**3.2.Software**

**3.2.1.Dataset Description**

In order to classify test subjects as alcoholics or normative, with a reasonable accuracy, the classifier has to be trained with several samples of the test subjects, and subsequently tested with more samples. These samples along with the appropriate labelling constitute the dataset.

**3.2.1.1.** The data in the dataset is an array of values which represent the ECG signal of the test subject.

**3.2.1.2.** The ECG data used has been recorded at the Autonomic Lab, Department of Neurophysiology, NIMHANS, Bengaluru. The data was recorded after taking informed consent adhering to Helsinki’s declaration.

**3.2.1.3.** At the Autonomic Lab , HRV is done using proprietary hardware and software setup by AD instruments, Australia. The product used in the lab premises for observing and recording HRV is PowerLab.

**3.2.1.4.** This device uses high sample frequency (of the order of 1kHz) to record electronic activity of the heart and other signals pertaining to other functions such as respiratory functions et cetera. The raw ECG data was extracted as five minute samples in European Data Format [EDF].

**3.2.1.3.** The dataset comprises of 67 samples, out of which 38 are ECG recordings of alcoholic test subjects and 29 are of normative test subjects.

**3.2.5.Classifiers**

**3.2.5.1.Extreme Learning Machine[ELM]**

Classification is the process of identifying which sub category (or class) a certain sample belongs to. Several classification algorithms have been developed and tested for a number of datasets, for example, Support Vector Machines, Naive Bayes Classifier and Neural Networks. Some classifiers perform better than others for a certain application, or for a specific dataset. There are a number of parameters that are looked into which selecting a certain classifier for a certain application, like accuracy, training time, testing time and for neural networks, the number of hidden layers, and the number of neurons in each layer. One of the classifiers that is chosen for this dataset is the Extreme Learning Machine [ELM].

**3.2.5.1.1.** A neural network is a processing unit consisting of sub units called neurons, which are interconnected to each other. These interconnects are assigned weights, representing the acquired knowledge, which may or may not be changed as the classifier is trained using the training samples. A neural network consists of an input and output layer, along with one or more hidden layers.

**3.2.5.1.2.** In machine learning, if the classifier is being trained using labelled data, that is, the corresponding target output is given for a certain input, it is known as supervised learning. If the classifier is being trained using unlabelled data, and clustering algorithms are required, it is known as unsupervised learning.

**3.2.5.1.3.** A feed forward network is one in which the values at the input are propagated towards the output through the hidden layers without being looped back to any preceding layers as an input.

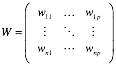
**3.2.5.1.4.** Back propagation consists of values that are fed forward through to the output, the error is calculated, and fed back to the preceding layers in order to correct the weights.

**3.2.5.1.5.** ELM is a single hidden layer, feed forward neural network. The weights connecting the input nodes to the hidden layer nodes are assigned randomly and never updated. The weights between the hidden nodes and output nodes are learnt in a single step. These models can learn several times faster than neural networks trained using the back-propagation algorithm. They are also known to produce decent generalisation capability.

**3.2.5.1.6.** The architecture of an ELM model is fixed a-priori, that is, the number of neurons in each of the layers is fixed before training. The number of input layer nodes is equal to the size of the feature set being input to the classifier. The number of output neurons is equal to the number of classes of samples being fed. The number of hidden layer neurons is chosen based on which value gives highest accuracy without over-learning.

**3.2.5.1.7.** Kernels are a computationally efficient way of changing the dimensions of the feature set of a sample using dot products. If a kernel is used before the classifier, the dimensions of the input layer will change to the size of feature set after kernel is applied. Examples of some popular kernels include Radial Basis Function [RBF] Kernel, Polynomial Kernel, Fisher Kernel, et cetera. The kernel in use in this project is the RBF kernel. It is given by equation( ), where x is the feature set of a training sample, 𝝁 is one of the kernel centres, and σ is the distance metric that is varied to check which gives best accuracy.

**3.2.5.1.8.** The dataset is split into training data and testing data. The training data is used to train the classifier, after which the test data is used to test the accuracy of the classifier. The percentage of training and testing data is varied in order to prevent under-learning and over-learning and obtain best accuracy.

**3.2.5.1.9.** The Input Weights and Biases of Hidden Neurons are generated and assigned randomly. These values are never changed. Let the Input Weights be designated as W and the Bias as B. The dimensions of Wis (n,k), where n is the number of hidden layer neurons and k is the dimension of the feature set of sample after passing it through a kernel. The dimensions of B is (n,p), where p is the number of input training samples given as a batch. If so, each column vector of matrix B will be identical.

**3.2.5.1.10.** The Input Weights are multiplied with the input training data and the Bias is added to it. Let the training data vector be designated as X. The output here is designated as Htemp, given in equation( ).

**3.2.5.1.11.** The Activation Function is used to calculate the output of each neuron in the hidden layer. Examples of Activation functions include Sigmoid, Sinusoidal, Hard Limit, Triangular Basis function, Radial Basis functions. Htemp is the input argument to the activation function, which results in H, as shown in equation( )

**3.2.5.1.12.** The activation function that is used in this project is the Sigmoid function, given by equation( ).

It is to be noted that H has the same dimensions as Htemp.

**3.2.5.1.13.** The Output Weights, denoted by Wo, are calculated by multiplying the Moore-Penrose pseudo-inverse of H with the targets of training data, denoted by T, whose elements represent the target values of each class of training data. This is represented in equation( ).



**3.2.5.1.14.** Once the classifier is trained, that is, all weights and biases are obtained, it is tested with the test sample data set.

First, the Input Weights, W, are multiplied with the Input Feature dataset (after passing through kernel function with centres as the same as that of training phase), given by Xtest, and added with the Bias, B, to obtain Htest as given by the equation( ).

**3.2.5.1.15.** The output of activation function is calculated by passing Htest as an argument to the activation function that was used in the training phase as well.

**3.2.5.1.15.** The actual output of testing data, Ytest is obtained by multiplying the output of activation function, H, with the Output Weights, Wo, which was assigned during the training phase. This is represented by equation( ).

**3.2.5.1.16.** The predicted output class is the index of the maximum value in the output vector Y, as given by the equation( ).

**3.2.5.1.17.** ELM has been found to be good at learning easy functions and performing well for small number of labelled data. They are also incredibly fast to train and have fewer parameters that need to be trained. They provide comparable accuracies to that of other classifiers like Support Vector Machines [SVM], and other deep neural network architectures, for a highly reduced training period and reduction in size of the neural network, owing to the single hidden layer and random initial assignment of input weights, which are not modified during the rest of the training process.

**3.2.6. Validation**

Validation is an incredibly important part of testing how well a classifier works. Having trained the classifier for a certain portion of the dataset, it is essential to find out how well the classifier performs when exposed to a set of test data it has not seen before. Sometimes, the data that a classifier is trained for might result in the classifier being biased towards a certain class and might perform poorly when given test data that belongs to a different class. So it is important to test the classifier using different test data sets or combinations of data for a classifier which has been trained for a corresponding different combination of training data set, so that upon averaging out the accuracies obtained from each train-test combination, we obtain an overall generalised picture of how the classifier performs in the real world and allows us to prevent over-fitting of the classifier. In cross-validation, the entire dataset is split into subsets of training and testing data, which are complementary to each other, and several rounds of training and testing are done, and results averaged over all the rounds of validation.

**3.2.6.1. Types of Validation**

There are two types of cross validation, namely Exhaustive Validation and Non-Exhaustive Validation. Exhaustive cross-validation consists of Leave-P-Out Cross-Validation and Leave-One-Out Cross-Validation. In exhaustive cross validation, all possible combination of samples of data set are chosen and used to train and test the classifier. Non-Exhaustive cross-validation consists of K-Fold Cross-Validation, Hold-Out Method and Repeated Random Sub-Sampling Validation. Non-exhaustive cross validation methods do not include all possible combinations of samples from the dataset for training and testing the classifier. These methods are an approximation of the leave-p-out cross validation method that falls under exhaustive cross-validation.

**3.2.6.2. Leave-One-Out Cross-Validation**

This is a type of exhaustive cross-validation and an explicit example of leave-p-out cross-validation for p=1, in which one sample from the dataset is chosen for testing and all other samples are chosen to train the classifier per iteration, and this procedure is iterated for as many times as there are samples in the dataset, only that the sample chosen for testing is different every single iteration. The accuracies over every single iteration are averaged over and an average accuracy is obtained. This method is better than leave-p-out cross-validation in that the number of combinations of train-test data are numerically equal to the number of samples in the dataset. For example, in a dataset having 100 samples, in each iteration, 1 sample is chosen as the test data while the other 99 samples are chosen as the training data.

**3.2.6.3. K-Fold Cross-Validation**

This is a type of non-exhaustive cross-validation in which the dataset is randomly partitioned into k number of equally sized subsets, and in each of k number of iterations, one of the subset is chosen as the test data set while the rest of the k-1 subsets constitute the training data set, and the subset chosen to test and the subsets chosen to train the classifier are unique in every iteration of training and testing. The accuracies obtained in each kth “fold” are averaged out to obtain an average accuracy that gives us an idea about the performance of the classifier. The advantage of this method of cross validation is that every single subset is used for both training and testing and a certain subset is used for testing only one time. The value of k determines the percentage of the dataset that is used for training and testing. For example, if k=4, for a dataset having 100 samples, the dataset is split into 4 subsets of 25 samples each. In each iteration, 75 samples are used for training and 25 samples are chosen as the test dataset.

This procedure happens 3 more times for the other 3 subsets of 25 samples, while the rest of the 75 samples in each fold participate as the training set. The accuracies over the 4 folds are averaged and presented as the accuracy of the classifier. When k equals the number of samples in the dataset, it becomes nothing but leave-one-out cross-validation.

**3.2.6.4. Confusion Matrix and Table of Confusion**

A confusion matrix is a table of values which provides us with a way to visualise the performance of an algorithm, or in the case of pattern classification problems, the performance of a classifier. The rows represent instances of the actual class while the columns represent instances of the predicted class. From this matrix, we can get to know how many times the classifier is classifying the sample correctly and and how many times a certain class of sample is misclassified as another class, and the number for each class. The table of confusion (also called the confusion matrix) is one consisting of 2 rows and 2 columns. The 4 elements of this matrix give us the number of True Positives, False Negatives, False Positives and True Negatives. True positives are those samples which were classified correctly for a specific class. False negatives are those in which the actual label was that of the class we are calculating the matrix for, and the classifier predicted wrongly. False positives are those in which the predicted label was that of the class for which we were finding the matrix for, but the actual label was that of another class. True negatives are those in which all the other classes apart from the class we were calculating the matrix for, were classified correctly.

|  |  |  |
| --- | --- | --- |
| Total Population | Prediction Positive | Prediction Negative |
| Condition Positive | True Positive | False Negative |
| Condition Negative | False Positive | True Negative |

**3.2.6.5. Sensitivity and Specificity**

Sensitivity is a measure of the proportion of positives that are correctly classified as positives. It is also known as recall, hit rate or true positive rate.

Specificity is a measure of the proportion of negatives that are correctly classified as negatives. It is also known as true negative rate.

Both of these are measures of the performance of a classifier for a certain dataset.