CHAPTER 1

INTRODUCTION

1.1 Electroencephalogram(EEG)

The electroencephalogram (EEG) is a recording of the electrical activity of the brain from the scalp. The recorded waveforms reflect the cortical electrical activity Signal intensity: EEG activity is quite small, measured in microvolts (mV). Signal frequency: the main frequencies of the human EEG waves are:

- **Delta**: has a frequency of 3 Hz or below. It tends to be the highest in amplitude and the slowest waves. It is normal as the dominant rhythm in infants up to one year and in stages 3 and 4 of sleep. It may occur focally with subcortical lesions and in general distribution with diffuse lesions, metabolic encephalopathy hydrocephalus or deep midline lesions. It is usually most prominent frontally in adults (e.g. FIRDA - Frontal Intermittent Rhythmic Delta) and posteriorly in children e.g. OIRDA - Occipital Intermittent Rhythmic Delta).
- **Theta**: has a frequency of 3.5 to 7.5 Hz and is classified as "slow" activity. It is perfectly normal in children up to 13 years and in sleep but abnormal in awake adults. It can be seen as a manifestation of focal subcortical lesions; it can also be seen in generalized distribution in diffuse disorders such as metabolic encephalopathy or some instances of hydrocephalus.
- **Alpha**: has a frequency between 7.5 and 13 Hz. Is usually best seen in the posterior regions of the head on each side, being higher in amplitude on the dominant side. It appears when closing the eyes and relaxing, and disappears when opening the eyes or alerting by any mechanism (thinking, calculating). It is the major rhythm seen in normal relaxed adults. It is present during

most of life especially after the thirteenth year.

- **Beta**: beta activity is "fast" activity. It has a frequency of 14 and greater Hz. It is usually seen on both sides in symmetrical distribution and is most evident frontally. It is accentuated by sedative-hypnotic drugs especially the benzodiazepines and the barbiturates. It may be absent or reduced in areas of cortical damage. It is generally regarded as a normal rhythm. It is the dominant rhythm in patients who are alert or anxious or have their eyes open.

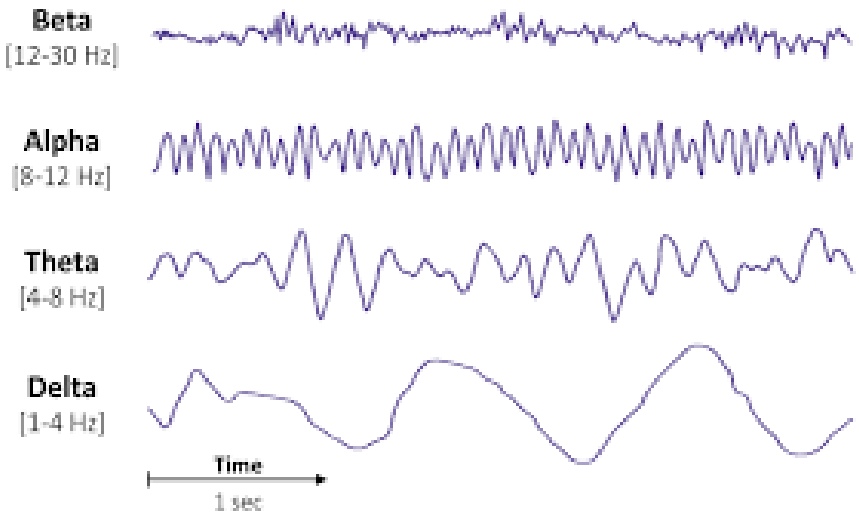


Fig 1.1 Threshold levels of EEG signals

1.2 Machine Learning

Machine learning (ML) is the scientific study of algorithms and statistical model that computer systems use to effectively perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model of sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the

task. Machine learning algorithms are used in a wide variety of applications, such as email filtering, and computer vision, where it is infeasible to develop an algorithm of specific instructions for performing the task.

Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

1.2.1 Approaches To Machine Learning

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

Supervised and semi-supervised learning

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and a desired output, also known as a supervisory signal. In the case of semi-supervised learning algorithms, some of the training examples are missing the desired output. In the mathematical model, each training example is represented by an array or vector, and the training data by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs. An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part

of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

Supervised learning algorithms include classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range. Similarity learning is an area of supervised machine learning closely related to regression and classification, but the goal is to learn from examples using a similarity function that measures how similar or related two objects are. It has applications in ranking, recommendation systems, visual identity tracking, face verification, and speaker verification.

Unsupervised learning

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms therefore learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to one or more pre designated criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity metric and evaluated,

for example, by internal compactness, or the similarity between members of the same cluster, and separation, the difference between clusters. Other methods are based on estimated density and graph connectivity.

Reinforcement learning

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. In machine learning, the environment is typically represented as a Markov Decision Process (MDP). Many reinforcement learning algorithms use dynamic programming techniques. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

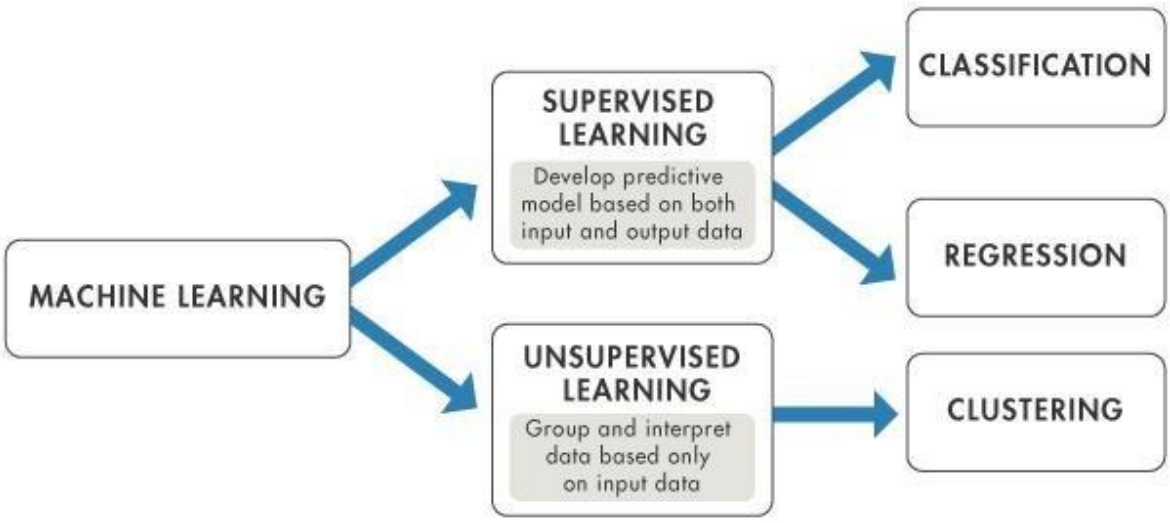


Fig 1.2 Types of Machine Learning Algorithm.

1.3 Support Vector Machine (SVM)

Support vector machines (SVMs), also known as support vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

1.4 LOGISTIC REGRESSION

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logit function.

1.5 NEURAL NETWORK

A neural network is a type of machine learning which models itself after the human brain. This creates an artificial neural network that via an algorithm allows the computer to learn by incorporating new data. While there are plenty of artificial intelligence algorithms these days, neural networks are able to perform what has been termed deep learning. While the basic unit of the brain is the neuron, the essential building block of an artificial neural network is a perceptron which accomplishes simple signal processing, and these are then connected into a large mesh network.

1.6 **Long short-term memory (LSTM)**

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections that make it a "general purpose computer" (that is, it can compute anything that a Turing machine can . It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. Bloomberg Business Week wrote: "These powers make LSTM arguably the most commercial AI achievement, used for everything from predicting diseases to composing music."A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

CHAPTER 2

LITERATURE SURVEY

2.1 Title: A First Step towards Eye State Prediction Using EEG

Authors: Oliver Rosler , and David Suendermann

This paper demonstrated that it is possible to predict eye state using EEG sensor input with an accuracy of more than 97%. The high accuracy and the fact that no special training is required suggest the use of eye state inferred from EEG signals for controlling tasks. However, the present study involved only a single subject which raises the question whether results are generalizable. We are currently investigating the presented technique's behavior across multiple probands including user independent training. Preliminary results of these experiments look very encouraging. To allow for applying the presented technology securely and effectively, the dependence of eye state prediction accuracy on other activities carried out by the subjects will have to be explored. Also other techniques for dimensionality reduction such as linear discriminant analysis could be useful to look at. Less sensors would reduce production cost of required EEG devices and also speed up instance-based classification.

2.2 Title: EEG Eye State Identification Using Incremental Attribute Learning with Time-Series Classification.

Authors: Ting Wang, Ka Long Man

This paper, a time-series classification approach based on IAL is proposed for EEG eye state identification. The approach is novel in a way that it firstly extracts features from the raw data and then sorts these features using IAL feature ordering

approach according to feature's discrimination ability. During the training process, the newly extracted features are imported into the neural predictive system in a sequential order based on the feature ordering. In comparison with the conventional batch-training methods and feature extraction method without considering the relation between time-series data, the experimental results of time-series IAL showed that such a machine learning approach can not only cope with time-series classification problems but also improve the accuracy of the classification results. Moreover, the experimental results also imply that the relation among time-series data is crucial to the data analysis in such classification problems.

2.3 Title: Eye state classification from electroencephalography recordings using machine learning algorithms.

Authors: Lukasz Piatek, Piatrek Fieldler.

This paper proposed Ten out of 23 tested classifiers fulfilled the determined requirements of high classification accuracy and short time of classification and can be selected as applicable for the process of real-time EEG eye state classification. Therefore, this article, we showed that it is possible to predict eye states using EEG recordings with an accuracy from about 96% to over 99% in a real-time system. Furthermore, all selected classifiers belong to supervised machine learning algorithms including decision rules and decision trees. Thus, all output models – for example, a set of decision rules induced by PART algorithm or other(s) – can easily be interpreted and further investigated by a person being an expert in the EEG domain.

2.4 Title: Ensemble Classifier for Eye State Classification using EEG Signals.

Authors: Ali Al-Taei.

This paper demonstrates that the results of the ensemble classification system are promising for the task of EEG based human eye state detection. For instance, a combination of two classification algorithms was applied (i.e. Kstar and RF) and the experimental result show that employing ensemble (multi-core) classifiers for the aim of eye state detection is better than the ordinary approach of using a single classification method. In addition, the proposed method's performance is higher than other previous methods. Furthermore, the proposed method's results highlight and encourage the area of real time eye state detection and the ability to work with incomplete/missing data.

2.5 Title: Eye State Prediction using EEG Signal and C4.5 Decision tree algorithm.

Authors: Sabarinathan Vachiravel

This paper presented an algorithm based identification of human thinking pattern using EEG sensor input for eye state prediction with 83.5% accuracy, which is acceptable for various practical applications and the model was built with
ten-fold cross-validation method. From the classification it is clear that feasibility of machine learning is greater with minimized usage of '10' EEG headset sensors where, the previous classification requiring higher number of sensors. The sensor 'P7' is found to be most significant among the 14 sensors used in the EEG headset. Similarly, many other EEG sensor data pattern can be predicted for number of human gestures or activities in future.

2.6 Title: Eye State Prediction Using EEG By Supervised Learning.

Authors: Neha Jain, Sandeep Bhargava, Savita Shivani, Dinesh Goyal.

This paper showed that data mining techniques can be used efficiently to model and predict eye state cases. The outcome of this study can be used as an assistant tool by neurologists to help them to make more consistent diagnosis of nerve disorder epilepsy. Furthermore, the resulting model has a high specificity rate which makes it a handy tool for junior neurologists to screen out patients who have a high probability of having the disease and transfer those patients to senior neurologists for further analysis.

2.7 Title: Eye State Prediction from EEG Data Using Boosted Rotational Forests.

Authors: Cameron R. Hamilton, Shervin, Khaled.

The ensemble constructed from the RRF and K* classifiers was the most accurate model with 97.4% accuracy. The reconstructed K* of retained its accuracy of 97.3%. The highest performing RRF model achieved an accuracy of 95.1% using 10 forests of 300 trees each, with each tree's decision nodes based on three features. The accuracy of the model did not appear to improve when the number of forests increased beyond ten, though 20 was the maximum number of forests tested within a model, due to the computational cost. The highest accuracy attained with the ada(RJ48F) was 97.2% when the rotational forest comprised often J48 trees was boosted for 50 iterations.

2.9 Title: Performance Analysis of Eye-State Characterization through Single Electrode EEG Device for Medical Application

Authors:SravanthK.Ramakuri,Sravanthkumar,Chinmay Chakraborty , sanchitag.

The component extraction from EEG signal has been measured in OpenVibe Platform. Initially we observe the crude EEG information for a specific time period. We have demonstrated to store the information through a Neurosky Mindwave gadget in CSV format file design. We would observe the subjects consideration or contemplation levels, engine aptitudes like Eye characterization for attention and meditation levels for health care monitoring.

2.10 Title: Eye state EEG signal classification using Complex Valued Neural Classifiers.

Authors: Keerthika.P, Sivachitra.M, Ponni Bala.M.

A recently developed PE-CELM (Phase Encoded Extreme Learning Machine), Fully Complex Valued Fast Learning classifier (FC-FLC) and ELM have been applied for motor imagery EEG signal classification. For every networks, the input weights are randomly selected and the output weights are attained in an analytical way. The performances obtained with the two complex valued classifiers are compared with results of real valued ELM network. Complex valued classifiers accomplish better performance than ELM real valued classifier.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

- The existing system designed a predictive model for eye state prediction using data mining techniques.
- The existing system used Supervised learning algorithms such as K star, SVM Linear, SVM Polynomial, SVM RBF from Electroencephalogram.
- Report dataset capable of enhancing the reliability of eye state.
- The performances of the models were evaluated using the standard metrics of accuracy, precision, recall and F- measure.10 Fold Cross Validation was adopted for randomly sampling the training and test data samples.
- AWS allows to increase the speed of research by running high performance computing in the cloud and to reduce costs by providing Cluster Compute or Cluster GPU servers' on-demand without large capital investments.
- Every Supervised learning algorithm is tested with 5 and 10 attributes for comparing accuracy level.
- Among all algorithms, the accuracy level achieved by SVM based algorithms are lower than K star Instance based classifier.

3.1.1 Limitations

- Missing values, noisy data, inconsistencies, and outliers presented a challenge in this process.
- Low performance computing. The classification accuracy and time complexity should be improved.

CHAPTER 4

PROPOSED SYSTEM

4.1 Proposed System

The proposed system uses different supervised machine learning algorithms for comparing the accuracy level to predict eye state of different persons. In this system algorithms such as Support Vector Machine(SVM), Logistic Regression, Neural network, Long short-term memory (LSTM) were used for both classification and prediction. For determining the result data sets are collected using EEG signals. Electroencephalogram (EEG) is the recording of the electrical activity of the brain from the scalp. The recorded waveforms reflect the cortical electrical activity. After collecting the EEG waves every data is tested with different algorithms used in this work. Based on the parameters to be considered for classifying the eye state every algorithm test the data sets. The parameters to be considered are TP rate, FP rate, TN rate, FN rate, Precision(Eye open),Precision(Eye close), F-measure(Eye open),F-measure(Eye close),AUC, Classification speed. Based on all the performance of all algorithms were evaluated using standard metrics. Finally the accuracy level is compared for all algorithms to find the algorithm which gives better result.

4.1.1 Advantages

- Increased number of algorithms were used for comparing to estimate the performance.
- Increased accuracy level and time complexity.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

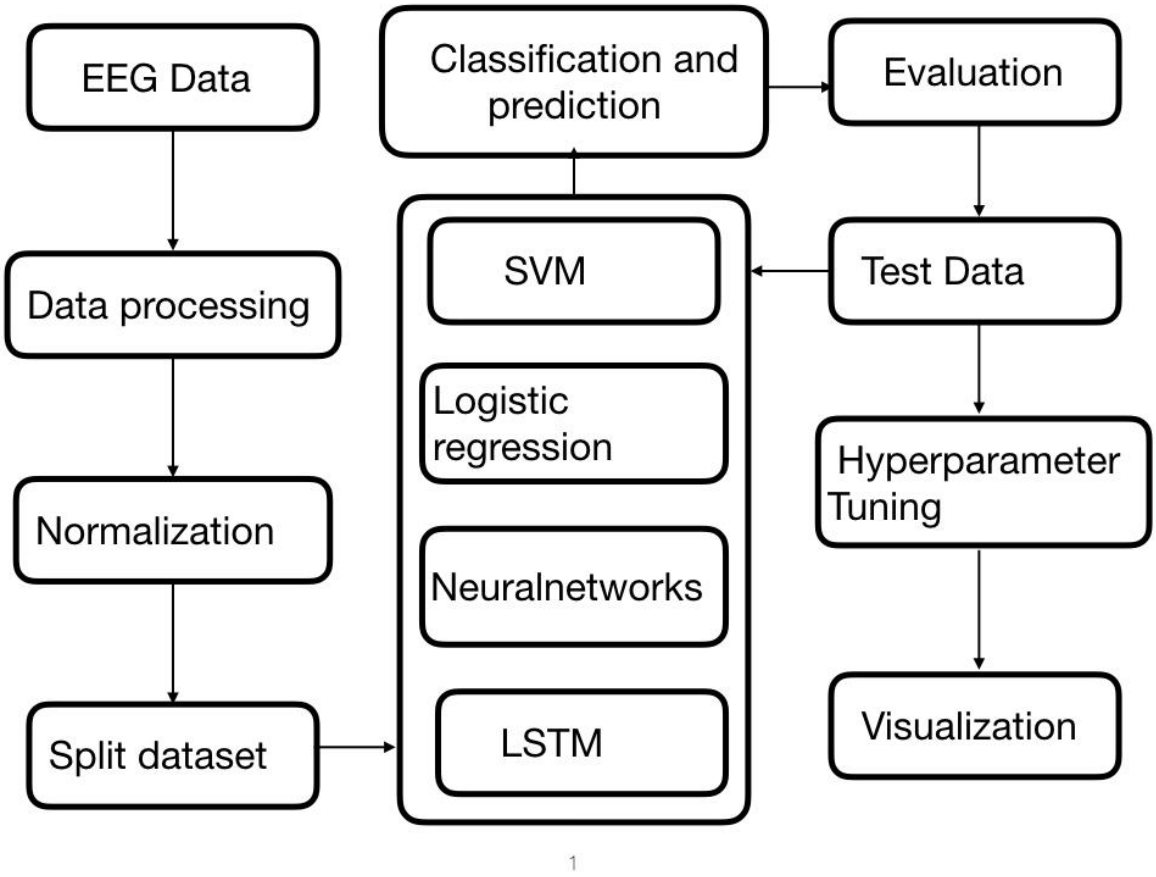


Fig 5.1 Architecture of proposed system

The above figure 5.1 shows the architecture of proposed system. The input data's are taken as text from EEG data set and are classified using four different algorithms to compare the accuracy level.

5.2 MODULE DESCRIPITION

The proposed system consist of some modules such as

- Data collection
- Data Preprocessing
- Supervised Learning Algorithms
- Evaluation
- Hyper Parameter Tuning

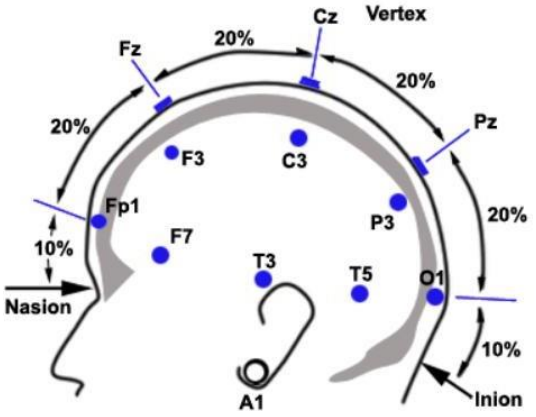
5.2.1 Data Collection

The electroencephalogram (EEG) is a bio-signal just like MRI, CT, f-MRI, ECG. It is the recording of the electrical activity of the brain from the scalp. Whenever we do some activities neurons are fired in the brain which generates some current, we use electrodes on the scalp and measure that very tiny current and then amplify it. In various researches, it is found that our brain generates different EEG time-series patterns for different activities like moving hands, legs, eyes opening and closing etc. So if we use machine learning to recognize those patterns then we can easily classify them.

Datasets

EEG Eye State Data Set
(http://archive.ics.uci.edu/ml/datasets/EEG+Eye+State)

Each row consist of 14 EEG values (means 14 time-series values) representing 14 electrodes (TF7, O2, F3, P8, T8, F4, FC6, AF3, FC5, T7, F8, P7, O1 and AF4) along with a value indicating the eye state.



5.2.1 Figure: Electrodes Placement

Labels - '1' indicates eye-closed.
'0' indicates eye-open.

Input - 14 EEG values

Output - '1' or '0'

Missing Data – 0

Data Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

Data preprocessing is used database-driven applications such as customer relationship management and rule-based applications (like neural networks).

Step 1: Import Libraries.

Step 2: Import the Dataset.

Step 3: Taking care of Missing Data in Dataset.

Step 4: Encoding categorical data.

Step 5: Splitting the Dataset into Training set and Test Set.

Step 6: Feature Scaling.

5.2.3Supervised Learning Algorithms

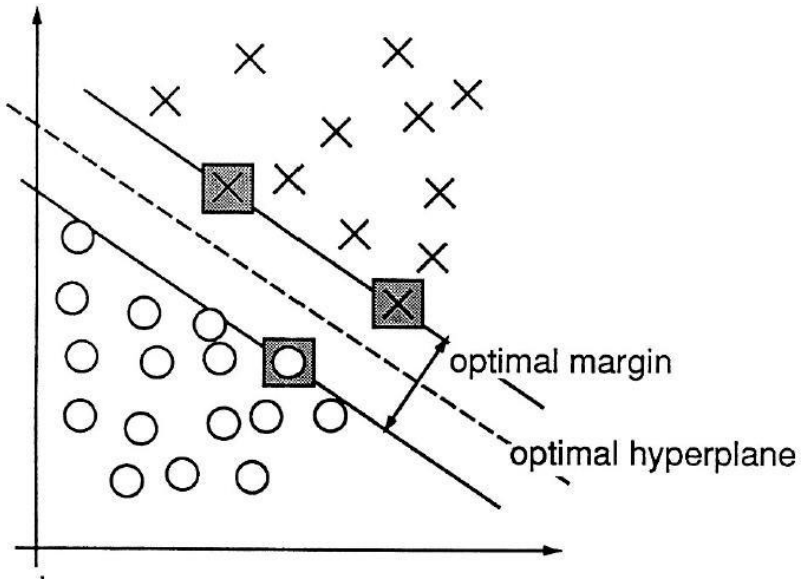
The supervised Learning Algorithms used in this project are

- Support Vector Machine(SVM)
- Logistic Regression
- Neural Networks
- LSTM

Support Vector Machine(SVM)

SVM maps the input data to high dimensional feature space through kernels, so that it can be separated by a hyperplane.

Justification - Because of different kernels SVM will provide good accuracy in our problem. In implementation I have used 'rbf' kernel which was giving better results than other kernels (poly, linear).



5.2.3 Figure: SVM Algorithm

Support vector machines (SVMs), also known as support vector networks, are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other. An SVM training algorithm is a non-probabilistic, binary, linear classifier, although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Logistic Regression

Logistic Regression is another technique for classification, which use sigmoid for two class classification and uses cross- entropy function as a loss function.

Justification - Logistic Regression are used as benchmark models because of simple algorithm with non-linear sigmoid function they provide good baseline accuracy.

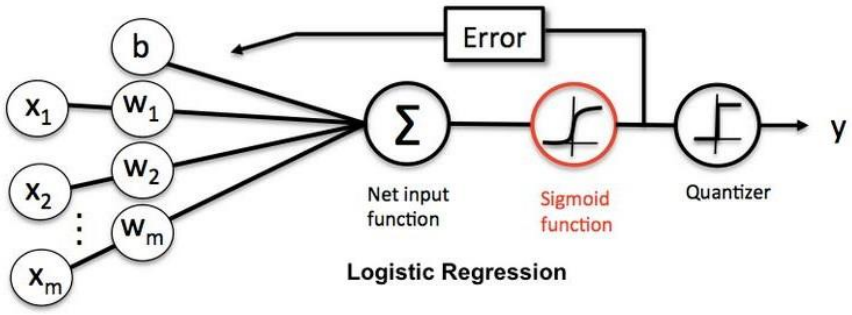


Fig 5.2.4 Logistic Regression

Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes). It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables. To represent binary / categorical outcome, we use dummy variables. You can also think of logistic regression as a special case of linear regression when the outcome variable is categorical, where we are using log of odds as dependent variable. In simple words, it predicts the probability of occurrence of an event by fitting data to a logistic function.

Neural Networks

Artificial Neural Networks are inspired from our Brain Neural Networks. ANN randomly assigns weights to all layers then use back propagation for decreasing the error rate and updating the weights.

Justification - Deep Neural Networks have specialty of function approximation

on lot of different type of data. Best part of neural network is that at each layer automatically important features get extract, that's what make neural network great in field of Machine Learning.

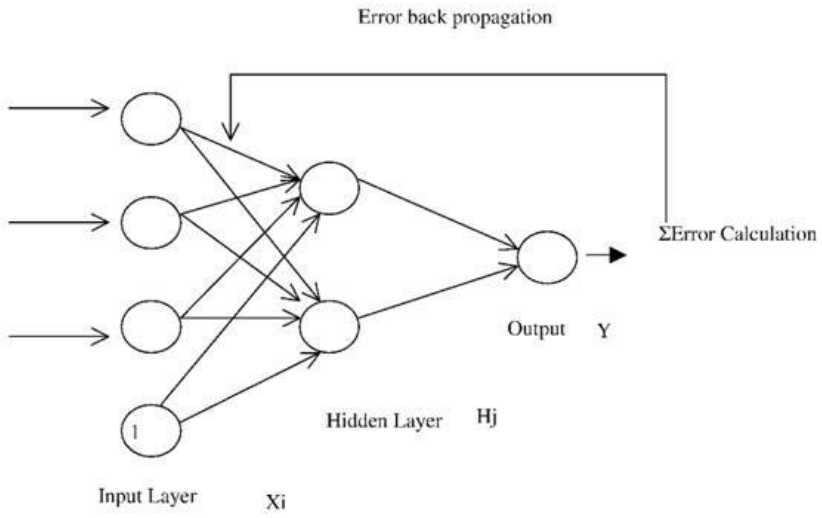
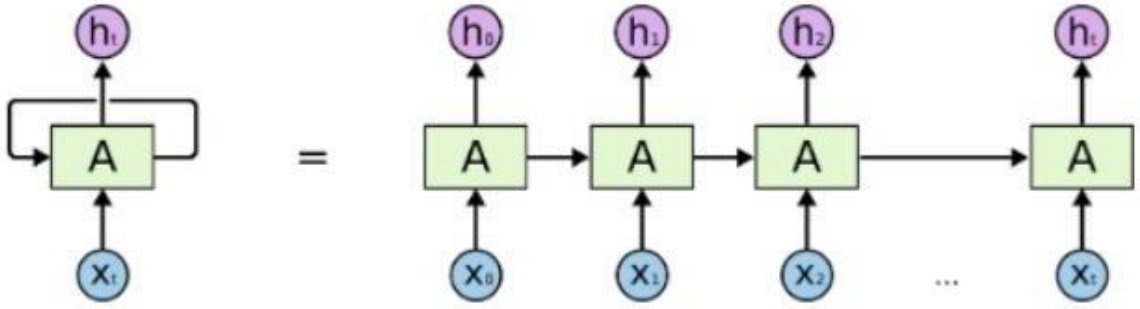


Fig:5.2.5 Neural Network

Recurrent Neural Network

The basic idea of RNN is to make use of sequential information. In traditional neural networks the data was assumed to be independent, but as we know data points in Time Series are not dependent, future values depend on previous values. A RNN has loops in them that allow information to be carried in neurons while reading the inputs but RNN has few drawbacks like vanishing and exploring gradient problem.



An unrolled RNN

Fig 5.2.6 RNN structure

LSTM (Long Short Term Memory)

To overcome vanishing gradient problem of Recurrent Neural Network, LSTM was made, which uses a different function to compute the hidden state. LSTMs also have chain structure but instead of having a single neural network layer, it uses 4 – cell, an input gate, an output gate and a forget gate as shown in figure.

Justification - LSTM will be very useful in our problem as our data is time – series data, it can capture recurrence relation of the data.

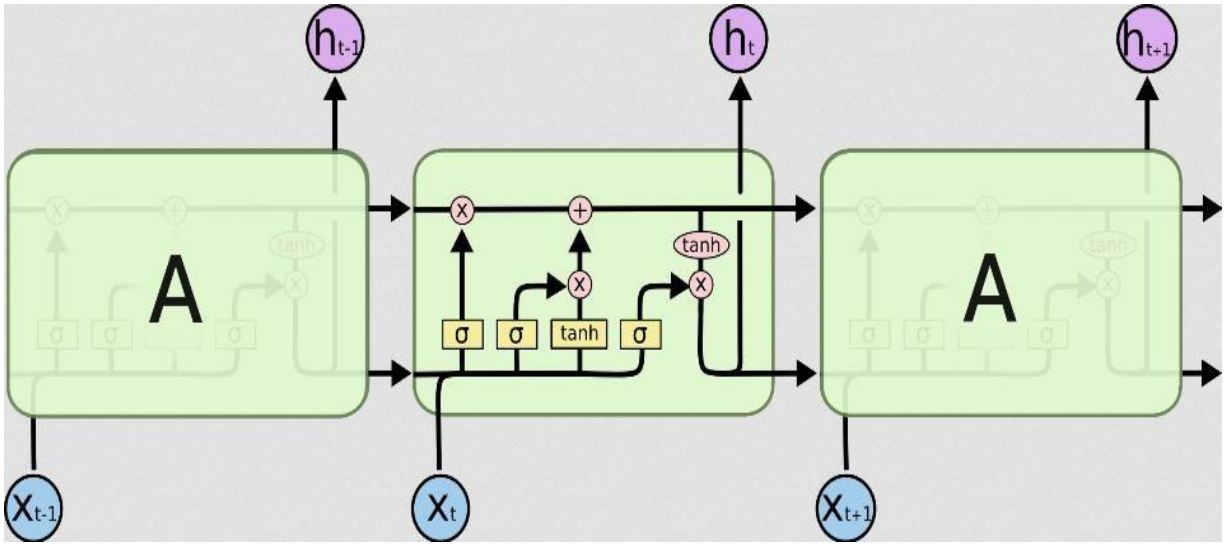


Fig 5.2.7 LSTM architecture

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture] used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections that make it a "general purpose computer" (that is, it can compute anything that a Turing machine can. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented,connected handwriting recognition or speech recognition. Bloomberg Business Week wrote: "These powers make LSTM arguably the most commercial AI achievement, used for everything from predicting diseases to composing music."A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration

between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

5.2.4Evaluation

Evaluating machine learning algorithm is an essential part of any project. The model may give satisfying results when evaluated using a metric accuracy *score* but may give poor results when evaluated against other metrics such as logarithmic l*oss* or any other such metric. Most of the times we use classification accuracy to measure the performance of our model, however it is not enough to truly judge our model. The different types of evaluation metrics available are

- Classification Accuracy
- Logarithmic Loss
- Confusion Matrix
- Area under Curve
- F1 Score
- Mean Absolute Error
- Mean Squared Error

5.2.5 Hyper Parameter Tuning

Most Machine Learning models have hyper-parameters. Once a user has specified the values of its hyper-parameters, the algorithm will be trained on the underlying data set. The choice of the hyper-parameters will affect the duration of the training and the accuracy of the predictions. Unfortunately, choosing hyper-parameters isn't straight forward as the number of possible

configurations explodes with their number and the evaluation time of a configuration is function of the complexity of the model and the size of the underlying training data.

Hyper parameter tuning relies more on experimental results than theory, and thus the best method to determine the optimal settings is to try many different combinations evaluate the performance of each model. With so much uncertainty it is understandable that most Data Scientists forego hyper-parameter tuning and stick with the default values provided by the model. This works in most cases, but to pick up on my introductory analogy, you might struggle to run a marathon in stilettos. The second most popular approach is to find the optimal configuration through grid-search (evaluating every single configuration) which will evantually lead to an answer. The third most popular approach is to find the optimal configuration through random-search, but this is only marginally better than grid-search. I'll spare you with an analogy for this one as I hope by now you see my point.

CHAPTER 6

SYSTEM SPECIFICATION

6.1 SOFTWARE REQUIMENTS

- COLAB
- JUPYTER NOTEBOOK

6.2 LANGUAGES USED

- PYTHON

6.3 LIBRARIES USED

- KERAS
- PANDAS
- NUMPY
- SCIPY
- SCIKIT

6.4 ABOUT SOFTWARE

COLAB

Google Colab is a free cloud service and now it supports free GPU.

- improve Python programming language coding skills.
- develop deep learning applications using popular libraries such as Keras, TensorFlow, PyTorch, and OpenCV.

The most important feature that distinguishes Colab from other free cloud services

is: Colab provides GPU and is totally free.

Benefits of Colab

Besides being easy to use (which I'll describe later), the Colab is fairly flexible in its configuration and does much of the heavy lifting for you.

- Python 2.7 and Python 3.6 support
- Free GPU acceleration
- Pre-installed libraries: All major Python libraries like TensorFlow, Scikit-learn, Matplotlib among many others are pre-installed and ready to be imported.
- Built on top of Jupyter Notebook
- Collaboration feature (works with a team just like Google Docs): Google Colab allows developers to use and share Jupyter notebook among each other without having to download, install, or run anything other than a browser.
- Supports bash commands
- Google Colab notebooks are stored on the drive.

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, macOS and Linux. Anaconda® is a package manager, an environment manager, a Python/R data science

distribution, and a collection of over 1500+ open source packages. Anaconda is free and easy to install, and it offers free community support. Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window.

Conda

Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.

Conda can be combined with continuous integration systems such as Travis CI and AppVeyor to provide frequent, automated testing of your code. The conda package and environment manager is included in all versions of Anaconda and Miniconda, Anaconda Repository. Conda is also included in Anaconda Enterprise , which provides on-site enterprise package and environment management for Python, R, Node.j.

Applications Available in Navigator

JupyterLab

Jupyter Notebook

QTConsole

Spyder

VSCode

Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

6.5 ABOUT LANGUAGE

PYTHON

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions- Python 2 and Python 3. Both are quite different. Some of the advantages of using python:

- Emphasis on code readability, shorter codes, ease of writing
- Programmers can express logical concepts in fewer lines of code in comparison to languages such as C++ or Java.
- Python supports multiple programming paradigms, like object-oriented, imperative and functional programming or procedural.
- There exist inbuilt functions for almost all of the frequently used concepts.

Features

- Interpreted
- Platform Independent

- Free and open source, Redistributable
- Embeddable
- Robust
- Rich Library support

6.6 ABOUT LIBRARIES

KERAS

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML.Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System),and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics Processing Units (GPU) and Tensor processing units (TPU).

PANDAS

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Pandas is a NumFOCUS sponsored project. This will help ensure the success of development of *pandas* as a world-class open-source project, and makes it possible to donate to the project.

NUMPY

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

SCIPY

SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linearalgebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack

which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack. SciPy is also a family of conferences for users and developers of these tools: SciPy (in the United States), EuroSciPy (in Europe) and SciPy.in (in India). Enthought originated the SciPy conference in the United States and continues to sponsor many of the international conferences as well as host the SciPy website.

SCIKIT

Scikit-learn is a free software machine learning library for the Python programming language. It features various classifications, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

In training any neural network very important topic is its convergence of loss function. Convergence helps us to know whether the model is under-fit or over-fit. For comparing convergence four algorithms were used. Among all accuracy level is checked to determine the better performance algorithm. Every algorithm is tested using data sets to give better result. LSTM algorithm gives better performance and accuracy.

7.2 FUTURE WORK

The main reason for LSTM and Neural Network not able to accuracy greater than 75% is EEG signals are non stationary signals means there frequency keep on changing. To further improving the model we need to use signal processing techniques and take it to spatial domain (Fourier Transform) or Time – Frequency Domain to better capture those features.

APPENDIX I

#Loading the files in Google colab

```
pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
fid = drive.ListFile({'q':"title='EEG Eye State.txt'"}).GetList()[0]['id']
f = drive.CreateFile({'id': fid})
f.GetContentFile('EEG Eye State.txt')
```

 # Loading the dataset

```
fname = "EEG Eye State.txt
with open(fname) as f:
content = f.readlines()
content = [x.strip() for x in content]
content = [x.split(",") for x in content]
```

 # Converting list to numpy array

```
import numpy as np
content = np.array(content, dtype = 'float32')
```

 # Shuffling the dataset

```
import random
random.shuffle(content)
```

 # Storing results of algorithms

```
score_p = []
```

#Training on Logistic Regression

```
 LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
 verbose=0, warm_start=False)
```

#Trainig on SVM

```
from sklearn.svm import SVC
clf = SVC()
clf.fit(X_train1, y_train1)
```

#Training on Neural Network

```
from keras.callbacks import ModelCheckpoint
checkpointer = ModelCheckpoint(filepath = 'MLP.weights.best.hdf5', verbose = 1, save_best_only = True)
hist = model.fit(x_train, y_train, epochs = 100, batch_size=256, validation_split = 0.1, callbacks = [checkpointer], verbose = 2, shuffle = True)
```

#Trainig on LSTM

```
 X_train = np.asarray(np.reshape(x_train, (x_train.shape[0], 14, 1)))
X_test = np.asarray(np.reshape(x_test, (x_test.shape[0], 14, 1)))
```

APPENDIX II

SCREENSHOTS

	mean	standard deviation	kurt	skewness
0	4316.593750	186.395279	-0.686817	0.439234
1	4312.748535	186.797379	-0.673855	0.459162
2	4312.160645	184.608978	-0.693921	0.443294
3	4312.748535	186.797379	-0.673855	0.459162
4	4312.160645	184.608978	-0.693921	0.443294
5	4316.593750	186.395279	-0.686817	0.439234
6	4310.731934	183.971970	-0.709318	0.420956
7	4316.189941	184.204300	-0.716226	0.414606
8	4312.748535	186.797379	-0.673855	0.459162
9	4309.229980	184.245834	-0.741166	0.445671
10	4309.229980	184.245834	-0.741166	0.445671

S1: Extracted Features from Time-Series

Training on SVM

```
[ ] from sklearn.svm import SVC
    clf = SVC()
    clf.fit(X_train1, y_train1)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

```
[ ] predicted = clf.predict(X_test1)
```

```
[ ] from sklearn import metrics
```

Performance

```
[ ] print("Accuracy = {}\nPrecision = {}\nRecall = {}\nF1 Score = {}".format(metrics.accuracy_score(y_test1, predicted), metrics.precision_score(
    score_p.append([metrics.accuracy_score(y_test1, predicted), metrics.precision_score(y_test1, predicted),metrics.recall_score(y_test1, predict
```

```
Accuracy = 0.6989319092122831
Precision = 0.7059190031152648
Recall = 0.7248880358285349
F1 Score = 0.7152777777777777
```

S2: Performance of SVM algorithm

Training on Logistic Regression

```
[ ] from sklearn.linear_model import LogisticRegression
```

```
[ ] clf_D = LogisticRegression()
    clf_D.fit(X_train1, y_train1)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
[ ] predict = clf_D.predict(X_test1)
```

Performance

```
[ ] print("Accuracy = {}\nPrecision = {}\nRecall = {}\nF1 Score = {}".format(metrics.accuracy_score(y_test1, predict), metrics.precision_score(y_
    score_p.append([metrics.accuracy_score(y_test1, predict), metrics.precision_score(y_test1, predict),metrics.recall_score(y_test1, predict),me
```

```
Accuracy = 0.5240320427236315
Precision = 0.5404607206142942
Recall = 0.5854126679462572
F1 Score = 0.5620393120393121
```

S3: Performance on Logistic Regression

```
 - 0s - loss: 0.1150 - acc: 0.9543 - val_loss: 0.5437 - val_acc: 0.8309

Epoch 00200: val_loss did not improve from 0.49123
```

```
] score = model2.evaluate(x_test, y_test, verbose=1)
  print("Accuracy: ", score[1])
```

```
3980/3980 [==============================] - 0s 77us/step
Accuracy:  0.7557788944124576
```

```
] predict3 = [1 if a>=0.5 else 0 for a in model2.predict(x_test)]
```

```
] rint("Accuracy = {}\nPrecision = {}\nRecall = {}\nF1 Score = {}".format(metrics.accuracy_score(y_test, predict3), metrics.precision_score(y_te
  core_p.append([metrics.accuracy_score(y_test, predict3), metrics.precision_score(y_test, predict3),metrics.recall_score(y_test, predict3),metr
```

```
Accuracy = 0.7557788944723618
Precision = 0.7142857142857143
Recall = 0.8642892521050025
F1 Score = 0.7821604661586732
```

```
] import matplotlib.pyplot as plt
```

S4: Performance on Neural Network

```
Epoch 00296: val_loss did not improve from 0.48148
Epoch 297/300
 - 2s - loss: 0.2446 - acc: 0.8880 - val_loss: 0.6009 - val_acc: 0.7091

Epoch 00297: val_loss did not improve from 0.48148
Epoch 298/300
 - 2s - loss: 0.2528 - acc: 0.8842 - val_loss: 0.5446 - val_acc: 0.7591

Epoch 00298: val_loss did not improve from 0.48148
Epoch 299/300
 - 2s - loss: 0.2403 - acc: 0.8913 - val_loss: 0.5794 - val_acc: 0.7373

Epoch 00299: val_loss did not improve from 0.48148
Epoch 300/300
 - 2s - loss: 0.2791 - acc: 0.8714 - val_loss: 0.5226 - val_acc: 0.7709

Epoch 00300: val_loss did not improve from 0.48148
```

```
score = model4.evaluate(X_test, y_test, verbose=1)
print("Accuracy: ", score[1])
```

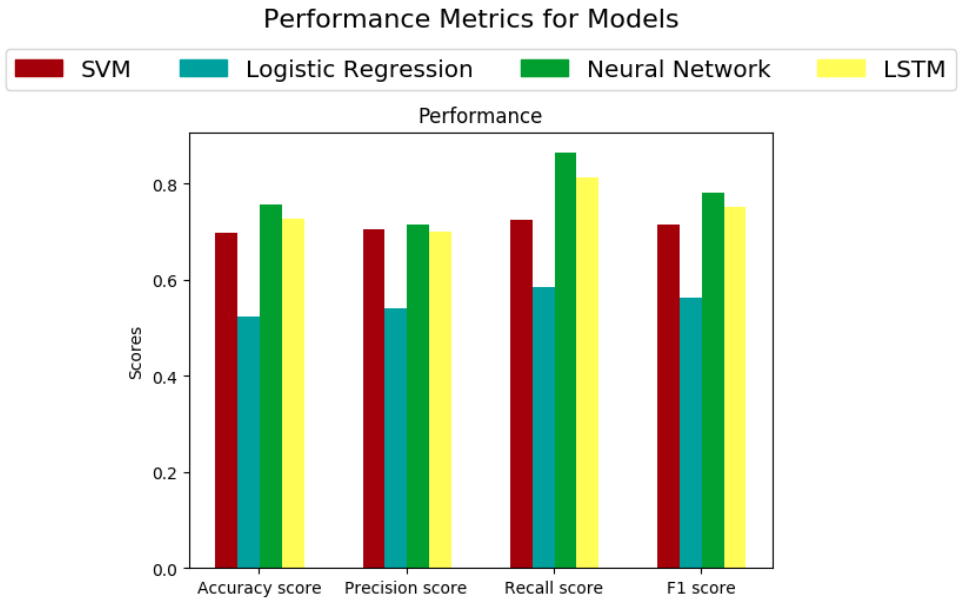
```
3980/3980 [==============================] - 1s 256us/step
Accuracy:  0.728140703577492
```

```
predict5 =  [1 if a>0.5 else 0 for a in model4.predict(X_test)]
```

```
print("Accuracy = {}\nPrecision = {}\nRecall = {}\nF1 Score = {}".format(metrics.accuracy_score(y_test, predict5), metrics.precision_score(y_
score_p.append([metrics.accuracy_score(y_test, predict5), metrics.precision_score(y_test, predict5),metrics.recall_score(y_test, predict5),me
```

```
Accuracy = 0.728140703517588
Precision = 0.6999573196756296
Recall = 0.8122833085685983
F1 Score = 0.7519486474094451
```

S5: Performance on LSTM algorithm



S6: Comparison of Algorithms

REFERENCES

[1] B.Azhagusundari, Antony Selvadoss Thanamani ”International Journal of Innovative Technology and Exploring Engineering (IJITEE)” ISSN: 2278-3075, Volume-2, Issue-2, January 2013

[2] Brijil Chambayil, Rajesh Singla ”EEG Eye Blink Classification Using Neural Network” Proceedings of the World Congress on Engineering 2010 Vol I WCE 2010, June 30 - July 2, 2010, London, U.K NEHA JAIN et al. Citation: 10.2348/ijset06150715

[3] Donald Herbert “Introduction to Generalized Linear and Nonlinear Models”

[4] Hong J. Eoh, Min K. Chung ”Electroencephalographic study of drowsiness in simulated driving with sleep deprivation” Volume 35, Issue 4, April 2005

[5] Isabelle Guyon” An Introduction to Variable and Feature Selection“Journal of Machine Learning Research (2003)

[6] Lawrence JK Wee, Tin Wee Tan “SVM-based prediction of caspase substrate cleavage sites” licensee BioMed Central Ltd, 2006

[7] Ling Li, Lei Xiao, and Long Chen “Differences of EEG between Eyes-Open and Eyes-Closed States Based on Autoregressive Method ”JOURNAL OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA, VOL. 7, NO. 2, JUNE 2009

[8] NIR FRIEDMAN “Bayesian Network Classifiers* “Machine Learning, 29, 131–163 (1997)

[9] Oliver R¨osler and David Suendermann “A First Step towards Eye State Prediction Using EEG” Wuerttemberg Cooperative State University (DHBW), Stuttgart, Germany, 2013

[10] Ori Ossmy, Ofir Tam”MINDDESKTOP: COMPUTER ACCESSIBILITY FOR SEVERELY HANDICAPPED”

[11] Payam Aghaei Pour, Tauseef Gulrez, Omar “Brain- Computer Interface: Next Generation Thought Controlled Distributed Video Game Development Platform”

[12] Razaki, R., Frasson, C. “Using machine learning to predict learner emotional state from brainwaves” Advanced Learning Technologies, 2007. ICALT 2007

[13] TingWang, Sheng-UeiGuan,” EEG Eye State Identification Using Incremental Attribute Learning with Time-Series Classification”2014

[14] YongSeog Kim, W. Nick Street, and Filippo Menczer” Feature Selection in Data Mining “, University of Iowa, USA NEHA JAIN et al. Citation: 10.2348/ijset06150715 ISSN (O): 2348-4098 ISSN (P): 2395-4752 International Journal of Science, Engineering and Technology- www.ijset.in 719