A MACHINE LEARNING CLASSIFICATION ALGORITHM TO DIAGNOSE EPILEPSY BASED ON THE ANALYSIS AND CLASSIFICATION OF VARIOUS COMBINATIONS OF FEATURES EXTRACTED FROM EEG SIGNALS.

PART B

RUPESH KUMAR DEY

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. ABSTRACT

Machine Learning (ML) in the industry has been rapidly developing over the years particularly in the field of healthcare. An area in healthcare where machine learning is currently growing is in disease diagnosis. Epilepsy is a neurological disease which is suffered by almost 50 million people all over the world with 80% of them coming from developing nations like Malaysia. [1]. Traditional methods to diagnose this disease is via manual analysis of EEG readings. However, this study seeks to improve the conventional method by introducing machine learning to automate the process. Part A of this study details the introduction to Epilepsy, EEG data and machine learning. It defines the problem and aim of this study. Furthermore, a detailed literature review was conducted on related works to Epilepsy, EEG and machine learning classification algorithms used in diagnosis of various diseases and Epilepsy itself. Based on analysis of literatures, 3 machine learning models were selected to be compared with one another which are Support Vector Machine (SVM), Artificial Neural network (ANN) and K-Nearest Neighbours Classifier (KNN).

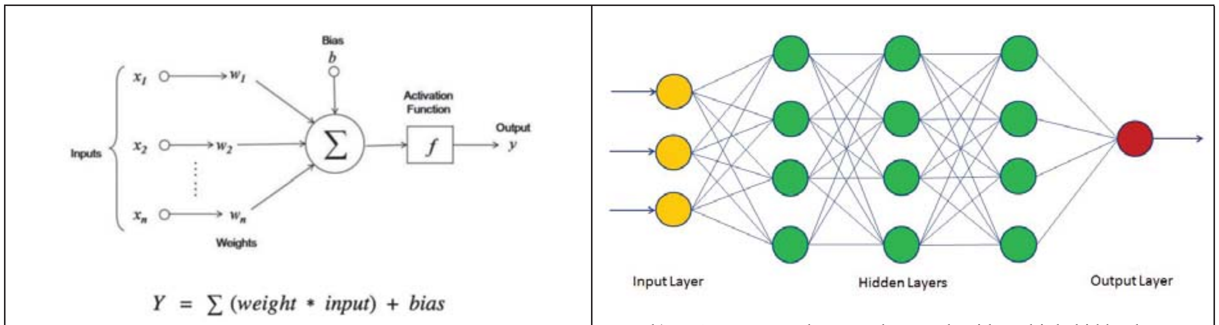
In continuation, the following sections are detailed in this Part B document. Introduction, data sourcing and preparation, model implementation, model validation, model tuning, individual model type evaluation and analysis to select the best model in its own class for comparison, comparing between the best model for SVM, ANN and KNN to select the best model for the problem followed by the study's conclusion and reflection.

2. INTRODUCTION

The aim of this study as stated in part A to develop an automated machine learning classification algorithm for epilepsy based on the analysis of EEG data. In part A, a detailed literature review on related works to the concepts of Epilepsy, analysing EEG, various machine learning classification models and the implementation of several machine learning classification models for automated diagnosis of epilepsy by analysing EEG data was conducted to properly grasp the fundamental concepts of the problem and the current related works to properly propose an effective solution. After detailed study, 3 machine learning algorithms were selected and compared to one another which are Support Vector Machine (SVM), Artificial Neural Network (ANN) and KNN classifier (KNN).

SVM is a commonly used machine learning classification algorithm. Is it advantageous in the sense that it is capable of handling both categorical and numerical data for classification problems. The model works by creating a hyperplane as the decision surface that segregates the positives and negative outcome class values from each other with high margin (soft margin). [15] A soft margin is an invisible border on both sides of the hyperplane with a certain distance that helps in reducing misclassification error by preventing data points close to the hyperplane from being misclassified to the other class. SVM makes use of data points near the decision surface only if the data point is useful information for classification. For more complex data configurations, the SVM transforms the data into a high dimensional space for ease of classification. SVM was selected due to its high accuracy in classification results based on literature review. It has a lesser risk in overfitting, its kernel functions that has the capability of solving complex problems and has the capability of not falling into the local optimum.

ANN classifiers classify different classes via a group of interconnected artificial nodes simulating neurons in the human brain. A basic neural network consists of 3 layers which are input layer, radial basis layer, and the output layer. The input layer is basically the input features of datasets that are fed into the ANN. The hidden layers can consist of 1 to any positive integer depending on the complexity of the problem and each layer has 3 components of synaptic weights, summing function and the activation function. The model calculates the weights for each input with respect to each node in the hidden layer and the weights between each node in the hidden layer to consecutive hidden layers node until the final output. During training, weights are adjusted iteratively to model the training data accurately. A basic architecture of a neural network is shown in Figure 1 below. Neural network was chosen due to its advantage of adaptive learning, real time operation fault tolerance and self-organization giving the capability of the model of in-built tuning of the model for improved accuracy.



a) A basic neural network with only a single hidden layer.

b) A more complex neural network with multiple hidden layers

Figure 1: Basic Architecture of an Artificial Neural Network [14]

K-Nearest-Neighbours-Classifier is an algorithm that stores all available instances of data and their classes. It classifies new data based on the similarity of the new data features with respect to the n number of in-stored information by the model within a certain distance. A new data point is classified based on a majority vote of its similarities with its neighbours. The new data point is assigned to the class most common amongst its K nearest neighbours measured by a distance function. The reason why KNN was selected was due to its very simple implementation, there is no requirement to train the model and it is capable to constantly evolve with new training data with labels. [12] Furthermore, it also produces relatively high accurate results based on literature review.

The machine learning classification algorithm was implemented using R-programming, a well-known computer program language used in the field of machine learning due to its capability in handling heavy amounts of data. It also allows the capability for diving deep into the data and allows very good data visualization. R also provides rapid prototyping and working with datasets for development of the machine learning algorithms. R programming is an open-source program and is free. It has exemplary support for data wrangling (transform messy data into structured format). It also has a high quantity of pre-programmed packages which are readily available, useful, and versatile for all sorts of application. It is highly compatible and can be paired with many other programming languages. It also provides very detailed and effective reports for the user. [13] For these mentioned reasons, R programming was chosen.

3. DATA PREPARATION

3.1 DATASET SOURCING

The dataset was sourced from the UCI Machine Learning Repository under the data title "Epileptic Seizure Recognition Data Set". The dataset is a multivariate time-series dataset of 11500 rows and 179 columns (178 of which are voltage readings over a period of time and the 179th column is the Y-value class). The original dataset used for this study is from the Bonn dataset obtained from the Department of Epileptology, University of Bonn [2]. The dataset contained 3 classes of data which are normal, epileptic background without seizure (pre-ictal) and epileptic with seizures (ictal). Data acquisition was done through a single channel EEG signals for a duration of 23.6 seconds consisting of 5 healthy normal patients for normal data and 5 Epileptic patients for pre-ictal and ictal data. The data is segregated into 5 sets i.e. A, B, C, D and E each with 100 single channels EEG segments. Each segment consists of 4096 sampling points over 23.6 seconds. Artifacts were manually removed from the dataset. Sets A & B are extracranial EEG from healthy volunteers with open and closed eyes, respectively. Sets C and D were obtained from epileptic patients in the pre-ictal phase (i.e., epileptic with no seizures) whereas Set E was obtained from epileptic patients during seizures. Probes for data collection were placed according to the standardized 10-20 electrode placement technique. The data obtained was recorded by a 128-channel amplified system, sampled at a rate of 173.61 Hz, and discretised with 12-bit A/D resolution.

Qiuyi Wu and Ernest Fokoue from the Rochester Institute of Technology divided and shuffled every 4097 data points of the original dataset into 23 segments, each with 178 data points. Each instance represented 1 second value worth of data representing EEG recording at a different point in time. There are a total of 23 x 500 = 11500 instances (row) of data each with 178 data. The last column represents the classification group of the row which are divided into 5 different groups represented in numerical order from 1 to 5 explained as below [17].

1 – Epileptic with seizure. (Ictal)
2 – Epileptic without seizure (readings measure from the affected part of the brain, pre-ictal)
3 – Epileptic without seizure (readings measured from healthy part of the brain, pre-ictal)
4 – Normal, eyes closed.

5 – Normal, eyes open.

The 5 groups are grouped into binary classes of epileptic and non-epileptic. Patients in classes 1,2, and 3 were categorized as epileptic whereas patients from groups 4 and 5 are non-epileptic. The raw dataset is a time series and does not have useful features that can be directly used for classification purposes. It only contained voltage readings of the brain for over a period of time. Hence, nonlinear features had to be extracted from the data using statistical methods in order to obtain useful features to process and feed into the machine learning algorithms. This will be further discussed in the dataset preparation section.

The general framework of the study is explained as such. The raw dataset was pre-processed, cleaned, and processed to extract its non-linear features and saved in the format of ".csv". For each model implementation, the data was imported, relevant features were extracted, and the Y-values were converted to factors. The data was then split into training and test sets, and then trained and tested with the basic machine learning model. Its performance parameters were then evaluated. Subsequently for each model, hyperparameter tuning was carried out in order to improve its accuracy and a few additional models with tuned parameters were trained and tested. The performances of each of the model was critically analysed and evaluated in order to select the best performing model as a solution for the problem. It is important to note that the method in which the programming was done for the R-files of this study is that data preparation, individual model implementation and tuning, and model comparisons between the 3 types for classifiers were done in different R-script files in order to make code readability clearer and for easier execution of each model as the structure of inputs required may differ from one another. A flowchart of the general framework for this study is shown in Figure 2.

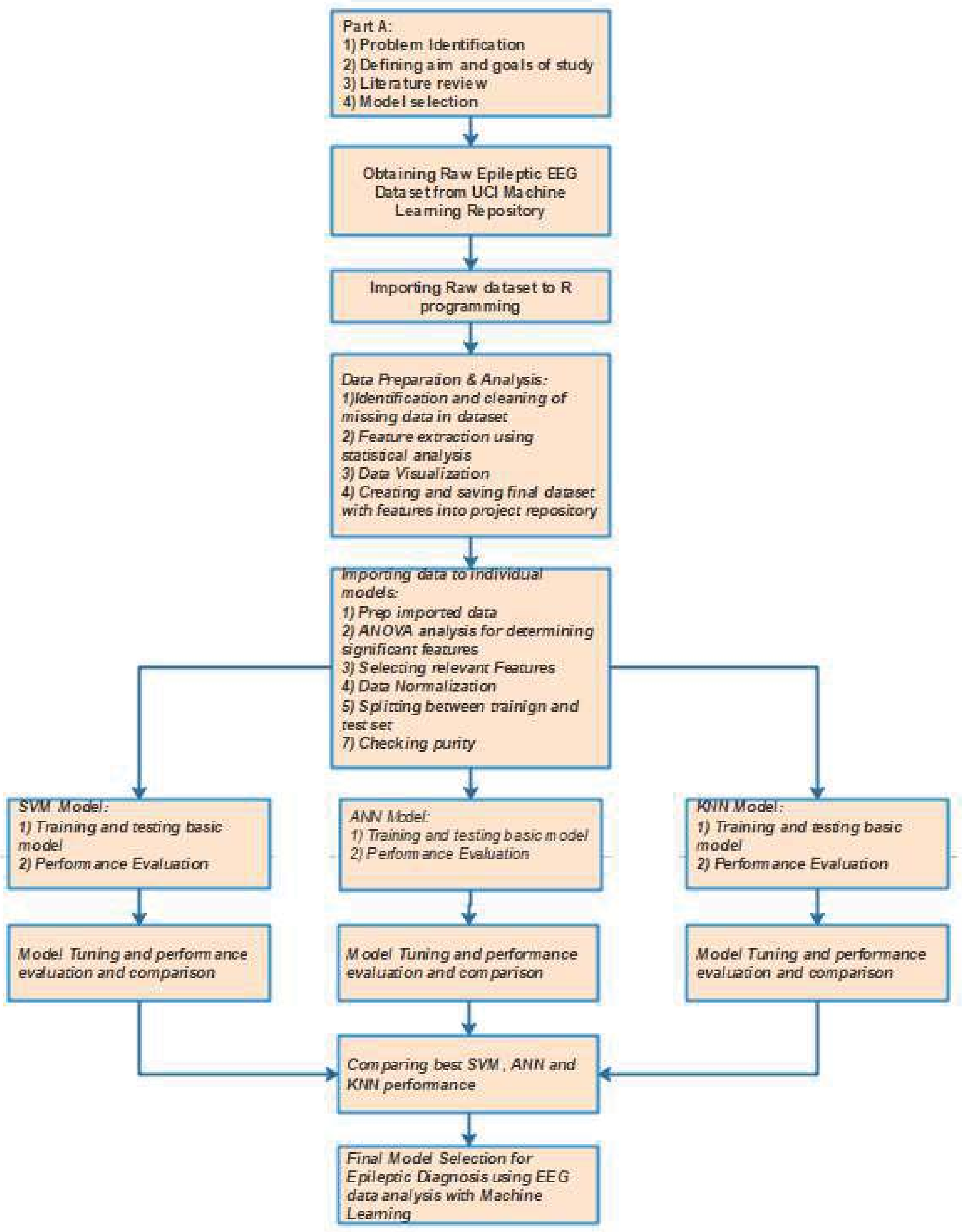

Part A:
1) Problem Identification
2) Defining aim and goals of study
3) Literature review
4) Model selection
Obtaining Raw Epileptic EEG Dataset from UCI Machine Learning Repository
Importing Raw dataset to R programming
Data Preparation & Analysis:
1)Identification and cleaning of missing data in dataset
2) Feature extraction using statistical analysis
3) Data Visualization
4) Creating and saving final dataset with features into project repository
Importing data to individual models:
1) Prep imported data
2) ANOVA analysis for determining significant features
3) Selecting relevant Features
4) Data Normalization
5) Splitting between trainign and test set
7) Checking purity
SVM Model:
1) Training and testing basic model
2) Performance Evaluation
ANN Model:
1) Training and testing basic model
2) Performance Evaluation
KNN Model:
1) Training and testing basic model
2) Performance Evaluation
Model Tuning and performance evaluation and comparison
Model Tuning and performance evaluation and comparison
Model Tuning and performance evaluation and comparison
Comparing best SVM, ANN and KNN performance
Final Model Selection for Epileptic Diagnosis using EEG data analysis with Machine Learning

Figure 2: General Framework of Study

There are several performance measurement parameters that can be considered which are accuracy, sensitivity, and specificity etc. When diagnosing a patient with a disease, there are several situations in which the test can result in which is True Positive (TP) (patient is sick and is diagnosed sick), True Negative (TN) (patient is not sick and is diagnosed not sick), False Positive (FP) (patient is not sick but diagnosed sick), and False Negative (FN) (patient is sick but diagnosed not sick). In reality, the model built is not perfect and there are some margins for errors in the prediction. However, the machine learning model trained for epileptic diagnosis should have a minimal error margin. Sensitivity is the measure of how good the model is in correctly identifying the epileptic patients. Specificity on the other hand is a measure of how good the model is in correctly identifying non-epileptic patients. Accuracy on the other hand measure how good the model is in predicting the overall results either true or false. It measures the veracity of the model. In the medical field of disease diagnosis, it is imperative that the disease detection is not only accurately diagnosed, but also ensures the number of possible positive conditions that is correctly diagnosed is as much as possible [16]. Hence, two performance parameters were considered in evaluating the models which are accuracy and sensitivity. However, greater weight was put on the accuracy of the model whereas the sensitivity of the model was checked to ensure that it was high enough to complement the accuracy. The equations to calculate accuracy and sensitivity are shown as below

$$Accuracy = \frac{(\mathrm{TN} + \mathrm{TP})}{(\mathrm{TN} + \mathrm{TP} + \mathrm{FN} + \mathrm{FP})}$$

$$Sensitivity = \frac{\mathrm{TP}}{(\mathrm{TP} + \mathrm{FN})}$$

3.2 ANALYSING THE RAW DATA, CLEANING DATA AND EXTRACTING FEATURES

The raw data was presented in the form of a numerical continuous time series with the Y-value label as a categorical data. As the raw data was in the form of a time series, there were no distinguishable features from the 178 columns which can directly be used for classification between the 11500 data instances. Hence, statistical analysis first had to be carried out to extract the features from the raw dataset for ease of analysis. Based on literature review, 9 non-linear statistical features were extracted from the raw non-linear time series data which are

Approximate Entropy (ApEn), Sample Entropy (SampEn), Hurst Exponent (HE), Kurtosis, Skewness, Mean, Variance, Spectral Entropy and Detrended Fluctuation Analysis (DFA). The significance of each data is detailed in Table 1 below.

Feature	Significance
Approximate Entropy	A measurement of the likelihood that data patterns within a time series continues to remain the same pattern. In other words, it measures the regularity of the data analysed. Regular and consistent data have smaller ApEn values whereas irregular data have higher ApEn values. [3][4][5]
Sample Entropy	An estimate of the probability that patterns within a time frame are similar within a certain tolerance range defined as "r". It is also a measurement of data regularity like ApEn. A clear distinction of this algorithm compared to ApEn is that it is independent of the entire data record length and displays relative consistencies as compared to ApEn [3]. The scale values index obtained from SampEn is similar to that of ApEn whereby larger values corresponds to more regular data and vice versa.
Hurst Exponent	Characterizes the degree of continuum of the pattern in the time series data. An exponent of $H = 0.5$ is a random walk, an exponent of $0 < H < 0.5$ indicates an anti-persistent behaviour and $0.5 < H < 1$ indicates persistence in the data pattern. [4]
Kurtosis	Kurtosis is a statistical parameter that measures the complexity of an EEG data. In addition, it also determines if the EEG signal has a peak or rather flat at the mean point of the signal. [7][8]
Skewness	Skewness is another higher order statistical measure of the lack of symmetry or the asymmetry of an EEG signal data set. [7][8]
Mean	The mean voltage value for the entire time series instance.
Variance	A measurement of how far the data is spread from its average value.
Spectral Entropy	Spectral entropy (SE) is a nonlinear method to summarize signal power irregularity over measured frequencies. It measures the

	signal irregularity. Since most physiological signals are nonlinear, entropy as a nonlinear method is ideal to study neural signals. [8]
Detrended Fluctuation Analysis	A long-range time-based relationship quantifying technique in EEG signal. Has the capability to handle non-stationary data and also removes noise in the EEG data. The DFA is denoted by α. When α = 0.5 the signal is a random walk. A range of $0 < \alpha < 0.5$ denotes there is power-law anti-correlations and a range of $0.5 < \alpha < 1$ indicates that a long-range power law correlation exists. [4]

Table 1: Statistical Features and significance.

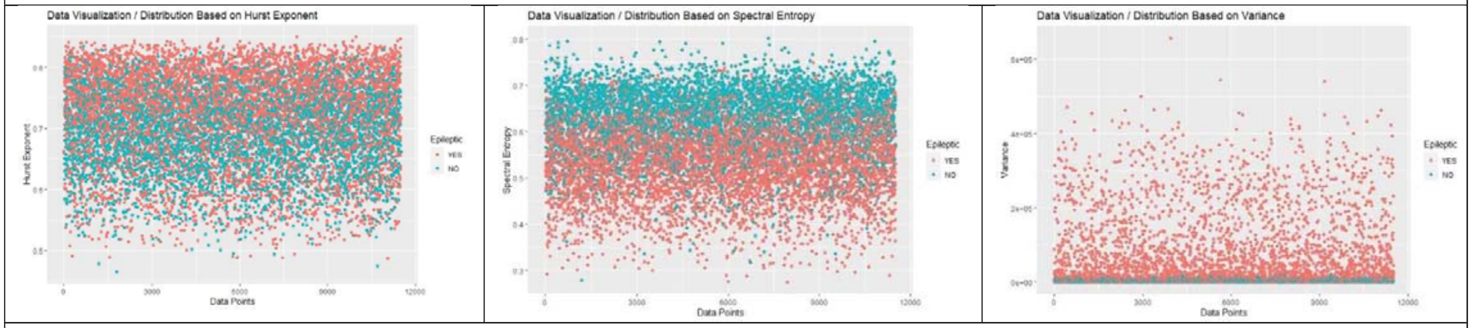
The first step involved importing the .csv file obtained from UCI repository from the project repository. A condition was set in the read.table() function for empty cells in the data to be set as "NA". Then, the imported data was checked with the sum(is.na()) function to check if there are cells in the data that has missing values. Based on analysis 10 total missing data points were identified. To replace those missing data points, their location was first determined using the which(is.na(filter), arr.ind=TRUE) function. Based on the row and column obtained for the 10 missing data points, the average value for that particular row 10 rows were calculated, and the missing data points were replaced with their respective rows averaged values.

To extract features using statistical analysis, new vector variables for each feature was created and specific functions were used on the entire dataset to extract them. To obtain ApEn and SampEn, the "**TSEntropies**" package was installed. Using parameters obtained from paper [5] for dim, lag and r as 2,1 and 0.2 respectively, the ApEn() and SampEn() functions were implemented. Next, to obtain the Hurst Exponent (HE), the "**pracma**" package was used with function hurstexp(). The hurstexp() function returns 5 different types of Hurst Exponnets. However, for this study, only the simplified HE using R/S approach was extracted from the returned list of results [4][9]. For Skewness and kurtosis, the package "**moment**" was called and used both functions kurtosis() and skewness() functions. Spectral entropy was calculated using the "**ForeCA**" package using spectral_entropy() function. Features of mean and variance were extracted from the data using the var() and mean() functions. Last but not least, in order to obtain DFA, the "**nonlinearTseries**" package was installed and the estimate(dfa()) function was used.

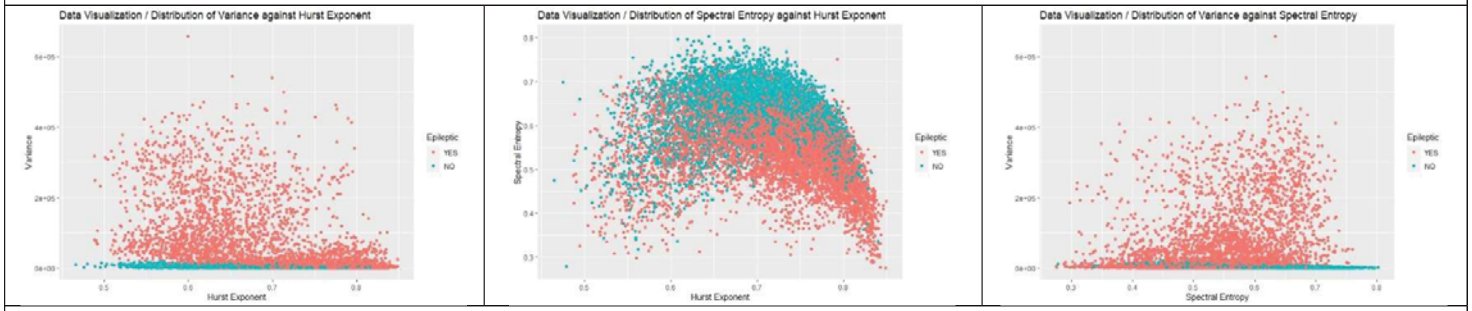
Once the features had been extracted, the vectors of data features were then cleaned to remove the repeated rows for the first instance of data. A data frame was then created consisting of these 9 features together with the final classification label. The data is then written to a .csv file and then saved in the project directory as a separate file to be imported by the specific model programs later on. During data preparation, an additional step was also taken to plot and visualize the data to understand how the data is distributed in relation to one another. There is a total of 9 features in the dataset and visualizing all of them at once is not possible. The maximum level in which the data could be visualized was up to 3-dimensional space. Hence, 3 features were selected based on their statistical significance, based on ANOVA analysis. These 3 features were variance, spectral entropy and hurst exponent. These 3 data were scatter plotted in 1-Dimensional form individually, 2-Dimensional form with all possible combinations of the selected features and finally a 3-Dimensional scatter plot with all 3 features together. The plots are displayed in Figure 3.

It can be observed from the 1-Dimensional plot that the data points for each class are very mixed up with each other non-linearly. This makes visualizing and separating the data difficult. However, in the 2-Dimensional plot, it can be seen that the data segregation improves between the 2 classes and there is clearer visualization of the data. In the 3-Dimensional plot, a clearer image is displayed on how the datapoints for Epileptic and Non-Epileptic patients are separated from one another as compared to the lower dimensional plots. From this it can be deduced that with higher number of dimensions of features and data points, data segregation between the 2 classes are displayed with better clarity and the machine learning models are capable of better visualizing and understanding the data while also being able to accurately distinguish between the two classes.

1 Dimensional Plot



2 Dimensional Plot



3 Dimensional Plot

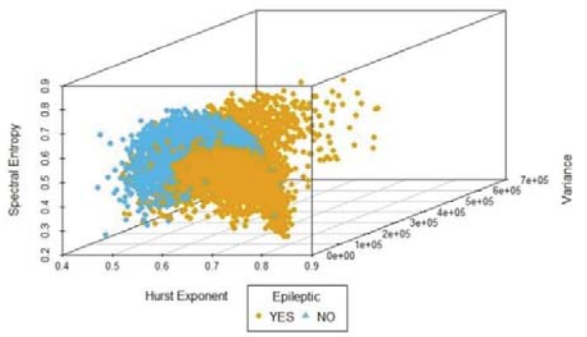
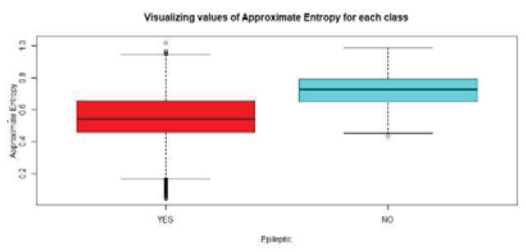
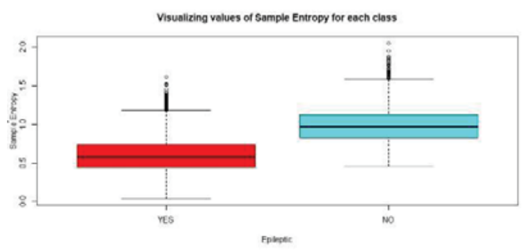


Figure 3: 1D to 3D visualization plots of the Data.

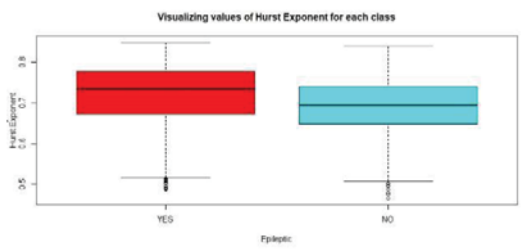
Additionally, the values of each feature within the dataset for each class was also plotted and analysed in Figure 4. From the plots it can be seen that generally, data instances that are non-epileptic have higher Approximate, Sample Entropy and Spectral Entropy values. This agrees well with findings presented in literature review discussed in Part A whereby normal EEG dataset have higher values of entropies during normal state as the EEG readings are highly irregular as compared to when the patient is epileptic (in the ictal and pre-ictal phase). [3]. It is observed that values for Detrended Fluctuations and Hurst Exponent are generally higher for Epileptic instances which agrees with the results from [5]. This is due to the fact that EEG readings for Epileptic states (pre-ictal and ictal) have more regularized and persistent data as compared to the normal state.



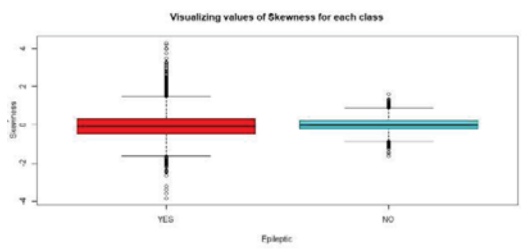
a) Visualizing values for Approximate Entropy



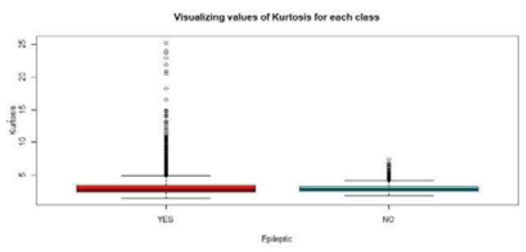
b) Visualizing values for Sample Entropy



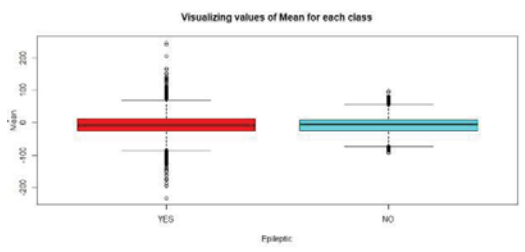
c) Visualizing values for Hurst Exponent



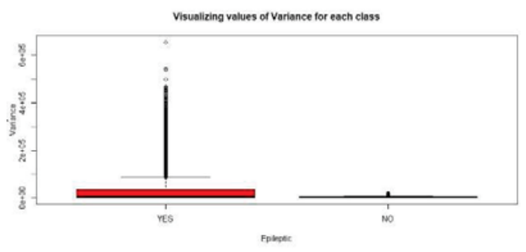
d) Visualizing values for Skewness



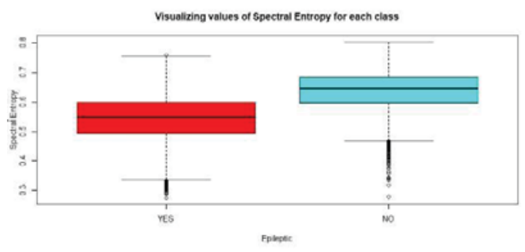
e) Visualizing values for Kurtosis



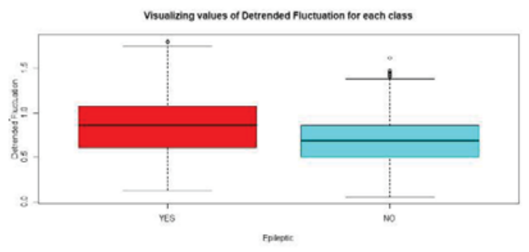
f) Visualizing values for Mean



f) Visualizing values for Variance



g) Visualizing values for Spectral Entropy



h) Visualizing values for Detrended Fluctuation

Figure 4: Feature data plot to visualize the values for each feature

3.3 IMPORTING THE PREPPED DATA WITH FEATURES, SELECTING FEATURES, DATA NORMALIZATION AND DATA SPLITTING

For each type of model implementation, the data was first imported from the project folder directory. As the imported data parameters were read as char when being imported, the as.numeric() function was implemented for each of the columns in the data converting them from character format to numerical format. In order to determine which features from the processed dataset were significant enough to contribute in the models' classification, Analysis of Variance (ANOVA) was carried out using the aov() function on the dataset. If the aov value of $p<0.05$, the data for that particular feature was considered to be significant, else the data would be removed. Low p-values are indications of strong evidence against the null hypothesis. Based on analysis, the Kurtosis feature had a p value of 0.202 which is significantly larger than 0.05. Hence, this feature was removed from analysis in further steps.

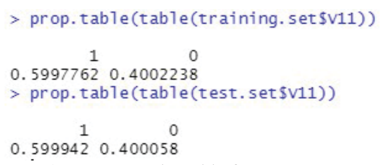
An additional step was carried out prior to data splitting which is data normalization. A separate data frame excluding the Y-values was created for the purpose of normalization called data.norm. The new data frame was created to ensure that the original data is not tampered with. A function is then created using the equation above which is then fed into the lapply() function with the data.norm variable having feature columns 1 to 8 to normalize. The normalized data is then created and saved in the data.norm variable. The Y-value is the appended to the normalized data frame and the data is split into training and testing set.

Data normalization is defined as the step to organize the data for it to appear similar across all records and fields. It increases the cohesion of entry types leading to cleansing, lead generation, segmentation, and higher quality data. It is the step to bring all the values in the data to a normalized value as numerical feature values may vary in range and have difference in significance values. This can significantly affect the classification accuracy. The data is then normalized using the function as below:

$$f(x) = (x - \min(x))/(\max(x) - \min(x))$$

Then, the y-values were then labelled as factors of 1 and 0. 1 indicating that the patient is epileptic whereas 0 labels the patient as non-epileptic. Groups 1,2 and 3 were labelled as 1 whereas groups 4 and 5 were labelled as non-epileptic. [2]. The next step would involve splitting the data between training and test set. A split variable was created to select random rows as TRUE for training set and FALSE for test set. The data was then split into 2 different variables of training.set and test.set with their respective randomly selected rows of data. The split ratio was set at 70% training data and 30% test data.

The purity of the data split for each class within the training and testing set was checked to ensure that the split of each class is adequate to train the model. If the split is reasonably good with more than 50% constituent of the positive class in the training and test set, the purity was acceptable. The purity of data split for all 3 models is shown in Figure XX below. It is observed that the data distribution of Y-Values for all 3 models in both the training and test dataset is approximately 60% : 40%, Epileptic : Non-Epileptic split which is adequate for model implementation. There is an almost equal distribution of Y-values in both training and test sets with a slightly higher quantity of data in the positive class.

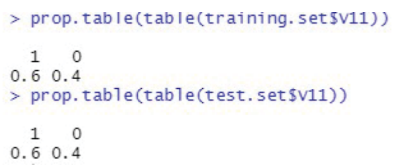


```
> prop.table(table(training.set$V11))

        1         0 
0.5997762 0.4002238 
> prop.table(table(test.set$V11))

       1        0 
0.599942 0.400058 
```

a) Data Purity table for SVM

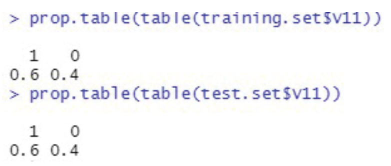


```
> prop.table(table(training.set$V11))

  1   0 
0.6 0.4 
> prop.table(table(test.set$V11))

  1   0 
0.6 0.4 
```

b) Data Purity table for ANN



```
> prop.table(table(training.set$V11))

  1   0 
0.6 0.4 
> prop.table(table(test.set$V11))

  1   0 
0.6 0.4 
```

c) Data Purity table for KNN

Figure 5: Purity check for each model

4. MODEL IMPLEMENTATION, VALIDATION, TUNING & EVALUATION

4.1 BUILDING INDIVIDUAL MACHINE LEARNING MODELS

4.1.1 SVM

The first step in building the SVM model is to import all of the necessary library packages. For SVM, the following packages were installed.

No	R-Package	Function
1	ggplot2	Used for graphical representation of data / model
2	e1071	Used for implementing the SVM model.
3	caTools	Used for data splitting
4	caret	Additional library installed for the use of the confusionMatrix() function

Then the data was imported, cleaned, and split between test and training set to be implemented with the model as detailed in Section 3.3. In order to implement the SVM model, the svm() function from the package e1071 was used. The generalised input functions into the model are shown below:

svm("DV"~ "IV", data = "TS", kernel = "K", epsilon = "E", cost="Gamma")

Parameter	Parameter Function / Explanation
DV & IV	Dependent Variable ~ Independent Variables
data = "TS"	Data frame to be used. In this case it will be the training set.
Kernel = "K"	Different Kernels with different functions to train the model and define the hyperplane. Three different kernels were used which are radial basis function (RBF), linear, and sigmoid. The best kernel will be chosen for hyperparameter tuning.
epsilon = "E"	Epsilon value. This was only implemented when the best kernel was determined, and further tuning process was done.
cost=" Gamma"	Cost value. This was only implemented when the best kernel was determined, and further tuning process was done.

3 different types of kernels can be used when creating the SVM model which are radial basis function, linear and sigmoid. The kernels are different mathematical methods used in defining the hyperplane decision boundary between the classes. The kernel functions are used to map the original dataset (linear/nonlinear) into a higher dimensional space with view to making it linear dataset. In order to determine which kernel was best suited

3 different SVM models were first trained using the training set using the 3 different kernels. The accuracy of the 3 different kernel-based models were compared to one another. This step was carried out in order to determine the most accurate kernel-based model among the 3 to further tune the model in order to obtain the best model. Once the 3 models were trained, they were used to predict the class of the test set data instances. During this step, the test set Y-values were masked from the model. Once prediction is completed, the confusion matrix of the models was computed comparing the predicted values by the models to the actual Y-values from the test set. The confusion matrix will tell us the amount of correctly and wrongly classified instances. The layout of the confusion matrix is displayed as below in Figure 6 Based on the confusion matrix, the accuracy of each model was evaluated using the formula below:

$$accuracy\ (\%) = \frac{sum\big(diagonal(confusion\ matrix)\big)}{total\ number\ of\ istances\ in\ test\ set}$$

	ACTUAL	
PREDICTED	Positive	Negative
Positive	True Positive	False Positive
Negative	False Negative	True Negative

Figure 6: Layout of the Confusion Matrix

The accuracies for each of the model were plotted on a bar graph and compared to one another. Results showed that radial basis function obtained the highest accuracy among the three with an accuracy of 91.64%. Linear-kernal based model obtained the 2nd highest accuracy with 89.79% and the sigmoid-kernel-based model only obtained 78.13% accuracy. The results obtained are presented on Figure 7 below.

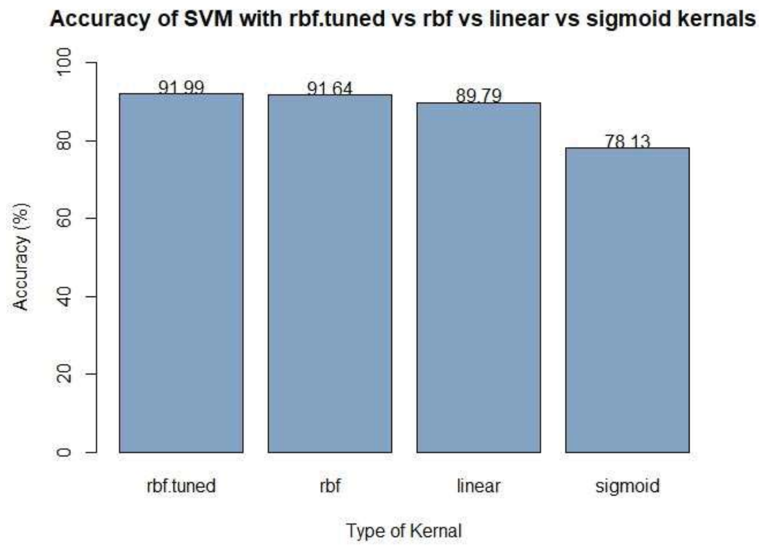


Figure 7: Accuracy of SVM models

Once it was determined that the RBF based SVM was the most accurate among the 3 kernels, model tuning was performed in order to tune the hyperparameters of the model. Hyperparameter tuning is carried out using the tune() function. For SVM models, the tuning parameters are the epsilon (ε) and the cost for the RBF SVM model. The ε and cost determine how the model learns the dataset and defines the hyperplane and the allowable tolerances for errors in the model i.e., determining the hyperplane separation line and the soft margin. For this particular study, the tuning process was carried out with ε value range of 0 to 2 with 0.1 increments whereas cost was set to a range of 2^0 to 2^3. The tune function returns a list with the results of the combinations of ε and cost values with their corresponding error. The best pair of epsilon and gamma values are then selected. ε values and cost were 0 and 4. Figure 8 below visualizes how the SVM model performs with each value of ε and cost values and Figure 9 displays the parameters obtained for the best tuned model.

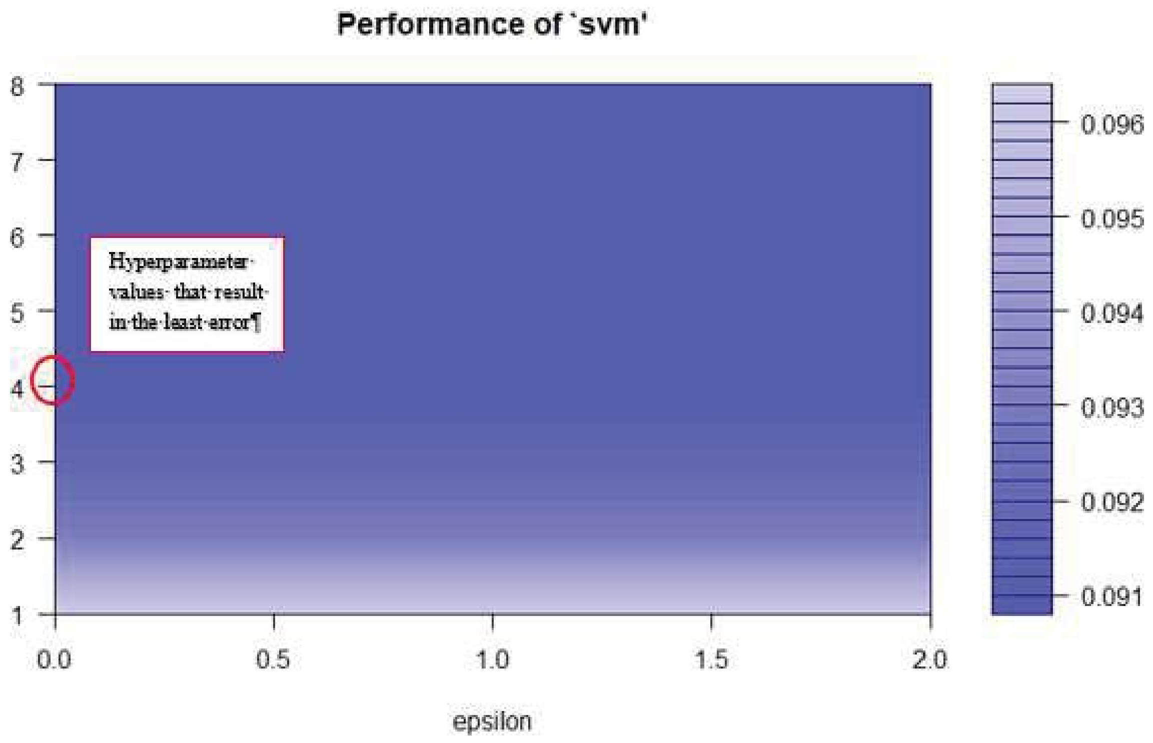
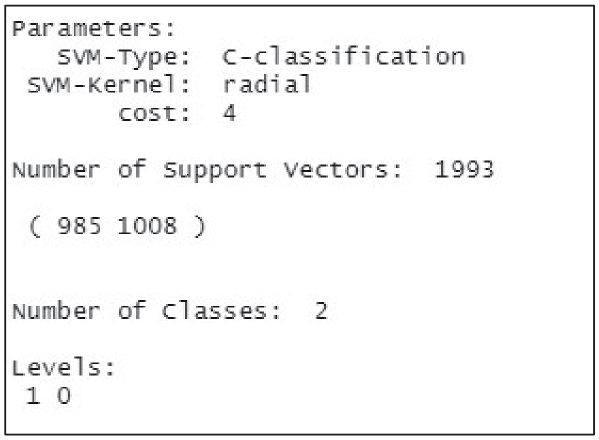


Figure 8: Performance of SVM under various Epsilon and Cost values



```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  4

Number of Support Vectors:  1993

 ( 985 1008 )


Number of Classes:  2

Levels:
 1 0
```

Figure 9: R output for best parameters for SVM model

The tuned model was then built and trained using the best parameters and was then used to predict the test set. The accuracy results obtained was plotted and compared against all of the earlier models. The results showed that the accuracy improved after the model was tuned resulting in an accuracy of 91.99% from 91.64% as shown in Figure 7

In addition, the sensitivity of each SVM model is also presented in Figure 10 below. The sensitivity was obtained using the confusionMatrix() function from caret package. It can be seen that the sensitivity of the RBF based SVM (92.07%) is the highest compared to the other 2 models (82.16 & 91.3%). With tuning, the sensitivity of the model increased as well from 92.07% to 92.21%. Given the results shown. The tuned RBF SVM model was selected as the best for its type.

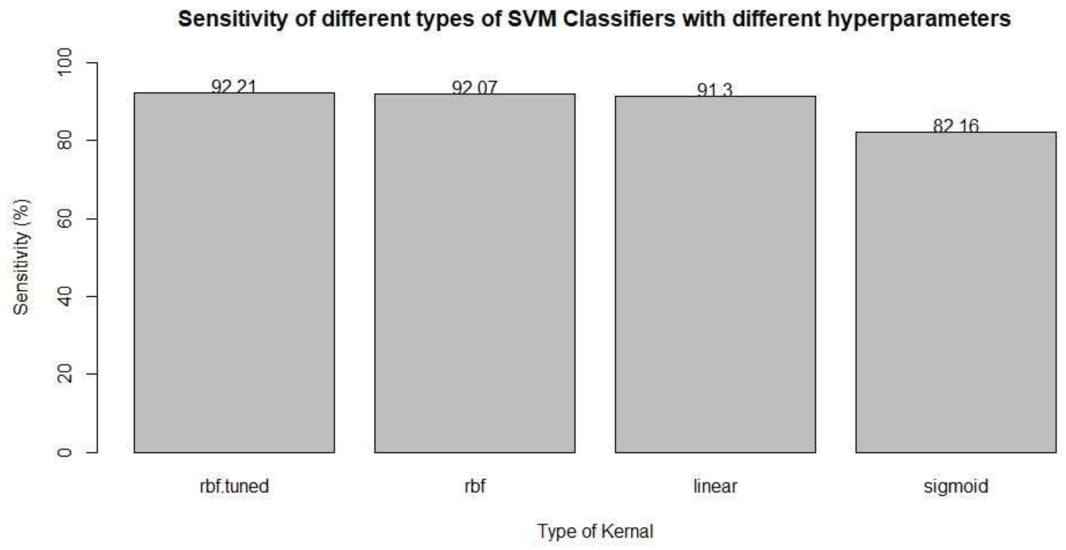


Figure 10: Sensitivity of SVM models

4.1.2 Artificial Neural Network

Similar to the previous model, the first step was to import all of the necessary library packages. For ANN, the following packages were installed.

No	R-Package	Function
1	Ggplot2	Used for graphical representation of data / model
2	Catools	Used for data splitting
3	Neuralnet	Used for implementing the neural network model
4	caret	Additional library installed for the use of the confusionMatrix() function

The data was imported, cleaned, labelled, and split similar to the SVM model. The initial step was to implement the basic neural network model to be set as benchmark and improved on. The neuralnet() function was used. The basic framework of the inputs for this function is shown as below

neuralnet(“DV”~”IV”, data = “TS”, hidden = NN, threshold = “TH”, stepmax = “SM”, act.fct = “AF”, linear.output = “FALSE”, err.fct = “CE”, likelihood = “TRUE”)

Parameter	**Parameter Function / Explanation**
DV & IV	Dependent Variable ~ Independent Variables
data = “TS”	Data frame to be used. In this case it will be the training set.
hidden = NN	Defines the number of neurons and layers. For this study this value is varied and is used as the hyperparameter tuning.
threshold = “TH”	Numeric value that specifies the threshold value for the error function when building the neural network model. The value acts as a stopping criterion when calculating the error during setting the weights. The value is set at 0.1.
stepmax = “SM”	Defined as the max number of iterations for neural network training. Once this value is reached during training, the neural network’s training process stops. The value is set at 1e+07 for this study.
act.fct = “AF”	The activation function that is used for predicting the final output for each layer based on the weights or the neurons. The activation function for this study is set as “logistic” for logistic function as it is suited for classification problems.
linear.output = “FALSE”	Set to FALSE since the problem is a classification problem. This parameter is set true only for regression problems.
err.fct = “CE”,	The function that is used for the calculation of the error when training the model. The error function is set as “ce” for cross entropy as it is suited for classification problems.
likelihood = “TRUE”	Parameter set to true to calculated information criteria AIC and BIC.

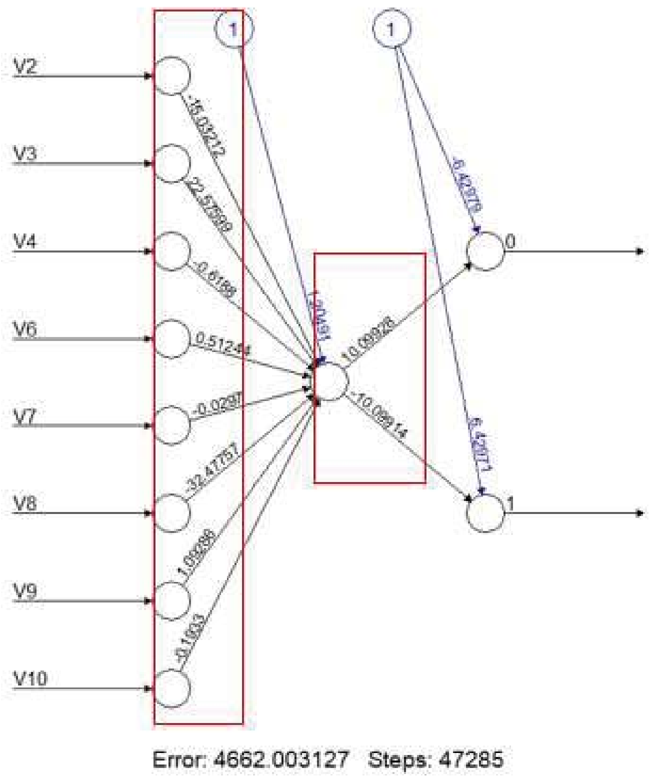


Figure 11: Basic Neural Network Model Structure with 1 neuron with 1 hidden layer

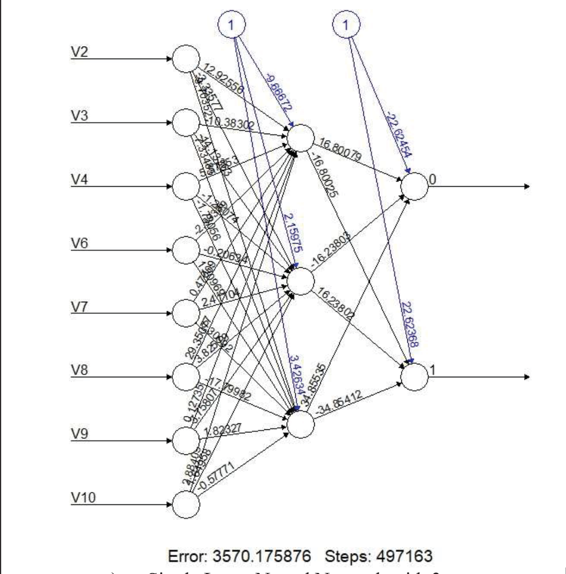
The basic neural network was first constructed with 1 neuron and a single hidden layer as shown in Figure 11. The structure of the neural network model is shown above in Figure 1. The plot above displays the weights of each input nodes to the node of the hidden layer and the weight of the hidden layer neuron to the output nodes. Weights are considered the strength of the connection between each neuron. It affects the amount of influence a change the input will have on the output. A low weight value indicates that a change in the input will have very little influence on the change of the output. Alternatively, a larger weight value indicates that a change in the input will result in more significant changes on the output. A negative weight indicates that an increase in this particular input value (X-value / independent variable) will decrease the probability for the particular output. The cross-entropy error is also displayed on the plotted network. This error gives a gauge of the deviation of the predicted outcomes from the actual outcomes based on the training set data. A large error value indicates a poorly fitted model and hence tuning needs to be performed by adjusting the number of nodes and neuron weights. The way the neural network works is that it first chooses a random value for the weights and predicts an outcome, which is then compared to the actual results and the error is evaluated. The weights are then readjusted to minimize the error, and this is done iteratively until the model achieves a satisfactory result for that particular model setup. The number of steps is also displayed and represents the number of iterations in which the neural network is trained. For the basic model, the neural network was trained for 47285 times. From the neural network plot above, it can be seen that input V2 (ApEN), V3 (SampeEn) and V8 (Variance) have greater weights in the model indicating that they have greater influence over the results during prediction [10].

Once the basic model was built, the compute () function was called with inputs of the model and the test set X values. The function returns the predicted value based on the inputs for each data instance for each network layer in the form of probability values. The outer layer predicted values are then taken and passed through the ifelse () function whereby a threshold probability of 0.5 was set in order to be classified as class 1, else class 0. The values are then converted to factors and compared to the actual results from the test set. The model prediction confusion matrix was then created, and the accuracy of the model was calculated similar to accuracy calculation in SVM. An additional parameter was also taken into consideration when evaluating the best neural network model which is the training time. As neural networks are quite a complex model and take a significant amount of time to train, the training time was also compared to one another and considered when choosing the final best model.

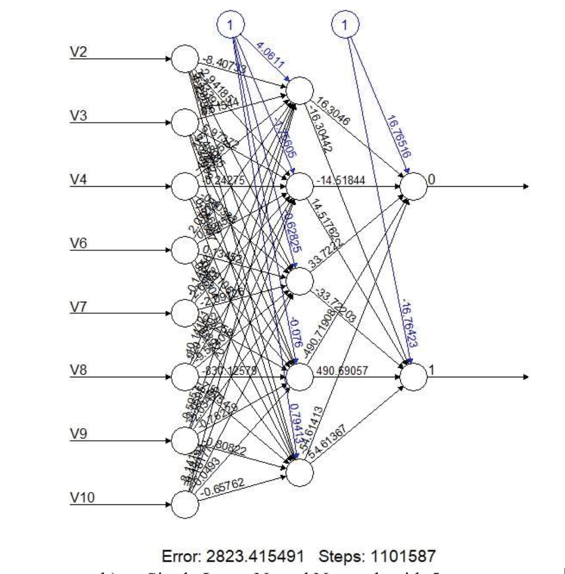
The basic ANN with a single neuron and hidden layer produced an accuracy of 89.51%. The model performed training with 47285 total steps and took an overall training time of 1.8 mins. As there are no automated way to determine the hyperparameters for the neural network model like SVM, in order to improve accuracy, the hyperparameter of number of neurons and hidden layers were manipulated manually to improve the accuracy of the model. Other parameters remained the same. 4 more additional models were trained. The basis of tuning was done on the premise that the basic model had only 1 neuron which was insufficient for the model to accurately train and predict the outcome. Hence, two steps were taken to increase the number of neurons in order to give the model more weightages and factors to consider during prediction. The first was to increase number of neurons within a single layer and the second was to increase the number of neurons with 2 hidden layers.

2 of the extra 4 models built had neurons of 3 and 5 within a single layer whereas the other two had 3+2 and 5+3 hidden layers. The maximum number of hidden layers was kept at 2 and the

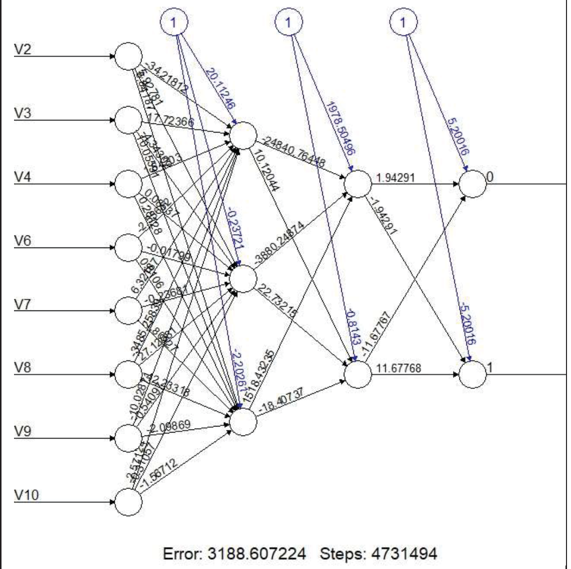
minimum number of neurons at the final hidden layer was kept at 2. A visualization of the additional NNs is displayed below in Figure 12 a-d



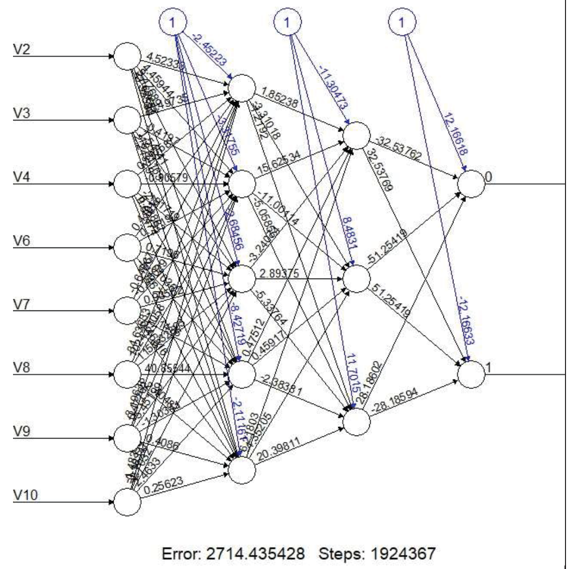
a) Single Layer Neural Network with 3 neurons



b) Single Layer Neural Network with 5 neurons



c) 2 Layered Neural Network with 3+2 neurons



d) 2 Layered Neural Network with 5+3 neurons

Figure 12: Visualizing the tuned neural network models (a-d)

From the plots and the results, the following observations can be noted.

a) Comparing only between neural networks with a single layer, as the number of neurons layer was increased, the cross-entropy error significantly reduced. The cross-entropy error for a single neuron of 4662.00 reduced to 3570.18 when the neurons increase to 3 and the cross-entropy error also further reduced to 2823.42 when the number of hidden layers were set to 5.

b) In terms of computational time and number of steps. A single hidden layer took 47285 steps and 1.8 mins to train, 3 neurons took 497163 steps and 31.44 mins whereas 5 neurons took 1101587 steps and 1.51 hours. It was observed that an increased number of neurons resulted in a higher computational time and the number of steps the model is trained. This indicates that the model learns more about the data with an increase of neurons.

c) Comparing the single layered neural network of 3 and 5 neurons to its dual hidden layer configuration, it can be seen that the cross-entropy error and overall accuracy did not significantly improve with the additional hidden layer. For 3 neurons vs 3+2 neurons, the cross-entropy error only reduced from 3570.18 to 3188.61 which translated to an overall accuracy improvement of only 1.39% (91.43% to 92.81%). For the 5 neurons vs 5+3 neuron neural network, the cross- entropy error only reduced from 2823.42 to 2714.44. However, the final accuracy showed that 5 neurons single layer neural network produced better results of 93.8% as compared to 93.33% for the 5+3 neural network.

d) In addition, the computational time significantly increased with the additional hidden layer (31.44 minutes vs 6.13 hours for 3 vs 3+2- & 1.51 hours vs 3.6 hours for 5 vs 5+3). Hence, it can be hypothesized that an increase of a hidden layer for this dataset did not significantly improve the accuracy of the model and its implementation was not justified given the trade-off for long computational time.

e) It was also observed that all the model put more weights on features ApEn, SampeEn and Variance as well as compared to other features which is consistent with the single neuron neural network.

These tuned models are then used to predict the test set outcomes and the predicted outcomes are compared to the actual test set Y-values. Results are shown in Figure 13 below. Based on the results obtained from predicting the test set, the models obtained an accuracy of 91.42%, 93.8, 92.81% and 93.33 respectively for 3, 5, 3+2 and 3+5 neural network model configurations. Training time on the other hand took 31.44 mins, 1.51 hours, 6.13 hours and 3.6 hours respectively. The results show that the neural network with 5 neurons produced the highest accuracy amongst the others.

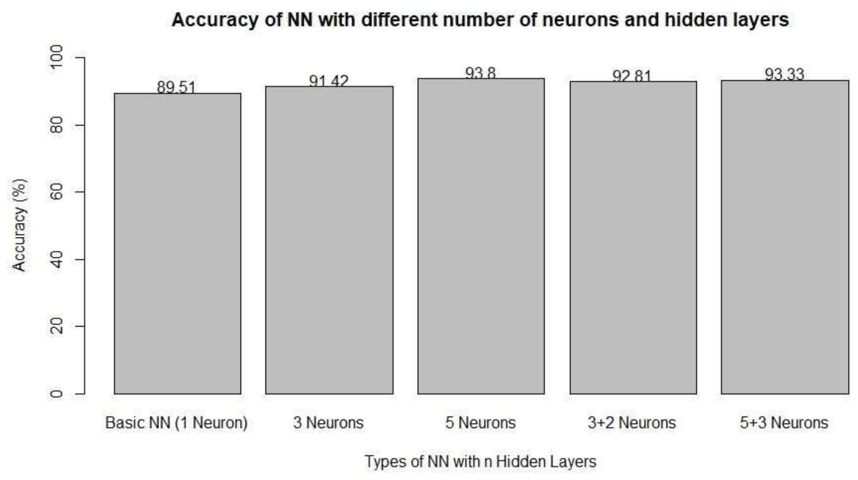


Figure 13: Accuracy of ANN models

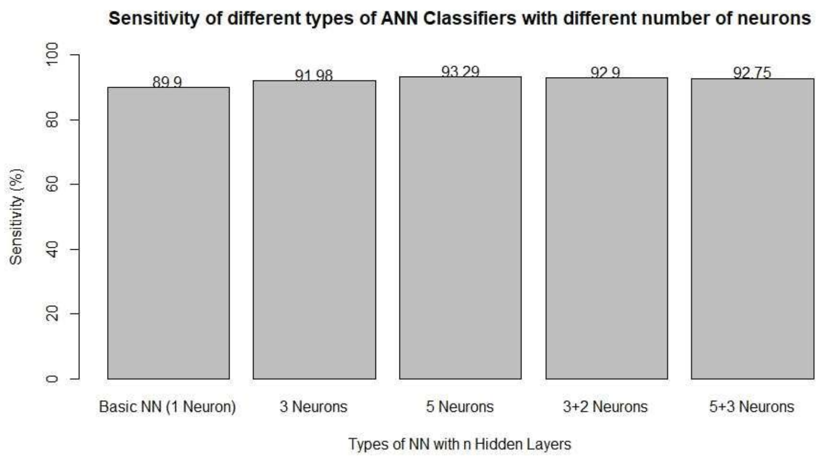


Figure 14: Sensitivity of ANN models

In addition, the sensitivity of the ANN models is presented as below in Figure 14. Based on the sensitivity results, it can be seen that when comparing between single layer neural network, an improved sensitivity is obtained as the number of neurons increases. When comparing between single layered network to its dual layer counterpart, the sensitivity did not significantly improve which shows a similar trend to accuracy. In overall, the neural network with 5 neurons in a single layer presented the best accuracy of 93.29%. Considering the results obtained in terms of accuracy, sensitivity and training time, the best neural network configuration is with 5 neurons in a single layer network and hence this model was selected as the best neural network model for the problem.

4.1.3 KNN Classifier

The first step was to import all of the necessary library packages. For KNN, the following packages were installed.

No	R Package	Function
1	Ggplot2	Used for graphical representation of data / model
2	Catools	Used for data splitting
3	caret	Used for implementing the KNN model and to compute the confusion matrix

The dataset was imported, labelled, and split similar to the SVM and ANN model. Then, the train() function was called from caret package to implement the KNN classifier model. The framework of model implementation in R is detailed as below.

knnFit. <- train(“DV” ~ “IV”, data = “TS”, method = "MODEL", trControl = “CF”, tuneLength = “XX”)

Parameter	**Parameter Function / Explanation**
DV & IV	Dependent Variable ~ Independent Variables
data = “TS”	Data frame to be used. In this case it will be the training set.
method = "MODEL"	Specifying which machine learning model to use. For this study, this is set to “knn” for KNN classifier.
trControl = “CF”	Parameters function that defines the method of training the model and the number of times the model is trained based on the training set data. This is one parameter that is manipulated for hyperparameter tuning of the mode. The method used is repeated cross validation with different numbers of cross validation.
tuneLength = “XX”	Defines the number of level / instances for each feature to be considered when classifying the new data point. For this study, this parameter was also modified in order to determine how this parameter affects the accuracy of the model

The basic KNN model was built using 3 times cross validation with tuneLength set to 20. The built model was then used to predict the test set data and the predicted y-values were compared to the test set actual y-values. The confusion matrix and accuracy were then computed. Based on the results the basic KNN classifier model produced 89.1% accuracy.

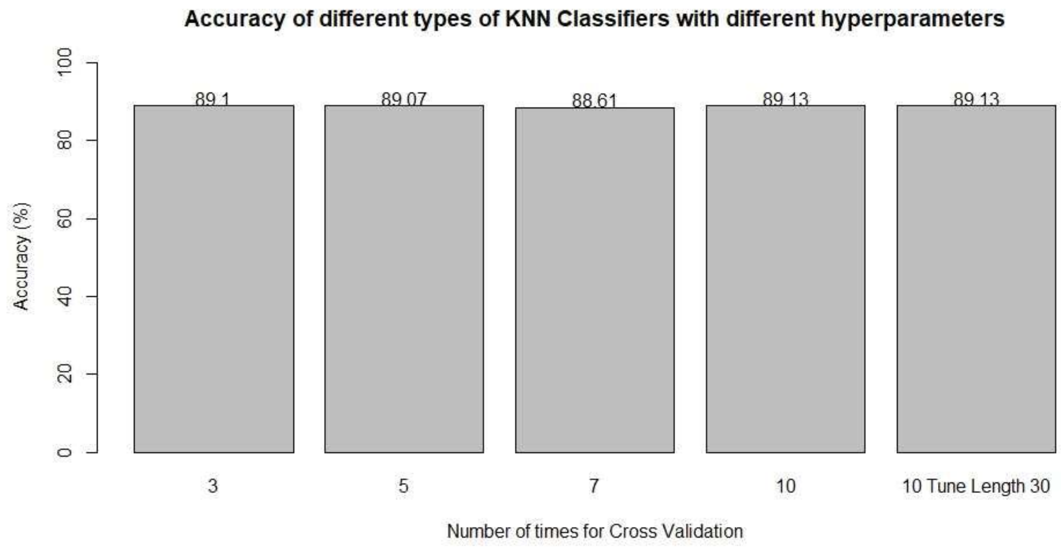


Figure 15: Accuracy of KNN models

In order to further tune the model and improve its accuracy, the number of times for cross validation was increased to 5,7, and 10 times respectively to observe the improvement in the models' accuracy. The accuracy results were tabulated and compared as shown in Figure 15. From the results, the model with 10 times cross validation showed the best accuracy as compared to the rest with an overall accuracy of 89.13%. In addition, efforts to further improve the accuracy of the model was done by increasing the tuneLength hyperparameter for the 10 times cross validated model from 20 to 30. The results were plotted and compared with one another as shown in Figure 15. From the results it showed that the accuracy of the model did not improve. It can be observed in Figure 16 & Figure 17 that though the range for tuneLength increased, however the optimum parameter value had already been reached at value k=23 which is within the tuneLength = 20 range. Hence, the tuneLength parameter of 20 was sufficient and need not be increased further. The k =23 results signify that when the model is classifying a new data point, it will consider the features and class values of the 23 closest data points (neighbours) to the new data point to be predicted. 23 is the optimum number of nearest neighbours to consider in order to obtain a highly accurate model.

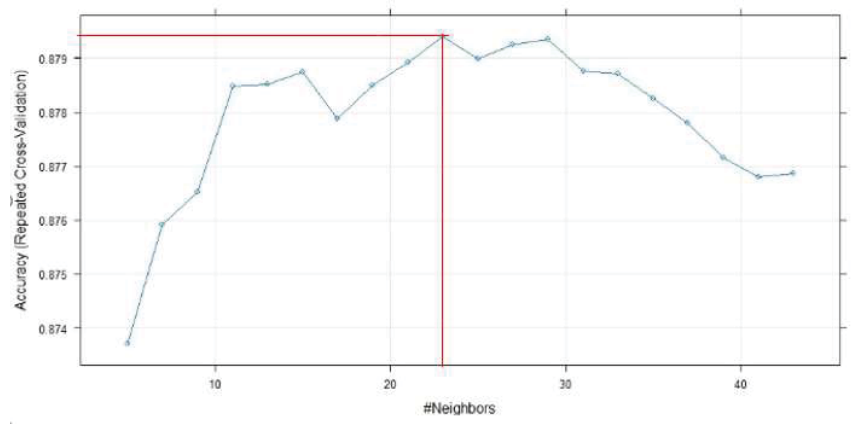


Figure 16 Accuracy vs #Neighbors (parameter that changes as tuneLength is modulated, tuneLength = 20)

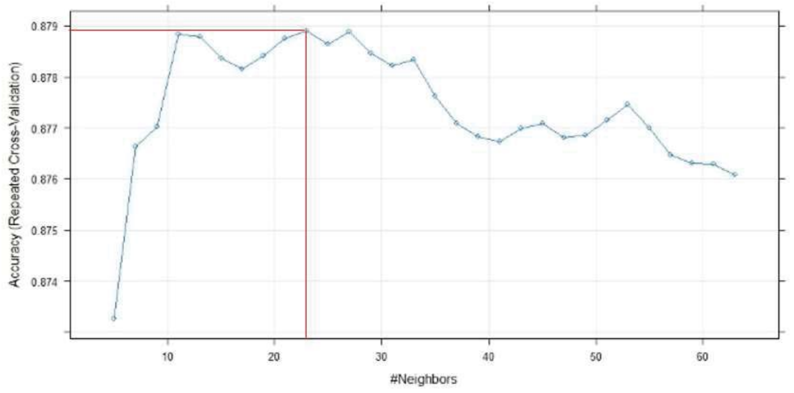


Figure 17 Accuracy vs #Neighbors (parameter that changes as tuneLength is modulated, tuneLength = 30)

In addition, the sensitivity results for the KNN classifier are presented as below. The results show that the sensitivity for all KNN models were almost similar to one another with a max deviation of 0.09% from the highest to the lowest. For this particular model, the model which attained the highest accuracy however did not attain the highest sensitivity. The best sensitivity was attained by the model with 3 times CV. As the CV times increase to 5 and 7, the sensitivity reduced to 88.12% and 88.17% respectively. From 7 times CV onwards to the final tuned model, the sensitivity did not fluctuate. The reduction can be attributed to the number of true positive values and false negative values that change as the model learns more about the training dataset. However, the differences of sensitivity are minor and negligible.

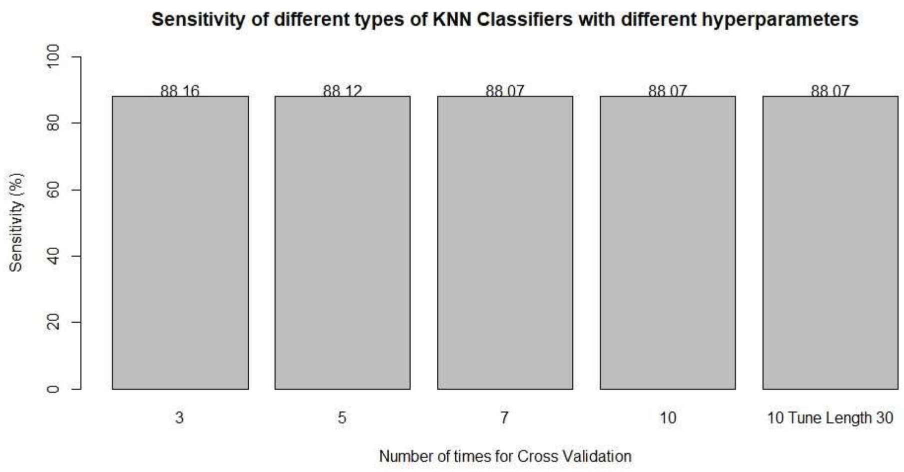


Figure 18: sensitivity of KNN models

Based on the results shown, the best KNN model selected is the KNN with 10 times CV and tune length 20. It attained an accuracy of 89.13% and a sensitivity of 88.07%

4.2 Comparing SVM vs ANN vs KNN

Once the best model was selected for each machine learning model class, the 3 best models were compared to one another in terms accuracy and sensitivity. The training and predicting time for each model is also discussed and compared to one another as well as the ROC curves for all 3 models.

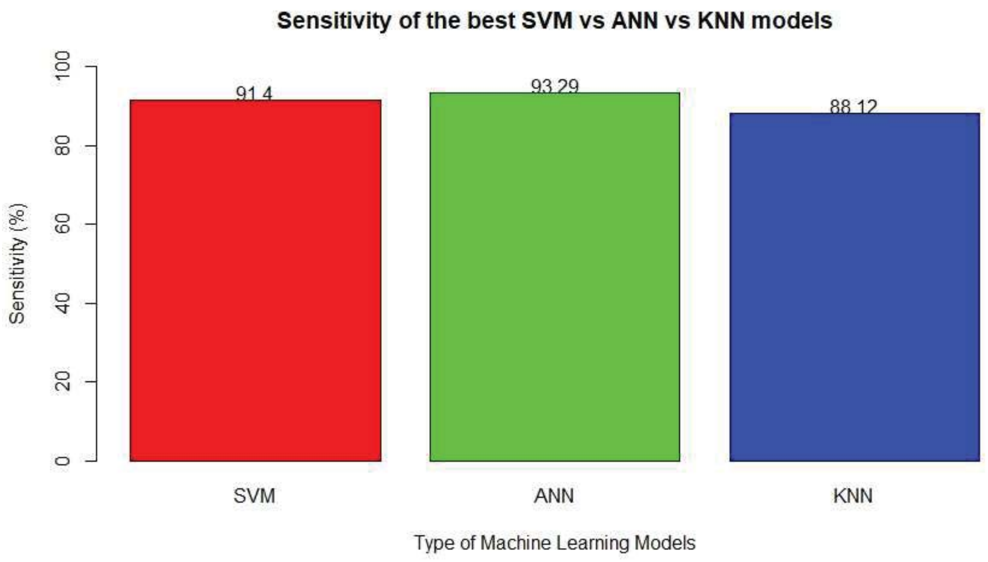


Figure 19: Comparing Sensitivity between SVM vs ANN vs KNN

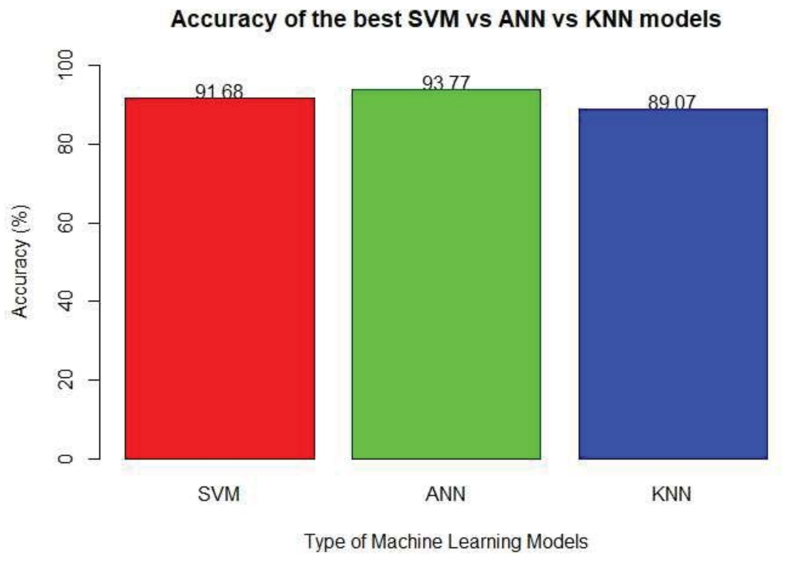
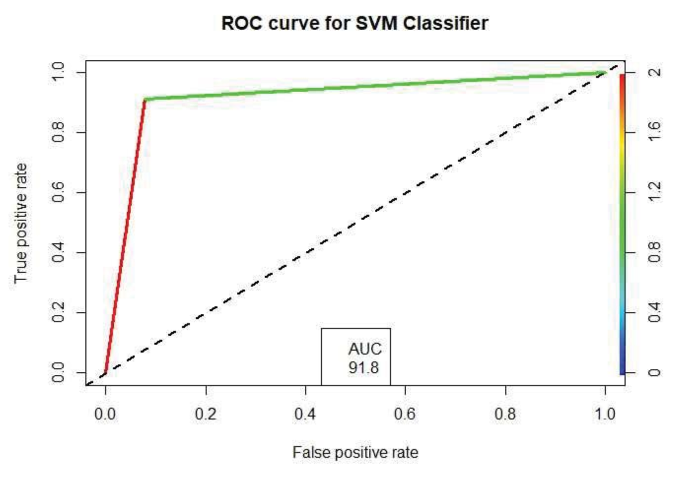


Figure 20: Comparing Accuracy between SVM vs ANN vs KNN

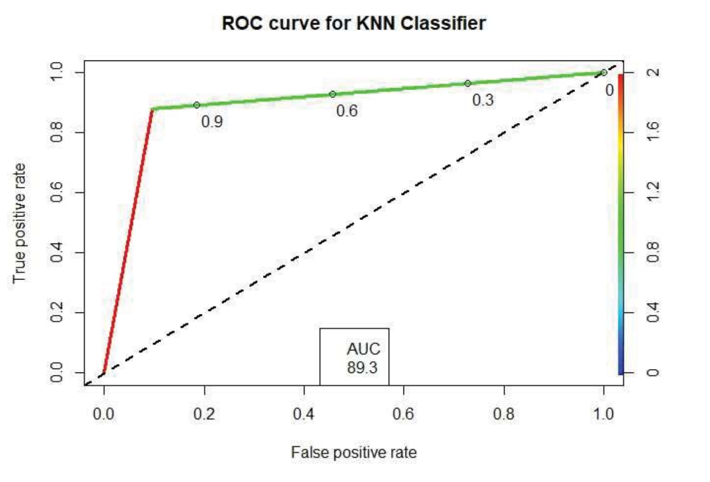
Based on the tabulated results in Figure 20, it can be seen that the best model in terms of accuracy is the Neural network model with an accuracy of 93.77% as compared to SVM and KNN respectively at 91.68% and 89.07%. In terms of sensitivity, the ANN model produced the best results as well with a 93.29% sensitivity followed by SVM at 91.4% and KNN at 88.12% as shown in Figure 19. The SVM model to 4.411229 s to train and 0.474731s to predict. The ANN model took significantly longer to train with 1.51 hours but had the fastest prediction time of 0.02194285 s. Finally, the KNN classifier model took 2.351132 mins to train and 0.313158 s to predict.

With SVM having the fastest model training time, it gives an indication that the data is in a multidimensional form which makes it more easily separable via a hyperplane decision boundary as the number of features get higher. ANN model took the longest to train due to its in-built learning of the training data to assign weights. The model assigns weights to the neurons and then implements back-propagation to evaluate the error cause by each node and subsequently reupdates the weights of the nodes such that the error is minimized. Neurons that contribute more to the error are given less weights whereas those that improve the accuracy are given better weights. This is done iteratively until a certain minimum error threshold is achieved. The size of the data and the number of features to be processed could explain the significantly longer training time. However, once the model was built and trained, it was capable of classifying the data the quickest (10 times faster compared to KNN and SVM). This coupled with its highest accuracy and sensitivity among the 3 models gives an indication of how well the model learned about the training data and its efficiency in classifying a new data. KNN model took a moderate amount of time to train. It took a longer time to train than SVM as the model had to compute the distance of each new data point instance to all training data points fed into the model and determine the optimum number of neighbouring data points to consider based on all training data points.

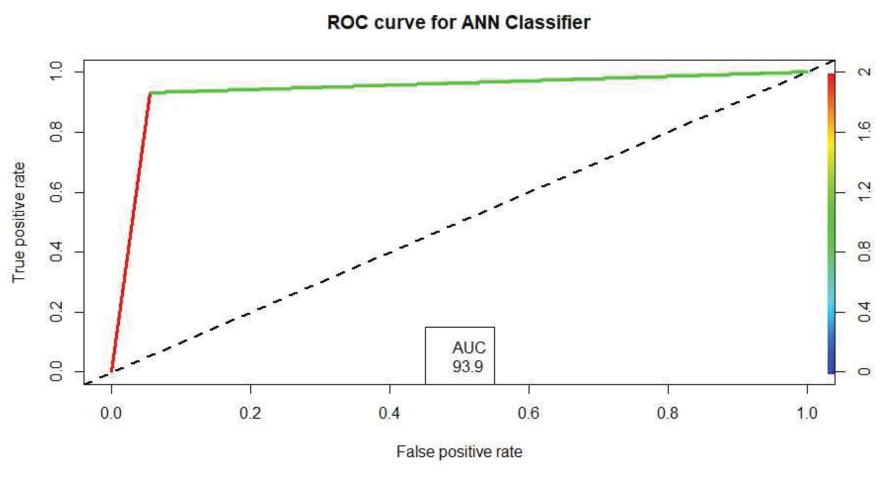
In addition, the ROC curves for the 3 best models were plotted as well for comparison and is shown below in Figure 21. The ROC curves and AUC values for all 3 models are good with the ANN classifier obtaining the highest AUC followed be SVM and KNN.



a) ROC Curve for SVM



b) ROC Curve for KNN



c) ROC Curve for ANN

Figure 21: ROC Curves for all 3 best models

The results obtained from this study agree well with the related literatures discussed in Part A. The results showed that the three classifiers SVM, ANN and KNN obtained a high and satisfactory level of accuracy in general. The results' value however variate by a certain level as compared to those obtained from other studies. This is due to the nature of the dataset being used, the features obtained, and the parameters set in the machine learning model. Another observation from this study that agrees with literature reviews is the computational resources required for Artificial Neural Network training [4]. The training took significantly longer than other models. This is attributed to the method in which the model trains using back-propagation. An improvement can be made to use other types of neural network from other packages such as keras and tensorflow which determines the weights of the model differently and has a wider options of activation functions to use. This can help in reducing computational time and resources and improve on the model's performance.

[11] paper results showed that KNNC is more accurate as compared to SVM model. However, the results in this study showed otherwise where SVM was more accurate. A reason for this can be the nature of the dataset, model parameters and features used. Another reason is attributed to the method in how both these models work in classifying the data. The data for this study is non-linearly related and has a high dimensional space. SVM works well with non-linear data and converts the data into a high-dimensional form making the dataset more distinguished and more separable between classes. This is done through different kernels built in the SVM model. The model also automatically learns about the data on its own and builds multiple hyperplanes and selects the best one considering the multidimensional features in the dataset. This coupled with the nature of the dataset could make it more easily separable with a decision hyperplane. KNN model on the other hand evaluates the classes of the close neighbours in the model's memory to the new dataset and evaluates the new data's class based on the most frequent class surrounding it. KNN is good in the sense that it is automatically non-linear [12]. However, KNN model has a downside in which it needs to be carefully tuned in terms of number of neighbours (K value) and the distance of the neighbours in order to perform with high accuracy. Hence an improvement is suggested, more robust testing needs to be carried out to find the optimum parameters to tune to model.

5. CONCLUSION

This paper has developed an automated machine learning classification algorithm for the diagnosis of epilepsy based on EEG data. The study began by taking a raw dataset of EEG readings which was in the form of a continuous time series and extracted statistical features from the time series which brought about meaningful information for interpretation. The extraction of statistical features also converted the dataset from an unstructured into a more structured data or numerical values which can be utilized by machine learning classification algorithms. Then, the features were filtered and used with 3 different machine learning classification algorithms of SVM, ANN and KNN. The best model for each type was developed and tuned and compared among the other two best machine learning models. From the results observed, the best model for diagnosing epilepsy based on EEG data analysis is Artificial Neural Network which achieved an accuracy of 93.77% as compared to SVM and KNN models. In addition, the Artificial Neural Network model also performed much better in terms of sensitivity (93.29%), AUC (93.9%) and prediction time (0.02194285 s). The results achieved by the Artificial Neural Network model is satisfactory is relatively good. Reflecting on the results and discussion, the aim of this study to devise a machine learning classification algorithm to diagnose epilepsy based on the analysis and classification of various combinations of features extracted from EEG signals had been achieved. The solution proposed if expanded further, will certainly help in creating a cheaper, effective, and practical solution in diagnosing Epilepsy at a very early on stage, something which is not a luxury that many have at the present moment. Furthermore, the objectives set in executing the study is also achieved and detailed in Table 2 below.

GOAL	**ACHIEVEMENT (YES / NO)**	**IMPLEMENTATION IN THE PROJECT**
Define a problem related to healthcare which can be solved by using machine learning techniques. For this particular study, the problem is devising a machine learning method to diagnose epilepsy based on analysis and classification of features from EEG signals.	Yes	- The topic of study was introduced, and the problem was defined in Part A of this study.

Conduct detailed literature review regarding the problem and related work / solutions to understand and propose a machine learning based solution.	Yes	- Detailed literature review was conducted in Part A from 15 different literatures related to Epilepsy, EEG data, Machine Learning models used for Healthcare problems and Machine Learning models used in diagnosis of Epilepsy. - The related studies were well understood in terms of problem statements, methodology, solution proposal and analysis. - Based on literature review and analysing the problem, the framework for the study was devised and proposed and 3 machine learning classifiers were selected for implementing the solution.
Gather data with a significant size to analyse and device a solution. The data should be raw data that is not perfectly cleaned and should be unbiased.	Yes	- An EEG data of Epileptic patients was obtained from the UCI Machine Learning repository. - The data was a non-linear time series and was unstructured, unbiased and was not perfectly cleaned. - The data had 11500 instances and 178 columns of voltage readings over a period of time.
Perform data analysis, visualization, and pre-processing.	Yes	- The non-linear time series was cleaned and pre-processed. Statistical analysis was used to extract 9 different useful features from the unstructured data to make it more meaningful and easier to analyse. - The data was visualized and analysed to get a picture of how the data points are spread out for each class and comparing their values. - Statistical Significance was calculated using Analysis of Variance to select relevant features.
The data for this study (EEG signals) is a continuous non-linear data. Hence, features had to be extracted based on statistical analysis. Features extract will be determined from literature review. The statistical significance of each feature will be calculated to determine the best combinations of features. Suitable features will then be selected as inputs into the machine learning algorithms.	Yes	

Define ratio of data split for training and testing.	Yes	- The ratio for training and test split was set at 70% training and 30% test for all models.
Input the features extracted (training and testing) into 3 different machine learning classification algorithms gathered from literature review and evaluate the algorithms' performance.	Yes	- 3 different models were tested with the training and test data which are SVM, ANN and KNN. - The basic model was first built and the results in terms of Accuracy and Sensitivity was analysed.
Based on the results obtained, perform tuning or cross validation steps in order to improve accuracy.	Yes	- Based on the basic model, further tuning was done for each model type and analysed. The best from each type was selected and compared between the other models. - The best 3 models were compared, and the final model proposed as the solution was selected which is ANN.

Table 2: Project Goals and Implementation

Albeit there are still room for improvements in the present study to further improve the accuracy of the models and the training time required. This can be achieved by using different packages of implementing the same models in R programming as other packages would have more features and processing capabilities. In addition, further analysis and tuning process of the models could also help improve the accuracy of the models which was not explored too deep in this study given time and resource constraints. Additionally, other types of classification models such as Decision Tree, Fuzzy Sugeno Classifier and Naïve Bayes classifiers can also be tested and compared with the results obtained. Additionally, more statistical features can also be included and parameters setting for feature extraction can also be more specific when using the functions. Last but not least, the current dataset used is static data which was recorded and analysed and not live readings. A great area of improvement will be giving the machine learning model capability to analyse and produce results in real time which can significantly improve the efficiency in diagnosing Epilepsy. Given additional time and resources, the following improvements can certainly be implemented.

6. REFLECTION

The author is satisfied with the work done in this study and the results it achieved. The main aim and goals were achieved which is devising a practical solution based on machine learning for diagnosing a real and very common disease, Epilepsy. The author gained a better understanding on data analysis, epilepsy, EEG data, programming on R-programming and particularly on the implementation of machine learning models and algorithms to solve real life problems driven by data. With the knowledge equipped and gained from this experience, it has definitely improved the author's skillset. This study also gave the author the opportunity to enhance his problem solving and researching skills. Furthermore, it was a great chance to tackle one of the major problems in healthcare and a good experience to contribute back to society. To conclude, the author is grateful for having carried out this study.

7. REFERENCES

[1] Arulsamy, A., Goh, B. H. and Shaikh, M. F. (2015) 'Current status of epilepsy in Malaysia and way ahead', *International Journal of Pharmacy and Pharmaceutical Sciences*, 7(1), pp. 2–5.

[2] EEG time series data (Department of Epileptology University of Bonn). http://www.meb.uni-bonn.de/epileptologie/science/physik/eegdata.html.(accessed November 2020).

[3] Acharya, U. R. *et al.* (2012) 'Automated diagnosis of epileptic EEG using entropies', *Biomedical Signal Processing and Control*. Elsevier Ltd, 7(4), pp. 401–408. doi: 10.1016/j.bspc.2011.07.007.

[4] Yuan, Q. *et al.* (2011) 'Epileptic EEG classification based on extreme learning machine and nonlinear features', *Epilepsy Research*. Elsevier B.V., 96(1–2), pp. 29–38. doi: 10.1016/j.eplepsyres.2011.04.013.

[5] Ocak, H. (2009) 'Automatic detection of epileptic seizures in EEG using discrete wavelet transform and approximate entropy', *Expert Systems with Applications*. Elsevier Ltd, 36(2 PART 1), pp. 2027–2036. doi: 10.1016/j.eswa.2007.12.065.

[6] Mudhiganti, Ramya Priya, "A Comparative Analysis of Feature Extraction Techniques for EEG Signals from Alzheimer patients" (2012). *Electrical Engineering Theses.* Paper 16. http://hdl.handle.net/10950/65

[7] Yamakawa, T. *et al.* (2017) 'Computer Simulation of Furniture Layout When Moving from One House to Another', in *Proceedings of the 33rd Spring Conference on Computer Graphics*. New York, NY, USA: Association for Computing Machinery (SCCG '17). doi: 10.1145/3154353.3154356.

[8] Hema Kashyap (2017) *No Title*, *Slide Share*. Available at: https://www.slideshare.net/hemak15/lecture-06-production-system#:~:text=structuring AI programs.-,2.,an expert thinking out loud (Accessed: 9 August 2020).

[9] Suri, J. S. (2009) 'Signals Using Nonlinear Parameters', 9(4), pp. 539–553.

[10] DeepAI (2020) *Weight (Artificial Neural Network)*, *DeepAI*. Available at: https://deepai.org/machine-learning-glossary-and-terms/weight-artificial-neural-network (Accessed: 19 December 2020).

[11] Fergus, P. *et al.* (2016) 'A machine learning system for automated whole-brain seizure detection', *Applied Computing and Informatics*. King Saud University, 12(1), pp. 70–89. doi: 10.1016/j.aci.2015.01.001.

[12] Max Miller (2019) *The Basics: KNN for classification and regression*, *Towards Data Science*. Available at: https://towardsdatascience.com/the-basics-knn-for-classification-and-regression-c1e8a6c955 (Accessed: 19 December 2020).

[13] Data Flair (2020) *Pros and Cons of R Programming Language – Unveil the Essential Aspects!*, *Data Flair*. Available at: https://data-flair.training/blogs/pros-and-cons-of-r-programming-language/ (Accessed: 19 December 2020).

[14] Navlani, A. (2019) *Neural Network Models in R*, *datacamp*. Available at: https://www.datacamp.com/community/tutorials/neural-network-models-r (Accessed: 19 December 2020).

[15] Shantha Selva Kumari, R. and Prabin Jose, J. (2011) 'Seizure detection in EEG using time frequency analysis and SVM', *2011 International Conference on Emerging Trends in Electrical and Computer Technology, ICETECT 2011*, (December 2015), pp. 626–630. doi: 10.1109/ICETECT.2011.5760193.

[16] 'A Comparative Analysis of Feature Extraction Techniques for EEG Signals from Alzheimer patients' (2012), 2012.

[17] UCI Machine Learning Repository (no date) *Epileptic Seizure Recognition Data Set*. Available at: https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition (Accessed: 19 November 2020).

ATTACHMENTS

**Attachments of codes (R-files) for this study has been e-mailed separately to lecturer.