of layers and types of activation functions for each layer) were experimented with as to seek for parameter values that would give the neural network optimum score.

- For the first layer the input dimension argument was set to 5, the number of neurons in this layer was set to 64 and the activation function with which the classifier showed a better performance was ReLU. The same selection procedure followed for the next two layers for which 6 neurons were selected. The activation function selected for the output layer was sigmoid to ensure that the output is between 0 and 1 and one neuron was specified to predict the survival class.

- To configure the learning of the model the binary cross entropy cost function was chosen as the problem is a binary classification problem. The adam optimizer was chosen because it quickly achieves good results and is the default recommended algorithm to use.

- The training data was next fitted into the model in batches of 10 and 200 iterations per training process.

- The model's performance on test data was next evaluated, and the model was able to do correct predictions on 80.68% of test data. The correct classifications for the two passengers, Winslet (who survived and is of class 1) and DiCaprio (who did not survive and is of class 0) are given by Figure 5.2.

- As a way to gradually tune the parameters of our model to find the best, the hyperas [39] library was used.

- To see which features have more contribution on the output the feature importance function was used and it gives the weights of the features in ranked order of highest weight. The features are given in Figure 5.3.

```
#Predicting
# Creating some data for DiCaprio
DiCaprio = np.array([[3, 0, 0, 0, 5.0000,]])


# Predict surviving chances

pred = model.predict_classes(DiCaprio)

pred
```

```
array([0])
```

```
# Creating some data for Winslet
Winslet = np.array ([[ 1, 1, 1, 2, 100.0000,]])

# Predict surviving chances

pred = model.predict_classes(Winslet)


pred
```

```
array([1])
```

Figure 5.2: Titanic Predictions

Weight	Feature
0.0108 ± 0.0013	diabetes
0.0044 ± 0.0338	death_age
0.0027 ± 0.0027	cough
0.0026 ± 0.0049	exc_drink
0.0026 ± 0.0024	exc_urine
0.0007 ± 0.0012	tuber
-0.0003 ± 0.0017	diarr
-0.0006 ± 0.0006	ch_cough
-0.0014 ± 0.0008	men_con
-0.0038 ± 0.0052	female

Figure 5.3: Artificial Neural Network Feature Importance

5.2.3 Random Forest Classifier for Agincourt Dataset

5.2.3.1 Libraries and Modules

Anaconda 4.3.27
Python 3.6
Scikit-learn [40]

5.2.3.2 Data Preprocessing

The first data pre-processing step was identifying the structure of our dataset. The positive cases were located and 79 of them identified. There is great class imbalance in the dataset, with very

little positive and a large number of negative cases. The data was next divided into input and target variables x and y.

5.2.3.3 Building the Model

The ensemble technique of bootstrap aggregation (bagging) was used to bring out the randomness in the data. One decision tree for this set of data would look like the one in Figure 5.4.

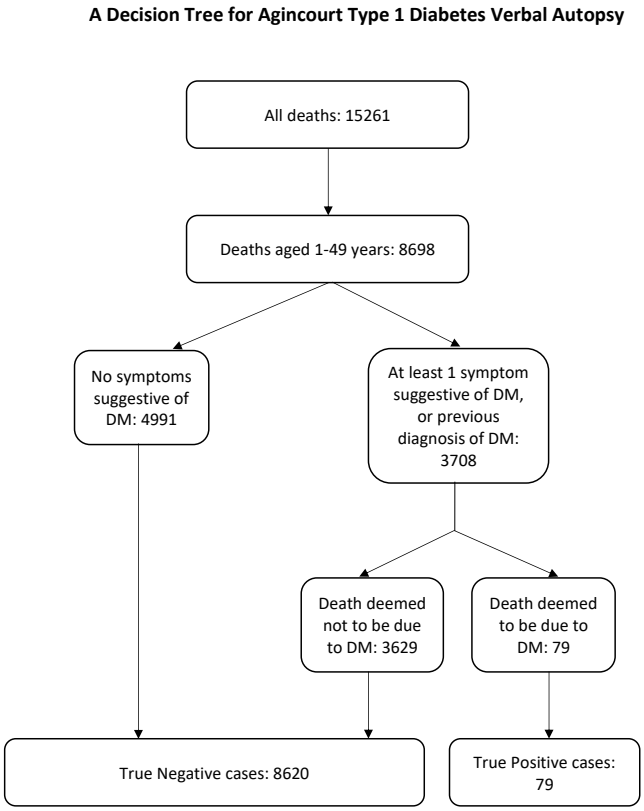

Figure 5.4: Decision Tree for Agincourt Type 1 Diabetes

A random search to find optimal parameters was done with the Python library, hyperas. Hyperas replaces the exhaustive listing of all combinations by selecting them randomly. This step is called hyper-parameter tuning and the parameters being tuned are the number of layers and neurons, best activation functions out of linear, reLU, tanh and the best optimizer to use among adam, rmsprog, sdg shown in Table 5.3.

Table 5.3: Random Forest Parameter Test

Parameter	Description
n-jobs	This parameter lets computer know how many processors it is allowed to use, the selected value was 1
n-estimators	This parameter specifies the number of trees in the forest, 25 gave the optimal score of the model
max-depth	This parameter represents the depth of each tree in the forest, the value of 4 gave the optimal score for the classifier
min-samples-leaf	This parameter is the number of samples that go into new leaves and the value that gave an optimal score for the classifier was 28
max-leaf-nodes or max-features	The parameter represents the maximum number of features considered when looking for the best split to do, the value of 5 gave the classifier the maximum score
OOB-Score	A cross-validation method that takes all observations used in the trees and finds the maximum score for each observation based on the trees which did not use the observation. This was set to true

Figure 5.5 graphically shows the optimal values of the parameters for a random forest classifier.

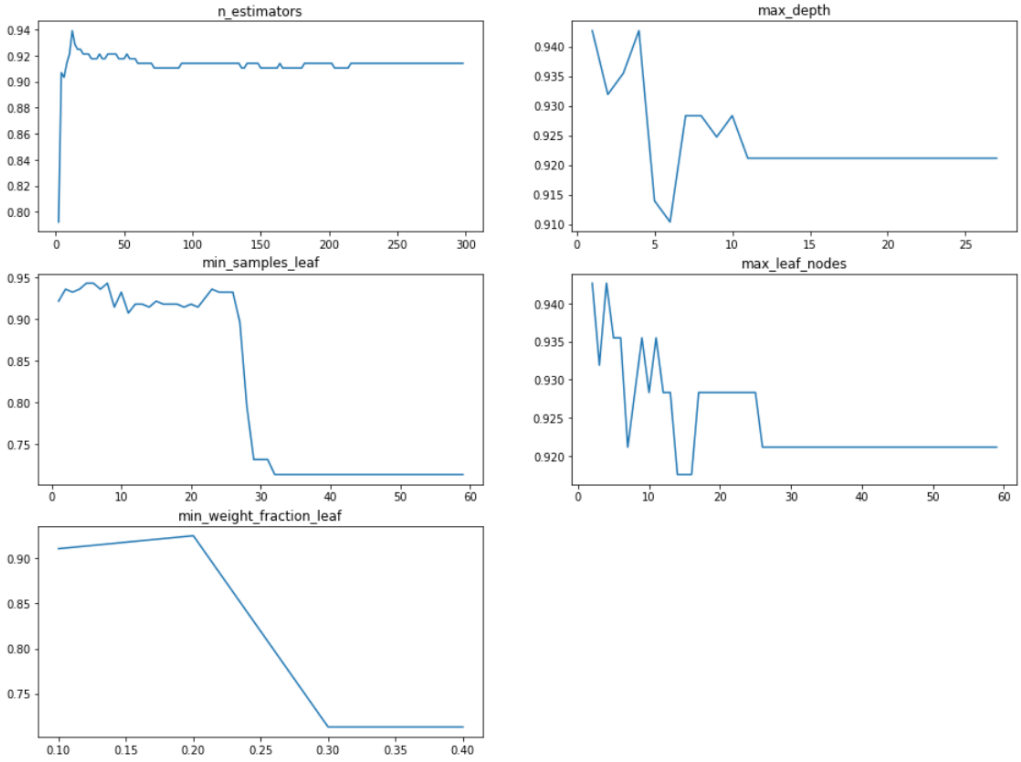

Figure 5.5: Random Forest Parameter Test

After the optimal parameter values were identified, the classifier was configured with these and then fitted with these values for x and y.

5.2.3.4 Variable Importance

Of the greatest characteristics of random forests is their ability to scale each of the features' importance when it predicting or making the right classification by the model. It looks at how much the tree nodes of the trees which use the respective feature reduce in their impurity across the forest. Figure 5.7 below graphically shows the feature importance.

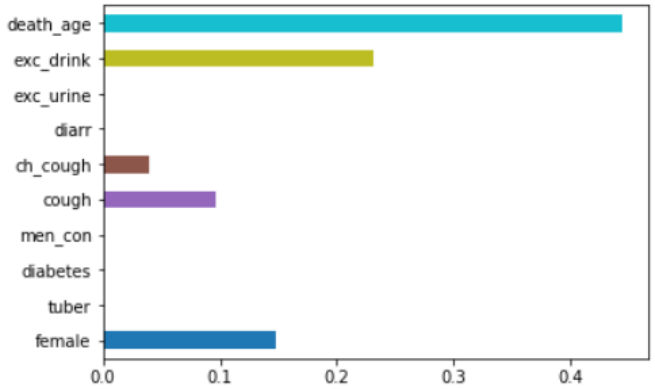

Figure 5.6: Random Forest Feature Importance

5.2.4 Model Evaluation

After building and training machine learning models the next step was to evaluate their performance in classifying data. Different machine learning algorithms use a wide range of different performance metrics depending on their task. For the classification task, the metrics used were accuracy, confusion matrix, Receiver Operating Curve (ROC) and Area Under the Curve (AUC), Precision and Recall and their curves.

5.2.4.1 Accuracy

This metric measures the number of times correct predictions are made by the classifier. It is given by diving the number of correct predictions by the total number of predictions.

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions} \tag{5.5}$$

- For building a neural network from scratch, the number of correctly predicted images to the total number of images in the data set was given by an evaluation function that took in as inputs the test variables. The accuracy value obtained was 87%.
- In the neural network for the Titanic data set, the number of correctly classified passengers to the total number of predictions of passengers on the boat as taken to calculate accuracy.

The evaluation function was used to get this metric on the test data and a score of 80.68% obtained.

- The neural network for Agincourt verbal autopsy data's accuracy was taken as the number of correctly classified cause of death to the total number of deaths of patients. The model's evaluation function was used on the test data and a score of 99.13% was obtained.
- Random Forest for Agincourt verbal autopsy data's accuracy is the number of correctly classified cause of death by type 1 diabetes to the total number of deaths of patients. To get this value, the classifier's prediction function was used and the score was 85.97%.

While accuracy worked well to give us the performance of the neural network for titanic dataset, the same cannot be said for the diabetes dataset. This is due to the fact that diabetes dataset is skewed;too many negative cases and less positive cases. This says to us the model cannot be 99.13% accurate in classifying the positive cases as there are only 0.9% positive cases but the negative cases. However to make a simulation of how well the model would have performed given an even distribution of equally negative cases to the positive cases was used and a score of 95.64% was obtained. This however is not satisfactory as it does not generalize to the overall dataset but to only a subset. The alternative was to use a confusion matrix.

5.2.4.2 Confusion Matrix

To make distinctions between the classifications made for the two classes a confusion matrix was used. It helps us to see how many samples were correctly classified and how many were not. Class 1 (Survived the boat crash/ Died of diabetes) and Class 0 (Did not survive the boat crash / Did not die of diabetes) respectively. The confusion matrix gives a summary of the classification of the two classes when a certain value that lets the model decide if an individual died of diabetes or not is set. This value is called a threshold.

5.2.5 Confusion Matrix of Diabetes Neural Network

The confusion matrix function used on test labels and predicted labels and the results as shown in Figures 5.7 and 5.9

5.2.5.1 Precision

This is the ratio between the true positive values and all positive values (sum of true positives and false positives), it is a measure of positive predictions and helps us answer the question, "out of the correctly predicted deaths due to diabetes how many are truly correct?"

$$Precision = \frac{Patientsthatdiedofdiabetes}{TotalNumberofdeathsreturnedbyclassifier} \tag{5.6}$$

NEURAL NETWORK CONFUSION MATRIX

True Positive (TP)	**False Positive (FP)**
Reality: Death by T1DM ML Model: Death by T1DM Number of TP results: **2834**	Reality: Death not by T1DM ML Model: Death by T1DM Number of TP results: **8**
False Negative (FN)	**True Negative (TN)**
Reality: Death not by T1DM ML Model: Death not by T1DM Number of TP results: **14**	Reality: Death by T1DM ML Model: Death not by T1DM Number of TP results: **15**

Figure 5.7: Agincourt Artificial Neural Network Confusion Matrix

The precision score function was used on the test labels and the predicted values. The classifiers returned 0.8 for the neural network and 0.96 for the random forest classifier. This says that when the model predicts that patients died of diabetes, it is correct 80% of the time for the neural network classifier and 96% of the time for the random forest classifier.

5.2.5.2 Recall

This is the ratio of the true positive to both the true positive and false positive values, i.e a performance measure of the whole positive part of the dataset. Recall answers the question, ”what proportion of the actual positives was correctly identified?”

$$Recall = \frac{Patients who died of diabetes}{Total number of deaths} \tag{5.7}$$

The recall score function was used on the test label data and the predicted outputs and the value returned was 0.28 for the neural network and 0.85 for the random forest classifier.This says that the neural network identifies 28% of patients who died of diabetes while the random forest classifier identifies 85% of the patients who died of diabetes

.

RANDOM FOREST CONFUSION MATRIX

True Positive (TP)	**False Positive (FP)**
Reality: Death by T1DM ML Model: Death by T1DM Number of TP results: **8618**	Reality: Death not by T1DM ML Model: Death by T1DM Number of TP results: **1**
False Negative (FN)	**True Negative (TN)**
Reality: Death not by T1DM ML Model: Death not by T1DM Number of TP results: **12**	Reality: Death by T1DM ML Model: Death not by T1DM Number of TP results: **67**

Figure 5.8: Agincourt Artificial Random Forest Confusion Matrix

The Precision-Recall curves of artificial neural network and a Random Forest model for the diabetes dataset over a set of different thresholds are shown by Figures 5.9 and 5.10.

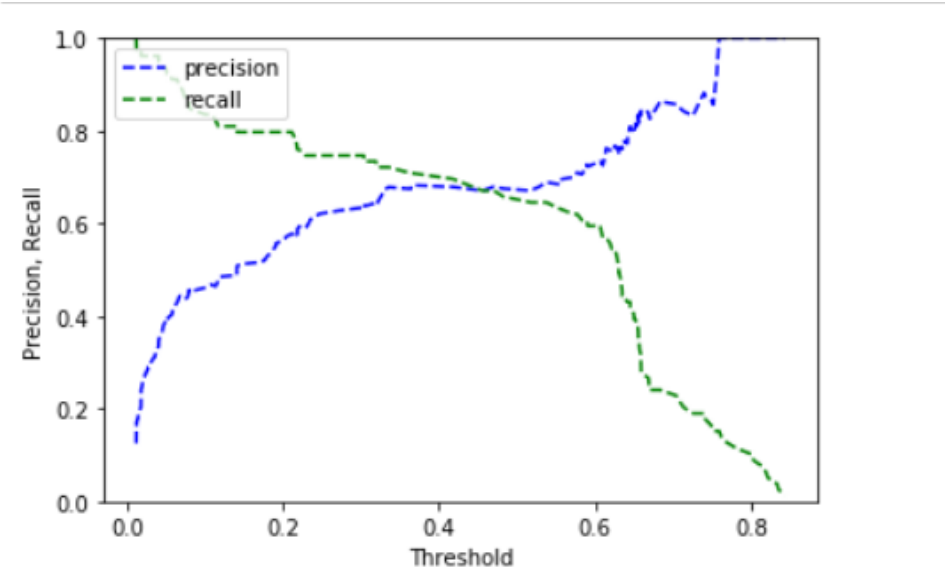

Figure 5.9: Random Forest Precision-Recall Curve

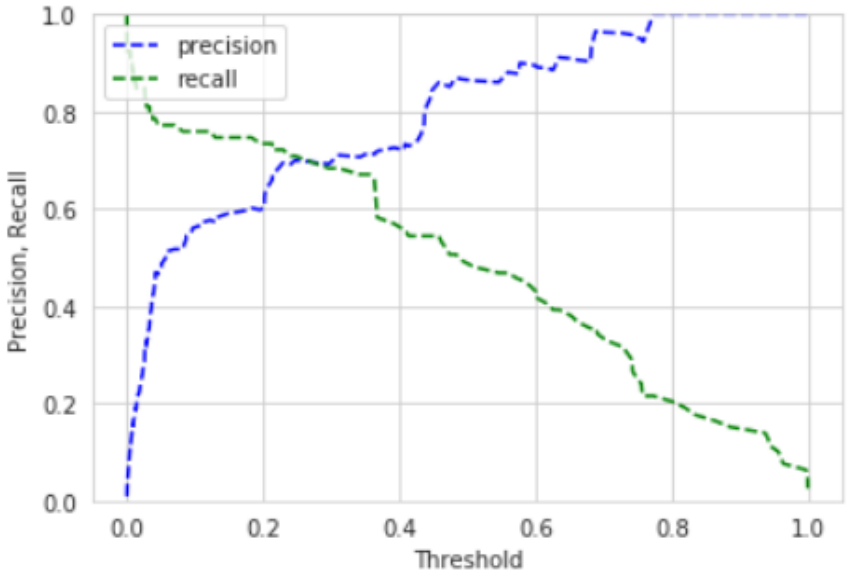

Figure 5.10: Neural Network Precision-Recall Curve

5.2.5.3 ROC and AUC

The receiver operating characteristic curve (ROC) is a graphical way of showing or visualizing the performance of a binary classifier across its varied discrimination threshold, while area under the curve (AUC) gives the percentage value of the area under this curve. This percentage value gives a summary of the classifier's performance;the expected proportion of positives ranked before a uniformly drawn random negative.

5.2.5.4 Functions

i) The **ROC CURVE** function takes the true outcomes of the 0 class and the 1 class from the test (oob sample in the case of random forests) and the predicted probabilities for the 1 class. It returns true positive rates and true negative rates at a given threshold value. The neural network and random forest ROC CURVES are given by Figures 5.11 and 5.12.

ii) The **ROC AUC score** function takes the true outcomes of the 0 class and the 1 class from test (oob sample in the case of random forests) and the predicted probabilities for the 1 class and returns the the AUC score between 0 and 1. They were 0.99 for the random forest Classifier and 0.83 for the neural network respective. The ROC curves show the false positive rate against the true positive rate.

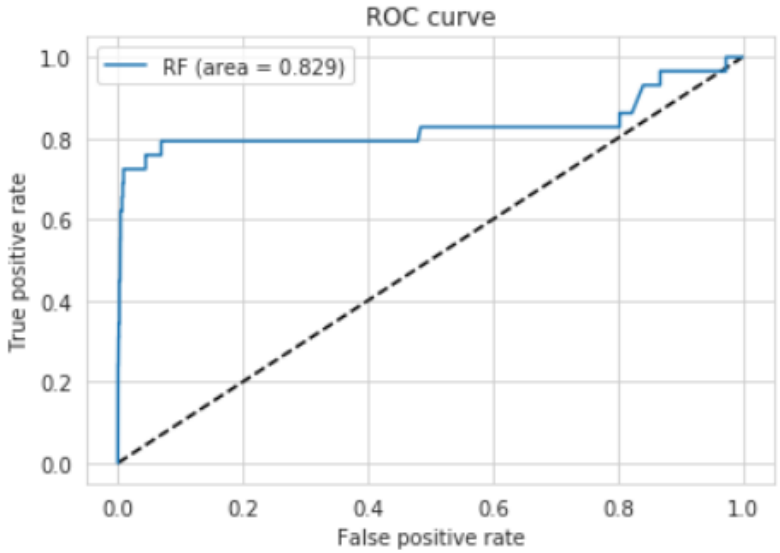

Figure 5.11: Neural Network ROC Curve

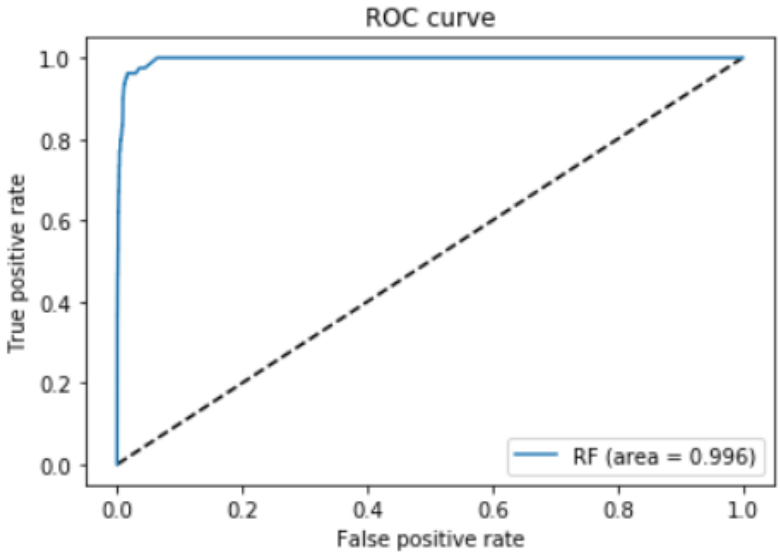

Figure 5.12: Random Forest ROC Curve

5.2.5.5 F1-Score

To fully evaluate the effectiveness of the model, we also focused only on patients for whom the probability score was high (these are the samples that the model generalizes very well to). The behaviour of the precision and recall is observed when a number of these (threshold) is changed. A single value that summarizes the precision-recall curve behaviour is the F-1 score, which is the

harmonic mean of the two.

$$F_1 = \frac{precision \cdot recall}{precision + recall} \tag{5.8}$$

and it was found to be 41% for the neural network and 91% for random forest classifier.
The model with a high F1 score performs better than one with a lower F1 because a higher value of the harmonic mean, means both precision and recall are high. Our interest is in making a correct prediction of the positive class more than of the negative class as this will let us know how many people actually died of the disease. Now neural network identifies only 28% of people who died of type 1 diabetes and when it does it is correct 80% of the time. A random forest classifier on the other hand identifies 85% of the patients who died of the disease and when it does it is correct 96% of the time. This says a random forest classifier is a better classifier for this data set than a neural network classifier. The performance metrics of the two classifiers on the Agincourt dataset are summarized in Table5.4.

Table 5.4: Performance of Classifiers

Classifier	**Precision**	**Recall**	**F1 Score**	**AUCROCCurve**
Neural Network	0.80	0.28	0.41	0.83
Random Forest	0.96	0.85	0.90	0.99

A comparison of the number of positive and negative cases as predicted against the study-physician of random forest, artificial neural networks and weighted score and InterVA4 algorithm is given by Table 5.7.

Table 5.5: Comparison of the numbers for each cases predicted by the different models against the Physician Certified case

Method	**Number of Negative Cases**	**Number of Positive Case**
Study Physician	8620	79
Random Forest	8621	78
Neural Network	8633	66
Specialist Weighted Score	8565	132
InterVA4	8639	58

CHAPTER 6

DISCUSSION

The research study was focused on using machine learning algorithms to identify patients whose cause of death was type 1 diabetes, that is, to determine if a cause of death as had been previously diagnosed by a clinician specialist in diabetes could be identified in the same data from Agincourt collected for the period between 1992 and 2015. A comparison of results was done against the weighted score computed from a study on the same dataset by physician specialists in the disease. As depicted by results, the number of deaths as predicted by their algorithms were in good agreement to those found using the artificial neural networks and random forests algorithms respectively.

The study was to also serve in unveiling insights by especially identifying symptoms or features of this disease that had a larger contribution to death. Both machine learning algorithms show that these features can be grouped in rank of 'importance'. Machine Learning algorithms, both artificial neural networks and random forest classifier were able to identify closely the importance of these features regarding their theoretical severity regarding the disease-causing death [41]

Machine Learning algorithms are powerful for making predictions and classifications based on large sets of data and although for this study the sample form which data was gathered is not very large(8000) as is the case with most medical data, results obtained from machine learning algorithms are still able to give insight to what could be observed with a wider range of datasets larger sets of data were used. Bertolaccini et al. [42] and Pasisni et al. [43] have shown that datasets exceeding the ranges of 10, 000 are the ones whose iterative processes work efficiently and effectively on machine learning algorithms.

It is for these reasons that an alternative algorithm of Random Forests was considered. Random

Forests have a way a dealing with imbalanced datasets and this is through their technique of sampling. Chao Chen [44] showed that in rare disease classifications, most of the common machine learning fail because they all aim at minimizing the overall error rate and fail to pay attention the positive class. He showed that random forests have two important characteristics that can tackle this problem of skewed data. Random Forests carry out cost-sensitive learning, where they assign a high cost through weights to the misclassification of the minority class while trying to minimize the overall error[45]. Random Forests also encompass the sampling technique together with the ensemble idea here they either down-sample or over-sample the minority class and sometimes even both[46].

Most real-life data are imbalanced and show skewness is class distributions and a great majority of cases are usually allocated to the class of less interest. As is the case with the verbal autopsy data the number of deaths that are not due to type 1 diabetes and a smaller portion of the cases to the class of utmost interest, deaths due to type 1 diabetes. The Agincourt verbal autopsy data has only 0.9 per cent of its data being cases that correspond to the cause of death being type 1 diabetes. For these kinds of data distributions, the performance metric of accuracy cannot be used as most inductive machine learning algorithms assume accuracy on an entire range of cases and therefore show accurate predictions on the majority cases while performing very poorly on cases of the less frequent class.

The question of which performance metrics to use or was the most appropriate was answered in guidance to the literature review. To balance data Provost F. and Fawcett T.'s proposed to build a hybrid classifier that uses ROC analysis for classifier performance[47]. ROC curves being the most commonly used performance metrics for binary problems in machine learning and being thewere used standard mfeasuror and biology vie in medicinesualization of the agreement between the false positive rate and the true positive rate. The area under the ROC curve of 0.99 for the random forest classifier and the to that of the neural network classifier of 0.94 showing still the random forest classifier to be performing better. These scores, as opposed to accuracy offer us the opportunity to take into account the size of each class when doing the analysis. A roc auc score of 0.94 says the classifier has a 94 per cent ability to rank a randomly selected positive case higher than a randomly selected negative. This means our classifier is good at analyzing classes as positives hold a 1, a value greater than 0 for negatives. Cause of death by type 1 diabetes will be ranked by the classifier higher than cause of death by something else. This says the classifier is good at what it was modelled to do.

In addition to algorithm performance visualization by ROC curves, Precision-Recall curves give more informative insights of the performance of the classification algorithm and are overlapping for an imbalanced set of data. At recall = 1 precision drops to 0.1 for imbalanced datasets and

to 0.5 for balanced datasets as also depicted from our result. Since we were doing a diagnostic test, recall lets us know how well the classifier is at identifying many cases of death due to type 1 diabetes as deaths due to type1 diabetes. A recall of from a random classifier says our classifier is good enough at doing this. Precision of a classifier lets us know how sure we can be that when a random person is predicted by the model to have died of type 1 diabetes, that they had infact died of diabetes. This value lets us know how well our model is at discriminating between deaths due to type 1 diabetes and deaths due to something else.

Although artificial neural networks were used to classify Agincourt verbal autopsy data results obtained from the analysis of a precision of 0.80 and a recall of 0.27 and an F1 score of 0.41 show it to not be the best classification technique of choice for this type of data. The classifier is highly precise but misses a large number of significant cases which it finds hard to classify, i.e a low value of recall. This is caused by a number of reasons, the first is that artificial neural networks need very large sets of data for their training and tend to overfit if datasets are too small.

The second is that the Agincourt verbal autopsy data is skewed and does not follow a normal distribution. Kumar et al. [48] shows that skewness in data variables plays an important role in the predictive accuracies of artificial neural network models. Methods to bring about transformations to skewed data like power transformations, Box-Cox procedures and researches like those of Subba Narasimba [49] [50] [51] were to serve as remedy the issue of skewed data in Neural networks but these fall short due to the fact that they do not consider and explain the effect of the distribution of the target dataset which has the same contribution to the learning of the neural network as the input data even though literature continue to show that the learning process in neural networks is influenced by both input and target data.

In summary two machine learning algorithms were chosen and to make classifications of cause of death due to type 1 diabetes using physician coded verbal autopsy data. These algorithms required in their development prior knowledge and insight into the disease symptoms and features from clinician specialists. This means that to use them in making future predictions, insights on the diseases from clinicians is necessary to help them not to overestimate or underestimate number of class cases. The confidence level of these algorithms for verbal autopsy validation is good in health system deprived areas of developing countries with random forests generalizing particularly well for this type of data distribution.

CHAPTER 7

CONCLUSION

Two machine learning techniques, neural networks and random forests were able to classify type 1 diabetes verbal autopsy data from Agincourt. The number of deaths caused by type 1 diabetes were identified in the same people as those identified by a clinician specialist of the disease. Of the 79 positive and 8620 negative cases determined by a physician specialist, 78 positive and 8621 negative cases were classified by a random forest classifier, 66 positive and 8633 negative cases were classified by the neural network classifier. These values were given by the area under the receiver operating characteristic curve and were 0.99 for the random forest and 0.83 for the neural network respectively.

While these machine learning algorithms used to classify the Agincourt verbal autopsy data have shown to be quite accurate, the performances by the two classifiers unveil that different classifiers are more suited for certain types of data more than others. This is attested to by the random forest classifier giving a high precision and accuracy despite the huge class imbalance of the data which on the other hand greatly affected the performance of the neural network. In these instances we are encouraged to use different evaluation metrics that provide more insight on what the classifiers can deliver. These metrics are the confusion matrix, precision, recall and the f-1 score in conjunction with ROC curves and AUC curves. The scope of the study can be extended and a larger set of data or many sets of data across several developing countries can be used in future to expand the scope of the study to generalize well across a larger horizon. With these results, better and well informed decisions can be made towards advancing the development of the country's health sector.

CHAPTER 8

APPENDIX A

Building a Neural Network from scratch

```
import numpy as np

#Parameter Initialization
class NeuralNetwork:
    def __init__(self, W):
        self.W = W

    def sigmoid(self, s):
        # sigmoid function as activation function
        return (1.0 / ( 1 + np.exp(-s)))

    def forward(self, X):
        #application of activation function and derivatives
        Z  = self.sigmoid(np.dot(X, self.W))
        return (Z)

    def train(self, X, Y, learning_rate):
        # number of classes/labels
        C =  len(np.unique(Y))
        #  number of datapoints
        N = X.shape[0]
        # Vectorication of labels(classese)
```

```
        Ytarget = np.zeros((N,C))
        Ytarget[np.arange(N), Y.astype(np.int)] = 1.0
        is_converged = False
        # number of runs
        epoch = 1
        while not is_converged:
            #  forward propagation
            Z = self.forward(X)
            # Backward propogation
            E = (Z - Ytarget)
            # slope of our cost function
            D = Z * (1-Z) * E
            # compute the mean square error
            MSE = 1.0 / (2 * N) * np.sum(np.sum(E**2, axis=1))
            ACC = self.performance(X, Y)
            print('epoch = {0}, MSE = {1:.3f}, ACC = {2:.3f}'.format(epoch, MSE, ACC))
            # update weight
            w_previous = self.W.copy()
            self.W = self.W - learning_rate * (1.0/N )* np.dot(X.T, D)
            epoch += 1
            # check for convergence
            if (np.max(np.abs(w_previous - self.W)) <= 1e-8):
                is_converged =True

    def predict(self, X):
        Z = self.forward(X)
        Ypredicted = np.argmax(Z, axis=1)
        return (Ypredicted)

    def performance(self, X, Y):
        N = X.shape[0]
        Ypredicted = self.predict(X)
        acc = np.sum(Ypredicted == Y) / N
        return acc

from keras.datasets import mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data();
train_images = train_images.astype(np.float) / 255.0
test_images = test_images.astype(np.float) / 255.0
```

```
train_images = train_images.reshape(-1, 28*28)
test_images = test_images.reshape(-1, 28*28)

#labels
C = len(np.unique(train_labels))
# input dimensions
d = train_images.shape[1]
# input weights
W = np.random.randn(d, C)*0.0
NN = NeuralNetwork(W)
NN.train(train_images, train_labels, 0.01)

print('{0:.3f}'.format(NN.performance(test_images, test_labels)))
0.873
```

BIBLIOGRAPHY

[1] E . Oh, T. Keun Yoo, and E. Park.
Diabetic retinopathy risk prediction for fundus examination using sparse learning: a cross-sectional study.
BMC Med Inform Decis Mak, 106(13):106, September 2013.

[2] S. Ibrahim, P. Chowriappa, S. DuaU, R. Acharya, K. Noronha, S. Bhandary, and H. Mugasa.
Classification of diabetes maculopathy images using data-adaptive neuro-fuzzy inference classifier.
PLoS ONE, 4(12):1345–1360, December 2015.

[3] S. Roychowdhury, D. Koozekanani, and K. Parhi.
Dream: diabetic retinopathy analysis using machine learning.
IEEE J Biomed Health Inform, 18(5):1717–1728, Sep 2014.

[4] Z. Torok, T. Peto, E. Csosz, E. Tukacs, A. M. Molnar, A. Berta, J. Tozser, A. Hajdu, V. Nagy, B. Domokos, and A. Csutak.
Combined methods for diabetic retinopathy screening using retina photographs and tear fluid proteomics biomarkers.
Journal of diabetes research, 2015.

[5] E. Kotsiliti, B. Al-Diri, and A. Hunter.
A classification model for predicting diabetic retinopathy based on patient characteristics and biochemical measures.
Journal for Modeling in Ophthalmology, 4(1), 2018.

[6] V. R. Kallem, A. Pandita, and G. Gupta.
Hypoglycemia: When to treat?
Clinical Medicine Insights: Pediatrics, 11:1717–1728, Dec 2017.

[7] B. Sudharsan, M. Peeples, and M. Shomali.
Hypoglycemia prediction using machine learning models for patients with type 2 diabetes.
Journal of Diabetes Science and Technology (JDST), 9(1):86–90, 2014.

[8] G. Thomas.

Ensemble methods in machine learning.
Proceedings of the First International Workshop on Multiple Classifier Systems, 13:1–15, May 2000.

[9] G. Huang, K. Huang, T. Lee, and J. Tzu-Ya Weng.
An interpretable rule-based diagnostic classification of diabetic nephropathy among type 2 diabetes patients.
BMC Bioinformatics, 16(1):86–90, 2015.

[10] S. DuBrava, J. Mardekian, A. Sadosky, E. Jay Bienen, B. Parsons, M. Hopps, and J. Markman.
Using random forest models to identify correlates of a diabetic peripheral neuropathy diagnosis from electronic health record data.
Pain Medicine, 18(1):107–115, 2017.

[11] D. de Savigny, I. Riley, D. Chandramohan, F. Odhiambo, E. Nichols, and S. Notzo.
Integrating community-based verbal autopsy into civil registration and vital statistics (crvs): system-level considerations.
Global Health Action, pages 272–278, 2017.

[12] J. Todd, R. Balira, H. Grosskurth, P. Mayaud, F. Mosha, G. ka Gina, A. Klokke, R. Gabone, A. Gavyole, D. Mabey, and R. Hayes.
Hiv-associated adult mortality in a rural tanzania population.
AIDS, 11:801–807, 1997.

[13] C. Coldham et al.
Prospective validation of a standardized questionnaire for estimating childhood mortality and morbidity due to pneumonia and diarrhoea.
Tropical Medicine and International Health, 5:134–144, 2000.

[14] MA. Quigley et al.
Prospective validation of a standardized questionnaire for estimating childhood mortality and morbidity due to pneumonia and diarrhoea.
International Journal of Epidemiology, 28:1081–1087, December 1999.

[15] S. Makridakis et al.
Statistical and machine learning forecasting methods: Concerns and ways forward.
PLoS ONE, 13:1081–1087, March 2018.

[16] E. Tu, S. M. Twigg, J. Duflou, and C. Semsarian.
Causes of death in young australians with type 1 diabetes; a review of coronial postmortem examinations.
Medical Journal, 188(12):699–702, 2008.

[17] Agincourt.
http://wwww.agincourt.co.za/.
Accessed: 2018-02-25.

[18] ibm.
https://www.ibm.com/annualreport/2013/bin/assets/2013_ibm_annual.pdf.
Accessed: 2018-01-05.

[19] machine.
https://en.wikipedia.org/wiki/Machine_learning.
Accessed: 2018-01-12.

[20] L. Wu, F. Tian, Y. Xia, Y. Fan, T. Qin, J. Lai, and T. Liu.
Learning to teach with dynamic loss functions.
Machine Learning, 9:1–10, 2008.

[21] A. Statnikov, L. Wang, and C. F. Aliferis.
A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification.
BMC Bioinformatics, 9:1–10, 2008.

[22] M. Li, Z. Tong, C. Yuqiang, and J. S. Alexander.
Efcient mini-batch training for stochastic optimization.
Knowledge Discovery and Data Mining 2014, 2:661–670, 2014.

[23] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan.
Better mini-batch algorithms via accelerated gradient methods.
Neural Information Processing Systems, 24:1647–1655, 2011.

[24] J. Duchi, E. Hazan, and Y. Singer.
Adaptive subgradient methods for online learning and stochastic optimization.
The Journal of Machine Learning Research, 12:2121–2159, 2011.

[25] M. D. Zeiler.
Adadelta: An adaptive learning rate method.
arXiv: Learning, 2:2121–2159, 2012.

[26] D. P. Kingma and J. Lei Ba.
Adam: A method for stochastic optimization.
International Conference on Learning Representations, 2, 2015.

[27] W.S. McCulloch and W. Pitts.
A logical calculus of the ideas immanent in nervous activity.
Bulletin of Mathematical Biology, 52:99–115, 1990.

[28] A. Averkin.
Time series forecasting based on hybrid neural networks and multiple regression.
Proceedings of the First International Scientific Conference "Intelligent Information Technologies for Industry (IITI'16). Advances in Intelligent Systems and Computing, vol 450. Springer, Cham, pages 76–82, 2016.
doi:10.15827/0236-235X.113.075-082.

[29] V. Nair and G. E. Hinton.
Rectied linear units improve restricted boltzmann machines.
Proceedings of the 27th International Conference on International Conference on Machine Learning, pages 807–814, 2010.

[30] A. Bordes, X. Glorot, and Y. Bengio.
Deep sparse rectifier neural networks.
Journal of Machine Learning Research, 2011.

[31] C. Djork-Arné, U. Thomas, and H. Sepp.
Fast and accurate deep networks learning by exponential linear units (elus).
Proceedings of the 27th International Conference on International Conference on Machine Learning, 1, 2015.

[32] B. Efron.
Bootstrap methods: Another look at the jackknife.
Institute of Mathematical Statistics, 7:1–26, 1979.

[33] G. James, D. Witten, T. Hastie, and R. Tibshirani.
An introduction to statistical learning: with applications in r.
Springer Texts in Statistics 103, 2013.
doi:10.1007/978-1-4614-7138-7_3.

[34] A. Natekin and A. Knoll.
Gradient boosting machines, a tutorial.
Front Neurobot, 7(21), 2013.

[35] Kaggle.
`https://www.kaggle.com/datasets`.
Accessed: 2018-02-12.

[36] Mnist database.
`https://en.wikipedia.org/wiki/MNIST_database`.
Accessed: 2015-01-08.

[37] Numpy.
http://www.numpy.org/.
Accessed: 2015-01-08.

[38] J. Peralta, G. Gutierrez, and A. Sanchis.
Artificial neural networks – icann 2010.
In *ICANN'10 Proceedings of the 20th international conference on Artificial neural networks: Part I*, volume 352, pages 50–53, 2010.
ISBN 3-642-15818-8 978-3-642-15818-6.

[39] hypers.
https://pypi.org/project/hyperas/.
Accessed: 2015-01-08.

[40] scikit-learn.
https://scikit-learn.org/stable/.
Accessed: 2015-01-08.

[41] A. Ramachandran.
Know the signs and symptoms of diabetes.
Indian Journal of Medical Research, 140:579–581, 2014.

[42] L. Bertolaccini, L. Boschetto, A. Pasini, A. Viti, A. Attanasio, A. Terzi, and C. Cassardo.
Analysis of spontaneous pneumothorax in the city of cuneo: environmental correlations with meteorological and air pollutant variables.
Surgery Today, 45:625–629, 2015.

[43] A. Pasini.
Artificial neural networks for small datasets.
Journal of Thoracic Disease, 7(5):953–960, 2015.

[44] C. Chen, A. Liaw, and L. Brieman.
Using random forest to learn imbalanced data.
University of California, Berkeley, 2004.

[45] P. Domingos.
Metacost: a general method for making classifiers cost-sensitive.
KDD '99 Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 155–164, 1999.

[46] L. Brieman.
Random forest.
Machine Learning, 45(1):5–32, 2001.

[47] F. Provost and T. Fawcett.
Robust classification for imprecise environments.
Machine Learning, 2000.

[48] A. Kumar.
Comparison of neural networks and regression analysis.
Expert Systems with Applications, 29(2):424–430, 2005.

[49] H. Altun and K. M. Curtis.
Acoustic-to-articulatory neural mapping under different statistical characteristics of articulatory pattern vector.
International conference on phonetic science, 7:2017–2020, 1999.

[50] H. Altun and K. M. Curtis.
The accurate estimation of articulatory synthesiser parameters through reducing the degree of saturation in a neural network hidden layer.
International conference on phonetic science,ICPhS'99, 7:2263–2267, 1999.

[51] H. Altun, K. M. Curtis, and T. Yalcinoz.
Neural learning for articulatory speech synthesis under different statistical characteristics of acoustic input patterns.
Computers and Electrical Engineering, 29(6):687–702, 2003.