Premier University

FINAL YEAR THESIS

A comparative study of Machine Learning algorithms for Epileptic Seizure Prediction using EEG Signal

Supervisor:

Md. Neamul Haque

Lecturer

Department of Computer Science & Engineering

Premier University

Submitted By:

Joyita Das Gupta [1703310201378]

Mrettika Das Gupta [1703310201444]

Himani Das Koli [1703310201468]

A thesis submitted in Partial fulfillment of the requirements for the degree of B. Sc. Engineering

Department of Computer Science & Engineering

Premier University

November, 2022

The final year thesis paper entitled **"A comparative study of Machine Learning Algorithms for Epileptic Seizure Prediction using EEG Signals"** submitted by Joyita Das Gupta, Mrettika Das Gupta, Himani Das Koli has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science Engineering (BSc in CSE) to be awarded by Premier University.

Signed:

Prof. Dr. Taufique Sayeed
Chairman,
Department of Computer Science & Engineering
Premier University, Chittagong

Signed:

Md. Neamul Haque
Lecturer,
Department of Computer Science & Engineering
Premier University, Chittagong

Declaration of Authorship:

We Joyita Das Gupta, Mrettika Das Gupta, Himani Das Koli, declare that this thesis titled, **"A comparative study of Machine Learning Algorithms for Epileptic Seizure Prediction using EEG Signals"** and the work presented in it are our own. We confirm that:

• This work was completed while applying for an undergraduate degree at this university.

• It has been made apparent where any portion of this thesis has already been submitted for a degree or other qualification at this university or another institution.

• This is always credited clearly when we have referenced another author's published work.

• We always cite the author of any quotations we make from other people's writing. Our team worked together to create this thesis, with the exception of such citations.

Signed:

_____

Joyita Das Gupta
ID: 1703310201378
Email: joyitadasgupta1998@gmail.com

Signed:

_____

Mrettika Das Gupta
ID: 1703310201444
Email: mrettikagupta@gmail.com

Signed:

_____

Himani Das Koli
ID: 1703310201468
Email: himani.das1998@gmail.com

Acknowledgements

This thesis provided us with an excellent opportunity to study and use our expertise. We feel ourselves really lucky to have been given the opportunity to participate in it. We would like to offer our heartfelt appreciation to everyone who provided us the opportunity to complete this thesis.

A special gratitude we give to our final year thesis supervisor, **Md. Neamul Haque**, Lecturer, Department of Computer Science and Engineering, Premier University whose guideline helped us to coordinate and complete our thesis. Without his supervision, we would never be able to accomplish this task. We are always grateful to him.We would like to express our special thanks of gratitude to him who gave us the golden opportunity to do this wonderful project on the topic. She has been there providing her heartfelt support and guidance at all times and has given us invaluable guidance, inspiration, and suggestions in our quest for knowledge.

Furthermore, we would also like to acknowledge with much appreciation the crucial role of **Dr. Taufique Sayeed**, Chairman, Department of CSE, Premier University and every teacher of Department of Computer Science and Engineering for their guidance as well as constant supervision and persistent motivation.

Abstract

Epilepsy is a neurological condition that causes unprovoked, recurrent seizures. A seizure is a sudden rush of abnormal electrical activity in our brain. Doctors diagnose epilepsy when we have two or more seizures with no other identifiable cause. Electroencephalogram (EEG) technology is frequently used to monitor these brain impulses. For this paper proposes, using nine machine learning algorithm approach to upgrade the prediction epileptic seizures in EEG. In this paper, we will present and evaluate a machine learning approach to constructing patient-specific classifiers that detect the dataset of an epileptic seizure patient through analysis of the scalp EEG, a non-invasive measure of the brain's electrical activity. The aim of this study is to determine the most suitable classification algorithm to classify the epileptic seizure dataset to determine whether a person would have a seizure by applying different classification techniques and to study the behavior of the classification algorithm with respect to changes to the classification parameters. We have finding the best machine learning algorithms for predicting epileptic seizures by using different machine learning algorithms. In this dataset, performing the full of data exploration using various types of data visualization and build various ML models that can predict the patients has disease or not. At first, we considered was a dataset of epileptic seizure patients by EEG signal rate of brain activity as an input. In this dataset, each file contains a 23.6-second recording of brain activity and each data point represents the EEG recording's value at a particular time point. Our algorithms generate a prediction data list of epileptic seizure patients brain signals with the prediction result so that anyone can see the best accuracy result of each algorithm. This result can help to determine whether a patient has a condition or not. We have got the best accuracy in Support Vector Machine which is 98.2609 from nine algorithms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A significant chronic neurological illness called epilepsy can be identified by examining the brain signals that brain neurons produce. Neurons are connected to each other in a complex way to communicate with human organs and generate signals. We will present an overview of the epileptic seizure prediction method in this chapter. This chapter explains the research's background, inspiration, and goals. In this chapter, we also discuss some research that is relevant to the system we have in mind

1.1 Background

Our project's title raises the question of whether epileptic seizure prediction is necessary. Seizure prediction focuses on predicting when an epileptic patient will experience a seizure rather than evaluating the prognostic or diagnostic significance of abnormal EEG data.

The transient occurrence of symptoms known as epilepsy is brought on by the synchronization of abnormally high levels of neuronal activity in the brain. According to estimates, 65 million individuals worldwide suffer from epilepsy. Neurologists still have to spend a lot of time reviewing continuous electroencephalograms (EEGs) to keep tabs on epileptic patients. Automatic EEG signal classification is not easy and still challenging. There is no apparent difference in EEG signal between non-epileptic seizures in people that close eyes to epileptic seizure patient. It is also difficult to recognize EEG signal between epileptic seizures to non-epileptic seizures in patient with the region of the tumor. Many researchers have published their study on automatic epileptic seizure, but most of which only used two classes, epileptic seizure EEG signal and non-epileptic seizure EEG signal. In machine learning domain, it is more difficult to classify five classes than two classes. If we can classify correctly the EEG data in five classes that was annotated in the dataset, it is very helpful for neurologist and medical practice to diagnose their patients.

The brain is the most intricate organ in the body and the hub of the neurological system. The biological components of the brain are called neurons (nerve cells). Over 100 billion neurons are linked among themselves in the average human brain. Both the executive functions—such as reasoning, thought, and planning—and the regulation of the body are performed by the brain. It's interesting because it's the bodily part with the most intricate organ. The entire mechanism of the brain's operation is still a mystery to science. The two methods that have been utilized to research the brain's activity are electrophysics and hemodynamic response. The electrical characteristics of biological cells and tissues are investigated using electrophysiological techniques. It entails measuring electric current or voltages from a single ion channel to a whole organ, such the brain. The electroencephalogram and magnetoencephalogram are two electrophysiological techniques that are frequently used in neuroscience and neuroengineering to assess the electrical activity of neurons. Typically, electrodes or sensors are positioned on biological tissue to perform electrophysiological measurements. Blood oxygen levels in the brain, for example, are functional activities that also offer very valuable and significant information into brain functions.

One of the most dangerous elements of the condition epilepsy is the abrupt and seemingly random nature of seizures. Most epileptic patients only experience seizures occasionally and don't exhibit any symptoms of their condition throughout the intervals between seizures, or inter-ictal intervals. But the continual worry about the next seizure and the sense of helplessness that goes along with it frequently have a significant influence on a patient's day-to-day existence.

1.2 Motivation

Since the manual detection of electrographic seizures in continuous electroencephalogram (EEG) monitoring is very time-consuming and requires a trained expert, attempts to develop automatic seizure detection are diverse and ongoing. Electroencephalogram signals are recordings of the electrical activity of brain, which are used extensively in many medical applications, including detection of epileptic seizures. Traditionally, neurologists made inferences by visual inspection. However, this was usually very time consuming and the results are subject to the expertise of the reader. Hence, automatic epileptic seizure detection techniques are needed, which are able to provide high quality results in a short time.

The prediction of epileptic seizures ensures enough time before it actually occurs; it is very useful because the attack can be avoided by the drug. Epileptic seizures have four different states: the preictal state, which is a state that appears before the seizure begins, the ictal state that begins with the onset of the seizure and ends with an attack, the postictal state that starts after ictal state, and interictal state that starts after the postictal state of 1st seizure and ends before the start of preictal state of consecutive seizure. After critical considerations of these states, we have proposed a model to detect the start of the preictal state. However, EEG data acquisition by placing electrodes on the scalp of the patient is out of the scope of our research.

By finding out the best accuracy, we were able to predict whether there is a disease or not in a very short time using machine learning algorithms. Early prediction helps patients, as medication can be done by the doctors to prevent the seizure. Due to this medication, the patient can now perform his or her routine activities without any interference from seizures.

Machine learning approaches are intensely being applied to this problem due to their ability to classify seizure conditions from a large amount of data, and provide pre-screened results for neurologists. Several features, data transformations, and classifiers have been explored to analyze and classify seizures via EEG signals. The primary goal of the epilepsy treatment is to prevent the seizure activity as early as possible with as few side effects as possible. Available anti-seizure medications are used by doctors and clinicians in treating epilepsy patients. Often, the epilepsy patients have to undergo surgery in special cases where medications do not work effectively. For surgical procedure, EEG has been used to localize the seizure focus region to identify the brain region candidate for surgery. In recent days, much focus has been put into developing implementable medical devices for offering an alternate form of treatment to ensure better life for the epilepsy patients.

1.3 Previous work

To predict epileptic seizures using EEG signals, many machine learning methods have been applied. This work is related to enhanced Detection of Epileptic Seizure Using EEG Signals in Combination with Machine Learning Classifiers focused by Wail Mardini Muneer, Bani Yassein, Rana Alrawashdeh [1].They developed an epileptic Seizures Detection from EEG Recordings Based on a Hybrid system of Gaussian Mixture Model and Random Forest Classifier which was finished by Garineh Sarkies Ohannesian, Esraa Jasim Harfash[2]. Khaled Mohamad Almustafa used Classification of epileptic seizure dataset using different machine learning algorithms [3]. The work is presented that epileptic Seizure Detection in EEGs by Using Random Tree Forest, Naïve Bayes and KNN Classification by Fauzia Lestari, Rizki Edmi Edison [4]. An approach based on seizure

detection in EEG signals using support vector machines by Cher Hau SengR. Demirli Lunal Khuon Donovan Bolger[5]. Patient Specific Epileptic Seizures Prediction based on Support Vector Machine by Abdalla Gabara, RetajYousri, Darine Hamdy, Michael H. Zakhari, Hassan Mostafa[6].Ahmet Alkan, Etem Koklukaya, Abdulhamit Subasihave focused on Automatic seizure detection in EEG using logistic regression and artificial neural network [7].Sarthak Gupta, Siddhant Bagga, Vikas Maheshkar& M.P.S. Bhatia have researched on Detection of Epileptic Seizures using EEG Signals[8]. Epileptic seizure detection: a comparative study between deep and traditional machine learning techniques completed by Rekha Sahu, Satya Ranjan Dash, Lleuvelyn Areglado Cacha, R. R. Poznansky[9]. Dwi Sunaryono, Riyanarto Sarno, Joko Siswantoro have proposed to gradient boosting machines fusion for automatic epilepsy detection from EEG signals based on wavelet features[10]. H. Rajaguru, S. Prabhakar had a study on analysis of adaboost classifier from compressed EEG features for epilepsy detection[11]. A paper had been reviewed titled application of Machine Learning To Epileptic SeizureDetection by Ali Shoeb, John Guttag[12]. A review of epileptic seizure detection using machine learning classifiers presented an approach by Mohammad Khubeb Siddiqui1, Ruben Morales-Menendez1, Xiaodi Huang2 and Nasir Hussain3[13]. A paper is done by Poomipat Boonyakitanont1, Apiwat Lek-uthai1, Krisnachai Chomtho2, and Jitkomut Songsiri1which is A review of feature extraction and performance evaluation in epileptic seizure detection using EEG[14]. Comparison Machine Learning Algorithms for Recognition of Epileptic Seizures in EEG which is done by Bekir Karlık1 and Şengül Bayrak Hayta2 [15].

1.4 Our Objectives

For our proposed system, we have collected data from UCI Machine Learning Repository.

1. To predict epileptic seizure of a patient by using EEG signals time recording.
2. To build various ML models that can predict whether patients has disease or not.
3. To perform dataset exploration using various types of data visualization.
4. To predict the accuracy among various algorithms.
5. To find the best algorithm for epileptic seizure prediction.

After preprocessing, we used nine different techniques K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Logistic Regression, Gaussian Naive Bayes, Decision Tree, Random Forest, Extra Tree Classifier, Gradient Boosting, AdaBoost to predict epileptic seizure.

1.5 Organization of Report:

This report is divided into different chapters. A brief description of related work in this field is mentioned in **Chapter 2**. Then we discussed about some preliminary knowledge that is required for the classification approach. In **Chapter3**, Methodology of our system is illustrated in **Chapter 4**. Experiment and evaluation of the proposed systems is described in **Chapter 5**. **Chapter 6** contains future work and conclusion of our system.

Chapter 2

Literature Review

Numerous studies have been done over a long period of time to determine how to predict epileptic seizures using EEG data, and the majority of these studies are based on English literature.

2.1 Epileptic Seizure Using EEG Signals

Wail Mardini Muneer, Bani Yassein, Rana Alrawashdeh are proposed for detecting epileptic seizures from EEG signals recorded from normal and epileptic patients[1]. The suggested approach is designed to classify the abnormal signal from the normal one automatically. They improved the accuracy of epileptic seizure detection and reduce computational costs. To address this, the proposed framework uses the 54-DWT mother wavelets analysis of EEG signals using the Genetic algorithm in combination with other four machine learning classifiers: Support Vector Machine, K-Nearest Neighbors, Artificial Neural Network and Naive Bayes. The performance of 14 different combinations of two-class epilepsy detection is investigated using these four ML classifiers. The experimental results showed that the four classifiers produce comparable results for the derived statistical features from the 54-DWT mother wavelets, however, the ANN classifier achieved the best accuracy in most datasets combinations, and it outperformed the other examined classifiers.

2.2 Based on a hybrid system of Gaussian Mixture Model and Random Forest Classifier

Garineh Sarkies Ohannesian, Esraa Jasim Harfash are developed an epileptic Seizures Detection from EEG Recordings Based on a Hybrid system of Gaussian Mixture Model and Random Forest Classifier [2]. This paper aims to increase epileptic seizure detection accuracy in a balanced dataset while reducing the execution time. They suggested using a hybrid system of supervised and unsupervised machine learning methods to build a computationally effective and scalable model for the early detection of epileptic seizures using two-class EEG datasets. The EEG signal was first divided into frequency sub-bands using the Discrete Wavelet Transform algorithm. The Gaussian Mixture Model which divides data into two clusters based on whether they are associated with epilepsy or not, is then given the EEG features that were retrieved. And finally, the random forest classifier was used to assess the output of the clusters. The experimental results show that the highest accuracy was achieved by the hybrid system of GMM with random forest with DWT features with an accuracy of 93.62 %.

2.3 Classification of epileptic seizure dataset using different machine learning algorithms

In this paper Khaled Mohamad Almustafa has focused on Classification of epileptic seizure dataset using different machine learning algorithms [3]. He classified the sample for epileptic seizures using a variety of classifiers. With 97.08% Accuracy, ROC = 0.996, and RMSE = 0.1527, it was demonstrated that the Random Forest classifier outperformed the K-Nearest Neighbor (K-NN), Naive Bayes, Logistic Regression, Decision Tree (D.T.), Random Tree, J48, and Stochastic Gradient Descent (S.G.D.) classifiers. The performance of the classifier to categorize the dataset for epileptic seizures was examined in relation to specific changes in its parameters using sensitivity analysis for some of these classifiers. The dataset was then predicted using a feature selection method based on attribute variance.

2.4 Epileptic Seizure Detection in EEGs by Using three Classifications

The work is presented that epileptic Seizure Detection in EEGs by Using Random Tree Forest, Naïve Bayes and KNN Classification by Fauzia Lestari, Mohammad Haekal, Mohammad Haekal[4].In this study, they compared the naive bayes, random tree forest and K-nearest neighbor (KNN) classification algorithms to detect epilepsy. The raw EEG data were pre-processed before doing feature extraction. Comparing those three classifiers, they calculated accuracy, sensitivity, specificity, and precision. In comparison to random tree forest (accuracy: 86.6%) and naive bayes classifier (accuracy: 55.6%), KNN classifier has the highest accuracy (92.7%). Naive Bayes classification, which has an 80.3% sensitivity rating, outperforms KNN (73.2%) and random tree forest (42.2%), in terms of sensitivity. KNN classification provides 96.7% specificity, followed by random tree forest at 95.9% and naive bayes at 50.4% for specificity. Therefore, KNN Classification gives better performance than naïve bayes and random tree forest classification.

2.5 Seizure Detection in EEG signals using Support Vector Machines

An approach based on seizure detection in EEG signals using support vector machines by Cher Hau SengR. Demirli Lunal Khuon Donovan Bolger[5]. They have worked with a linear Support Vector Machine (SVM) classifier is designed to detect and classify seizures in EEG signals based on a few simple features such as mean, variance, dominant frequency, and the mean power spectrum. A benchmark EEG database is used to test the SVM classifier. Classification rates of up to 98% were attained by combining these features. The suggested classifier, which makes use of a few basic features, is computationally effective enough to be used in a system for real-time seizure monitoring.

2.6 Patient Specific Epileptic Seizures Prediction based on Support Vector Machine

The work that is patient specific epileptic seizures prediction based on Support Vector Machine by Abdalla Gabara, RetajYousri, Darine Hamdy, Michael H. Zakhari, Hassan Mostafa [6]. The aim of this work is to build practical hardware-implementable Machine Learning classifiers capable of predicting the seizure onsets prior to their occurrences with high sensitivity and accuracy. The classification method proposed involves removing certain channels for each patient, extracting the features from the EEG signal, selecting the best feature combination for each patient, and finally training the selected SVM classifier accordingly. Evaluating the performance of the proposed classification technique yields promising results for the selected patients with accuracies exceeding 95%.

2.7 Automatic seizure detection in EEG

In another paper Ahmet Alkan, Etem Koklukaya, Abdul Hamit Subasi have focused on Automatic seizure detection in EEG using logistic regression and artificial neural network [7]. In this study, they have used multiple signal classification and the EEG power spectra were used as an input to a classifier. The classic statistical methodology based on logistic regression and the developing, computationally intensive techniques based on artificial neural networks were offered as two fundamentally distinct approaches for creating classification models classifiers. Classifiers based on multilayer perceptron neural networks and LR were created, and their classification accuracy for EEG signals was compared. In this classification the MLPNN-based classifier fared better than its LR-based rival. The MLPNN-based classifier performed better than the LR-based classifier within the same group.

2.8 Detection of Epileptic Seizures using EEG Signals

Sarthak Gupta, Siddhant Bagga, Vikas Maheshkar & M.P.S. Bhatia have researched on Detection of Epileptic Seizures using EEG Signals[8]. They have proposed that uses Discrete Wavelet Transform to convert the EEG signal into the time-frequency domain. Approximation and detail coefficients were obtained after the discrete wavelet transform of EEG signals. Then, various features are extracted and then classification is carried out on a number of classifiers including convolutional neural networks, random forests etc for the detection of epilepsy seizure. Results show that our processing technique and the combination of features extracted provide far superior results than those obtained by applying the classifiers on the EEG signals directly. In our work, an accuracy of 99.29 was achieved which outperformed the conventional epileptic seizure detection techniques.

2.9 Epileptic Seizure Detection: A comparative study between deep and traditional machine learning techniques

This paper is about epileptic seizure detection: a comparative study between deep and traditional machine learning techniques completed by Rekha Sahu, Satya Ranjan Dash, Lleuvelyn Areglado Cacha, R.R.Poznansky [9]. This work is to categorize electroencephalography signals on different channels' recordings for classifying and predicting epileptic seizures. The collection of the electroencephalography recordings contained in the dataset attributes 179 information and 11,500 instances. Instances are of five categories, where one is the symptoms of epilepsy seizure. They have used traditional, ensemble methods and deep machine learning techniques highlighting their performance for the epilepsy seizure detection task. One dimensional convolutional neural network, ensemble machine learning techniques like bagging, boosting (AdaBoost, gradient boosting, and XG boosting), and stacking is implemented. Traditional machine learning techniques such as decision tree, random forest, extra tree, ridge classifier, logistic regression, K-Nearest Neighbor, Naive Bayes (gaussian), and Kernel Support Vector Machine (polynomial, gaussian) are used for classifying and predicting epilepsy seizure. After sorting and comparing algorithms, they have found the convolutional neural network and extra tree bagging classifiers to have better performance than all other ensemble and traditional classifiers.

2.10 Gradient Boosting Machines fusion for Automatic Epilepsy Detection

Dwi Sunaryono, Riyanarto Sarno, Joko Siswantoro have proposed to gradient boosting machines fusion for automatic epilepsy detection from EEG signals based on wavelet features[10]. This study is to propose a classification method for automatic epilepsy detection from EEG signals. The original EEG signals were processed using discrete Fourier transform (DFT) and discrete wavelet transform (DWT) prior to feature extraction. The proposed method has been evaluated using three classes EEG signals (normal-interictal-ictal) included in EEG dataset from University of Bonn. The experimental result shows that the proposed GBMs fusion can improve the performance of a single GBM in classifying EEG signals. Furthermore, the proposed GBMs fusion can perfectly detect epilepsy from EEG signals with an accuracy of 100%.

2.11 Analysis of Adaboost Classifier from compressed EEG features for Epilepsy Detection

H. Rajaguru ,S. Prabhakar had a study on analysis of adaboost classifier from compressed EEG features for epilepsy detection[11]. Here they have presented the concept of code converters is implemented and as the classification results through it are not satisfactory it is further optimized with the help of Adaboost Classifier. Results show that when Adaboost Classifier is utilized as a post

classifier, an average perfect classification rate of about 94.58%, an average classification accuracy of about 97.29%, an average performance index of about 94.51 % and an average quality value of 21.82 is obtained.

2.12 Application of Machine Learning to Epileptic Seizure Detection

A paper have been reviewed titled application of Machine Learning To Epileptic Seizure Detection by Ali Shoeb, John Guttag[12]. They have presented and evaluated a machine learning approach to constructing patient-specific classifiers that detect the onset of an epileptic seizure through analysis of the scalp EEG, a non-invasive measure of the brain's electrical activity. When trained on 2 or more seizures per patient and tested on 916 hours of continuous EEG from 24 pa- tients, our algorithm detected 96% of 173 test seizures with a median detection delay of 3 seconds and a median false detection rate of 2 false detections per 24 hour period.

2.13 A review of Epileptic Seizure Detection using Machine Learning Classifiers

A review of epileptic seizure detection using machine learning classifiers presented an approach by Mohammad Khubeb Siddiqui1, Ruben Morales-Menendez, Xiaodi Huang and Nasir Hussain[13]. This paper is to present an overview of the wide varieties of these techniques over the last few years based on the taxonomy of statistical features and machine learning classifiers 'black-box' and 'non-black-box'. The presented state-of-the-art methods and ideas will give a detailed understanding about seizure detection and classification, and research directions in the future.

2.14 A review of feature extraction and performance evaluation in Epileptic Seizure Detection using EEG

The paper is done by Poomipat Boonyakitanont1, Apiwat Lek-uthai1, Krisnachai Chomtho2, and Jitkomut Songsiri1 which is A review of feature extraction and performance evaluation in epileptic seizure detection using EEG[14]. They have done that comprehensively summarize feature descriptions and their interpretations in characterizing epileptic seizures using EEG signals, as well as to review classification performance metrics. A redundancy analysis using a correlation-based feature selection was applied. The results showed that the following features - variance, energy, nonlinear energy, and Shannon entropy computed on a raw EEG signal, as well as variance, energy, kurtosis, and line length calculated on wavelet coefficients were able to significantly capture the seizures. An improvement of 4.77-13.51% in the Bayesian error from the baseline was obtained.

2.15 Comparison Machine Learning Algorithms for Recognition of Epileptic Seizures in EEG

Comparison Machine Learning Algorithms for Recognition of Epileptic Seizures in EEG which is done by Bekir Karlık and Şengül Bayrak Hayta[15]. In this study is to diagnose epileptic seizures by using different machine learning algorithms. For this, the discrete wavelet transform (DWT) and parametric approaches based on the autoregressive (AR) model are used to extract the frequency components of the EEG. The results show that k-NN, ANN and SVM were the most efficient method according to test processing of both DWT and AR as feature extraction for recognition of epileptic seizures in EEG.

Chapter 3

Preliminaries

This chapter describes some basic definitions and concepts that are prerequisites for the proposed system.

3.1 K-Nearest Neighbor (KNN)

3.1.1 Theoretical Background

The abbreviation KNN stands for "K-Nearest Neighbor". It is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements. KNN aims for pattern recognition tasks. The number of nearest neighbors to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'. It is based on the idea that the observations closest to a given data point are the most "similar" observations in a data set, and we can therefore classify unforeseen points based on the values of the closest existing points. By choosing K, the user can select the number of nearby observations to use in the algorithm. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. For classification, a majority vote is used to determine which class a new observation should fall into. Larger values of K are often more robust to outliers and produce more stable decision boundaries than very small values. If the value of k is 5 it will look for 5 nearest Neighbors to that data point. In this example, if we assume k=4. KNN finds out about the 4 nearest Neighbors. All the data points near black data points belong to the green class meaning all the neighbors belong to the green class so according to the KNN algorithm, it will belong to this class only. The red class is not considered because red class data points are nowhere close to the black data point. We usually use Euclidean distance to calculate the nearest neighbor. If we have two points (x, y) and (a, b). The formula for Euclidean distance (d) will be

$$d = \text{sqrt}((x-a)^2+(y-b)^2)$$

We try to get the smallest Euclidean distance and based on the number of smaller distances we perform our calculation.

Assume there are two categories, Category A and Category B, and we receive a new data point x1. Which of these categories will this data point fall into? A K-NN algorithm is required to address this type of problem. We can simply determine the category or class of a dataset with the help of K-NN. Consider the diagram below:

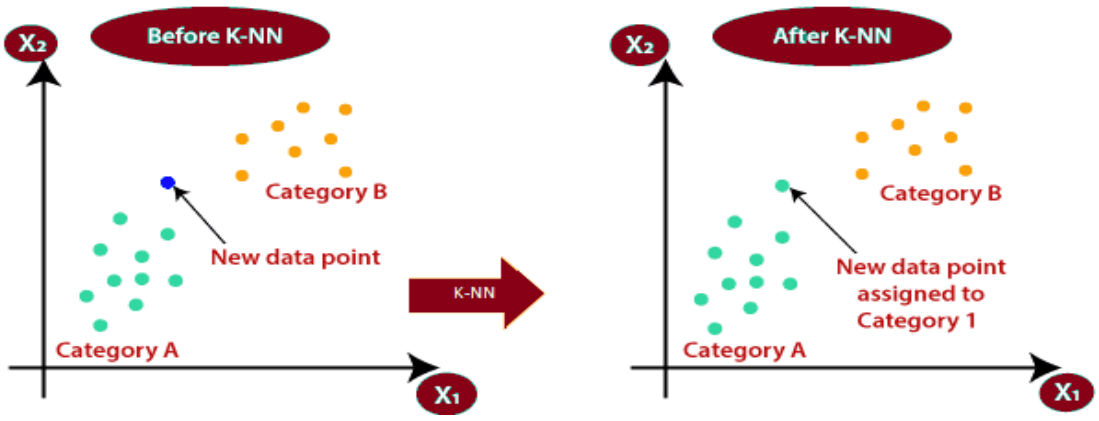

Fig. 3.1: Identify the class of dataset by using KNN

3.1.2 Properties

K-NN is an example of a uniform kernel, variable-bandwidth, kernel density "balloon" estimator. By calculating the distances between each stored example and the test example, the naive form of the technique is simple to use but computationally demanding for large training sets. Even for big data sets, the K-NN algorithm can be computationally inefficient by using an approximate nearest neighbor search method. Over the years, a variety of nearest neighbor search algorithms have been put forth; these typically aim to minimize the amount of real distance evaluations.
K-NN has some strong consistency results. As the amount of data approaches infinity, the two-class K-NN algorithm is guaranteed to yield an error rate no worse than twice the Bayes error rate (the minimum achievable error rate given the distribution of the data). Various improvements to the K-NN speed are possible by using proximity graphs.

For multi-class K-NN classification, Cover and Hart (1967) prove an upper bound error rate of $R* \leq$ RkNN

$$\leq R^* \left(2 - \frac{MR^*}{M-1}\right)$$

3.2 Support Vector Machine

3.2.1 Theoretical Background

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. A support vector machine is a machine learning model that is able to generalize between two different classes if the set of labelled data is provided in the training set to the algorithm. The main function of the SVM is to check for that hyperplane that is able to distinguish between the two classes. There can be many hyperplanes that can do this task but the objective is to find that hyperplane that has the highest margin that means maximum distances between the two classes, so that in future if a new data point comes that is two be classified then it can be classified easily. SVM chooses the extreme points/vectors that help in creating the hyperplane with maximum margin, i.e the maximum distance between data points of both classes. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:

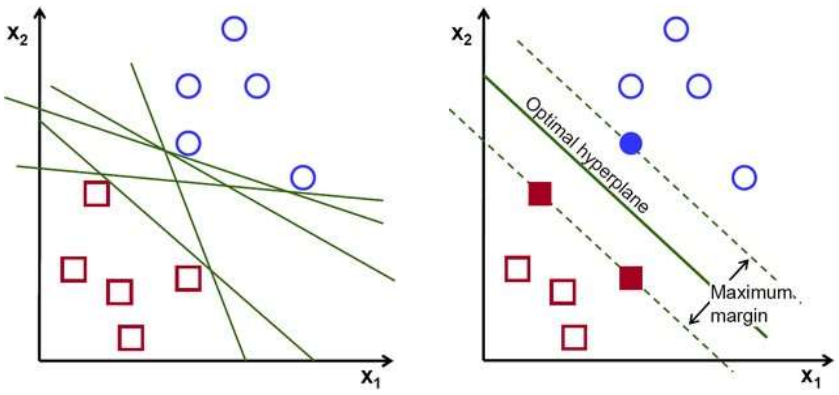

Fig. 3.2: Possible hyperplanes

The SVM algorithms are used to classify data in a 2-dimensional plane as well as a multidimensional hyperplane. The multidimensional hyperplane uses the "Kernels" to categorize the multidimensional data. SVM algorithm is implemented with kernel that transforms an input data space into the required form. SVM uses a technique called the kernel trick in which kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, kernel converts non-separable problems into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible and accurate. The following are some of the types of kernels used by SVM.

- Linear Kernel: It can be used as a dot product between any two observations. The formula of linear kernel is as below

K(x,xi)=sum(x∗xi)K(x,xi)=sum(x∗xi)

From the above formula, we can see that the product between two vectors say $x\&xi$ is the sum of the multiplication of each pair of input values.

- Polynomial Kernel:

It is more generalized form of linear kernel and distinguish curved or nonlinear input space. Following is the formula for polynomial kernel –

k(X,Xi)=1+sum(X∗Xi)^dk(X,Xi)=1+sum(X∗Xi)^d

Here d is the degree of polynomial, which we need to specify manually in the learning algorithm.

- Radial Basis Function (RBF) Kernel:

RBF kernel, mostly used in SVM classification, maps input space in indefinite dimensional space. Following formula explains it mathematically –

K(x,xi)=exp(−gamma∗sum(x−xi^2))K(x,xi)=exp(−gamma∗sum(x−xi^2))

Here, gamma ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of gamma is 0.1.

The aim of using SVM is to correctly classify unseen data. SVMs have a number of applications in several fields. Some common applications of SVM are-Face detection, Text and hypertext categorization, Classification of images, Bioinformatics, Protein fold and remote homology detection, Handwriting recognition, Generalized predictive control (GPC).

3.4 Logistic Regression

3.4.1 Theoretical Background

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. For example, a logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will be admitted or not to a particular college. These binary outcomes allow straightforward decisions between two alternatives.

Logistic regression has become an important tool in the discipline of machine learning. It allows algorithms used in machine learning applications to classify incoming data based on historical data. As additional relevant data comes in, the algorithms get better at predicting classifications within

data sets. It can also play a role in data preparation activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load (ETL) process in order to stage the information for analysis.
That model can take into consideration several input criteria. In the event of college acceptance, the logistic function can take into account the student's grade point average, SAT score, and number of extracurricular activities. Using historical data on previous outcomes using the same input criteria, it then assigns new instances a rating depending on how likely it is that they will fall into one of two outcome groups. Logistic regression is a branch of machine learning that has gained importance. It enables machine learning algorithms to categorize fresh input in light of previous information. The algorithms become better at guessing classes within data sets as more pertinent data is made available.

3.3.2 Model

The logistic function is of the form:

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}} \quad \ldots\ldots\ldots (1)$$

where is a location parameter (the midpoint of the curve, where displaystyle

p(mu)=1/2p(mu)=1/2) and s is a scale parameter. This expression may be rewritten as:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 - \beta_1 x)}} \quad \ldots\ldots\ldots (2)$$

With only one explanatory variable and a binary categorical variable that can take on one of two categorical values, this straightforward model serves as an illustration of binary logistic regression. The expansion of binary logistic regression to accommodate any number of categories and explanatory variables is known as multinomial logistic regression. The model builds a regression model to predict the probability that a given data entry belongs to the category numbered as “1”. Just like linear regression assumes that the data follows a linear function, Logistic regression models the data using the sigmoid function.

3.4 Gaussian Naive Bayes

3.4.1 Theoretical Background

Naive Bayes Classifiers are based on the Bayes Theorem, which one assumption taken is the strong independence assumptions between the features. These classifiers assume that the value of a particular feature is independent of the value of any other feature. In a supervised learning situation, Naive Bayes Classifiers are trained very efficiently. Naive Bayes classifiers need a small training data to estimate the parameters needed for classification. Naive Bayes Classifiers have simple design and implementation and they can apply to many real life situations.

Based on the Bayes theorem, the probabilistic machine learning technique known as Naive Bayes is employed for numerous classification applications. The generalization of naive Bayes is called gaussian naive bayes. The Gaussian or normal distribution is the most straightforward to implement among the several functions used to estimate data distribution, as only need to figure out the mean and standard deviation for the training data.

Gaussian a probabilistic classification approach called Naive Bayes is based on using the Bayes theorem under the premise that variables are highly independent. Independence in the categorization context refers to the notion that the existence of one value of a feature does not affect the existence of another. When we say something is naive, we mean that we think that an object's features are unrelated to one another. The naive Bayes classifier is well recognized for being very expressive, scalable, and reasonably accurate in the context of machine learning, however as the training set grows, so does its performance quickly. Naive Bayes classifiers work well because of a variety of characteristics.

This classifier, or simply Nave Bayes, is a delightfully straightforward method that frequently produces extremely accurate and stable models with relatively small sample sets. Naive Bayes frequently performs so well because it simplifies predictive modeling issues to avoid the dimensionality curse. The fundamental tenet of Naive Bayes is that all independent variable features are conditionally independent. Conditional independence guarantees that the effects of one feature on an outcome do not interact in any manner with the effects of that same outcome of another characteristic. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:

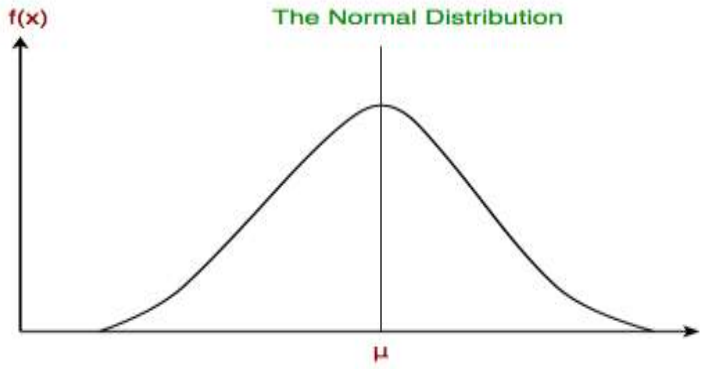

Fig. 3.3: Mean of the feature values

We have seen that the X's are in categories but how to compute probabilities when X is a continuous variable? If we assume that X follows a particular distribution, you can use the probability density function of that distribution to calculate the probability of likelihoods. If we assume that X's follow a Gaussian or normal distribution, we must substitute the probability density of the normal distribution and name it Gaussian Naïve Bayes. To compute this formula, you need the mean and variance of X.

$$P(X|Y=c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{-(x-\mu_c)^2}{2\sigma_c^2}} \quad \ldots\ldots\ldots (1)$$

In the above formulae, sigma and mu is the variance and mean of the continuous variable X computed for a given class c of Y.

Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian distribution. An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.

3.5 Decision Tree

3.5.1 Theoretical Background

Non-parametric supervised learning techniques called decision trees are employed for classification and regression. By learning straightforward decision rules derived from the data attributes, the objective is to develop a model that predicts the value of a target variable. An approximate nonlinear constant can be thought of as a tree. Decision trees are a common tool in machine learning as well as operations research, notably in decision analysis to help find a method most likely to achieve a goal.
In decision analysis, a decision tree and the closely related influence diagram are used as a visual and analytical decision support tool, where the expected values (or expected utility) of competing alternatives are calculated.

A decision tree consists of three types of nodes:

- Decision nodes – typically represented by squares.
- Chance nodes – typically represented by circles.
- End nodes – typically represented by triangles

Operations management and operations research both frequently use decision trees. A probability model should be used as a best choice model or online selection model algorithm if, in actuality, decisions must be made online with no recollection and little knowledge. Decision trees can also be used to calculate conditional probabilities in a descriptive manner.

The accuracy of the decision tree can change based on the depth of the decision tree. In many cases, the tree's leaves are pure nodes. When a node is pure, it means that all the data in that node belongs to a single class. For example, if the classes in the data set are Cancer and Non-Cancer a leaf node would be considered pure when all the sample data in a leaf node is part of only one class, either cancer or non-cancer. It is important to note that a deeper tree is not always better when optimizing the decision tree. A deeper tree can influence the runtime in a negative way. If a certain classification algorithm is being used, then a deeper tree could mean the runtime of this classification algorithm is significantly slower. There is also the possibility that the actual algorithm building the decision tree will get significantly slower as the tree gets deeper. If the tree-building algorithm being used splits pure nodes, then a decrease in the overall accuracy of the tree classifier could be experienced. Occasionally, going deeper in the tree can cause an accuracy decrease in general, so it is very important to test modifying the depth of the decision tree and selecting the depth that produces the best results.

3.6 Random Forest

3.6.1 Theoretical Background

Random Forest it can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

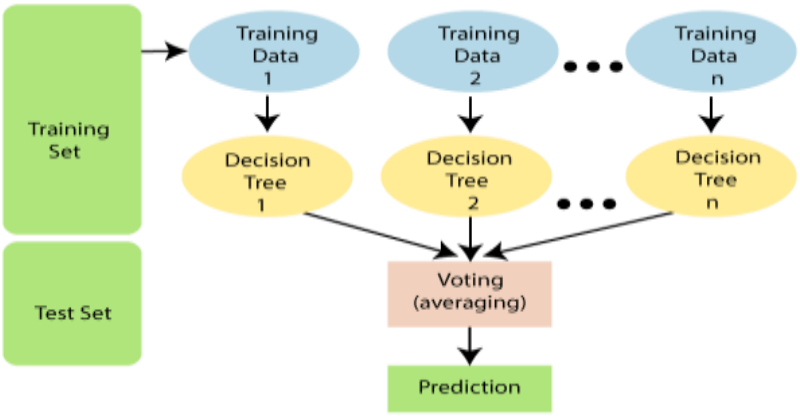

Fig. 3.4: Performance of model

The first step in measuring the variable importance in a data set $D_n=\{(X_i,Y_i)\}^n_{i=1}$DD is to fit a random forest to the data. During the fitting process the out-of-bag error for each data point is recorded and averaged over the forest (errors on an independent test set can be substituted if bagging is not used during training).

To measure the importance of the j-th feature after training, the values of the j-th feature are permuted among the training data and the out-of-bag error is again computed on this perturbed data set. The importance score for the j-th feature is computed by averaging the difference in out-of-bag error before and after the permutation over all trees. The score is normalized by the standard deviation of these differences.

Features which produce large values for this score are ranked as more important than features which produce small values. The statistical definition of the variable importance measure was given and analyzed by Zhu et al. This method of determining variable importance has some drawbacks. For data including categorical variables with different number of levels, random forests are biased in favor of those attributes with more levels. Methods such as partial permutations and growing unbiased trees can be used to solve the problem. If the data contain groups of correlated features of similar relevance for the output, then smaller groups are favored over larger groups.

3.6.2 Bagging

Bagging is a type of ensemble_machine learning approach that combines the outputs from many learner to improve performance. These algorithms function by breaking down the training set into subsets and running them through various machine-learning models, after which combining their predictions when they return together to generate an overall prediction for each instance in the original data. In machine learning, bagging is frequently used for classification tasks, especially when decision trees or artificial neural networks are used as part of a boosting ensemble. Support vector machines, decision stumps, artificial neural networks (including multi-layer perceptrons), and maximum entropy classifiers are just a few of the machine-learning algorithms to which it has been applied. The effectiveness of bagging for regression issues has been proven to be less than for classification issues. It can be also called bootstrap aggregation, which is used in ensemble learning methods to increase the accuracy and the performance of ensemble learning methods. How bagging is increasing the performance is by reducing the variance in a dataset. Bagging is commonly used in decision tree algorithms. It will be used in both classification and regression methods as it helps to reduce the overfitting problem.

.

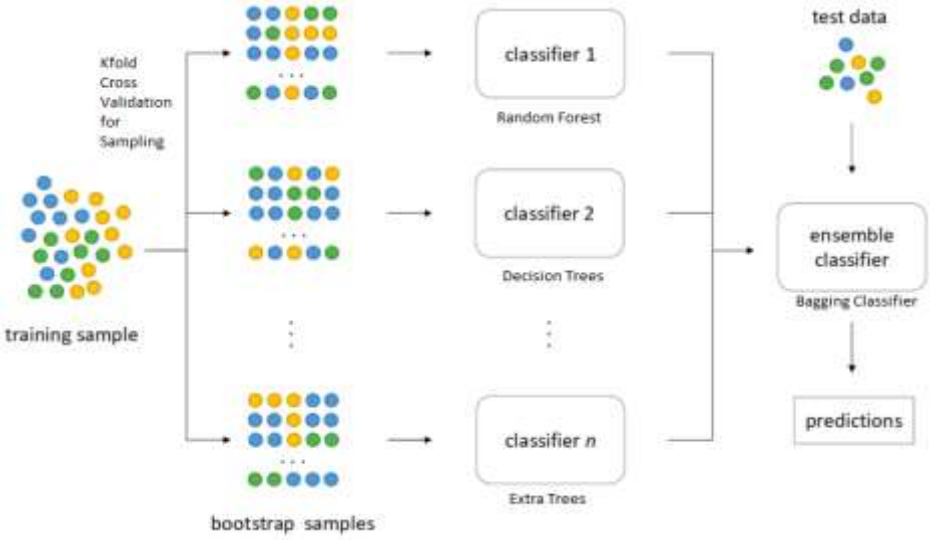

Fig. 3.5: performance of bagging

The generic method of bootstrap aggregating, often known as bagging, is used by the training algorithm for random forests to apply to tree learners. Bagging repeatedly (B times) chooses a random sample with replacement of the training set and fits trees to these samples given a training set X = x1,...,xn and responses Y = y1,..., yn:

For b = 1,..., B, sample n training instances from X and Y with replacement; refer to these as Xb and Yb. Train a regression or classification tree using the Xb, Yb data.

By averaging the forecasts from each separate regression tree on x' after training, predictions for unobserved samples x' can be made:

$$f^{`} = \_1 \sum B\, fb(x') \ldots \ldots \ldots (1)$$

$$B\ b{=}1$$

or by taking the majority vote in the case of classification trees. This bootstrapping procedure leads to better model performance because it decreases the variance of the model, without increasing the bias. This means that while the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated. Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.

Additionally, an estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}} \quad \ldots \ldots \ldots (2)$$

The number of samples/trees, B, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees B can be found using cross-validation, or by observing the out-of-bag error: the mean prediction error on each training sample xi, using only the trees that did not have xi in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

3.7 Extra Tree Classifier

3.7.1 Theoretical Background

Extra Tree Classifier is a form of ensemble learning method that combines the outcomes of various de-correlated decision trees, much like Random Forest Classifier. Classifier and only differs from it in how the forest's decision trees are built.
The Train Using AutoML tool employs the ensemble supervised machine learning technique known as extra trees (short for excessively randomized trees), which uses decision trees. A huge number of unpruned decision trees are generated from the training dataset as part of the Extra Trees algorithm's operation. In the case of regression, predictions are made by averaging the predictions of the decision trees; in the case of classification, predictions are made via majority voting.

- Regression: Forecasts derived from the average of predictions using decision trees.
- Classification: Predictions derived from decision trees that were classified by democratic majority.

From the initial training sample, the decision tree in the Extra Trees Forest is built. The decision tree must select the optimal feature to split up the data according to some mathematical criteria from a random sampling of k features from the feature-set at each test node (typically the Gini Index). Multiple de-correlated decision trees are created as a result of this random sampling of features.

Although Extra Trees generates numerous trees and splits nodes using random subsets of features, it differs from Random Forest in two important ways: it does not bootstrap observations, and nodes are split on random splits rather than the optimal splits.

So, in summary, ExtraTrees:

- builds multiple trees with **bootstrap** = **False** by default, which means it samples without replacement
- nodes are split based on **random** splits among a **random subset** of the features selected at every node

In Extra Trees, randomness doesn't come from bootstrapping of data, but rather comes from the random splits of all observations.

Extra Trees can be implemented from Scikit-learn. The three hyperparameters important for tuning are max_feature, min_samples_leaf, and n_estimators. The three key parameters explicitly, with the following statement.

- "The parameters K, nmin and M have different effects: K determines the strength of the attribute selection process, nmin the strength of averaging output noise, and M the strength of the variance reduction of the ensemble model aggregation."

3.8 Gradient Boosting

3.8.1 Theoretical Background

A loss function is necessary for the Gradient Boosting Classifier to operate. Gradient boosting classifiers can handle a variety of standardized loss functions in addition to customized loss functions, however the loss function must be differentiable. Classification algorithms frequently use logarithmic loss, while regression algorithms can use squared errors.

Gradient boosting systems don't have to derive a new loss function every time the boosting algorithm is added, rather any differentiable loss function can be applied to the system. Gradient boosting systems have two other necessary parts: a weak learner and an additive component. Gradient boosting systems use decision trees as their weak learners. Regression trees are used for the weak learners, and these regression trees output real values. Because the outputs are real values, as new learners are added into the model the output of the regression trees can be added together to correct for errors in the predictions. The additive component of a gradient boosting model comes from the fact that trees are added to the model over time, and when this occurs the existing trees aren't manipulated, their values remain fixed.

A procedure similar to gradient descent is used to minimize the error between given parameters. This is done by taking the calculated loss and performing gradient descent to reduce that loss. Afterwards, the parameters of the tree are modified to reduce the residual loss. The new tree's output is then appended to the output of the previous trees used in the model. This process is repeated until a previously specified number of trees is reached, or the loss is reduced below a certain threshold

- **Steps to Gradient Boosting**

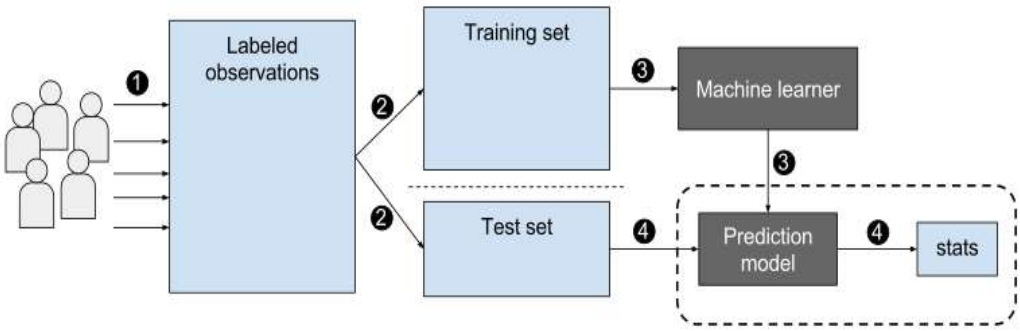

Fig. 3.6: Gradient Boosting Steps (classification)

In order to implement a gradient boosting classifier, we'll need to carry out a number of different steps. We'll need to:

- Different Improved Gradient Boosting Classifiers

Because of the fact that grading boosting algorithms can easily over fit on a training data set, different constraints or regularization methods can be utilized to enhance the algorithm's performance and combat overfitting. Penalized learning, tree constraints, randomized sampling, and shrinkage can be utilized to combat overfitting.

➢ Penalized Learning:

Certain constraints can be utilized to prevent overfitting, depending on the structure of the decision tree. The type of decision tree used in gradient boosting is a regression tree, which has numeric values as leaves or weights. These weight values can be regularized using the different regularization methods, like L1 or L2 regularization weights, which penalizes the radiant boosting algorithm.

➢ Tree Constraints:

There are many different ways to constrain a decision tree, including setting limits on its depth, the number of leaves or nodes, the number of observations each split, and the number of observations it is trained on. In general, more trees will be required for the model to accurately fit the data the more restrictions we employ when building trees.

- Random Sampling/Stochastic Boosting:

Stochastic gradient boosting, a method that involves randomly selecting subsamples from the training data set, can also aid in avoiding overfitting. In essence, this method weakens the link between the trees. The data set can be subsampled in a variety of ways, including subsampling rows before generating a tree, subsampling columns before each split, and more. In general, the model appears to benefit from subsampling at high rates that don't surpass 50% of the data.

- Shrinkage/Weighted Updates

Because the predictions of each tree are summed together, the contributions of the trees can be inhibited or slowed down using a technique called shrinkage. A "learning rate" is adjusted, and when the learning rate is reduced more trees must be added to the model. This makes it so that the model needs longer to train.

There's a trade-off between the learning rate and the number of trees needed, so we'll have to experiment to find the best values for each of the parameters, but small values less than 0.1 or values between 0.1 and 0.3 often work well.

- XGBoost

XGBoost is a refined and customized version of a gradient boosting decision tree system, created with performance and speed in mind. XGBoost actually stands for "eXtreme Gradient Boosting", and it refers to the fact that the algorithms and methods have been customized to push the limit of what is possible for gradient boosting algorithms.

3.9 AdaBoost

3.9.1 Theoretical Background

Ada Boost, also known as Adaptive Boosting, is a machine learning method used in an ensemble setting. Decision trees with one level, or Decision trees with only one split, are the most popular algorithm used with Ada Boost. Another name for these trees is Decision Stumps. This algorithm creates a model while assigning each data piece an equal weight. Then, it gives points that were incorrectly categorized larger weights. The next model now gives more weight to all the points with higher weights. If no low error is received, it will continue to train the models.

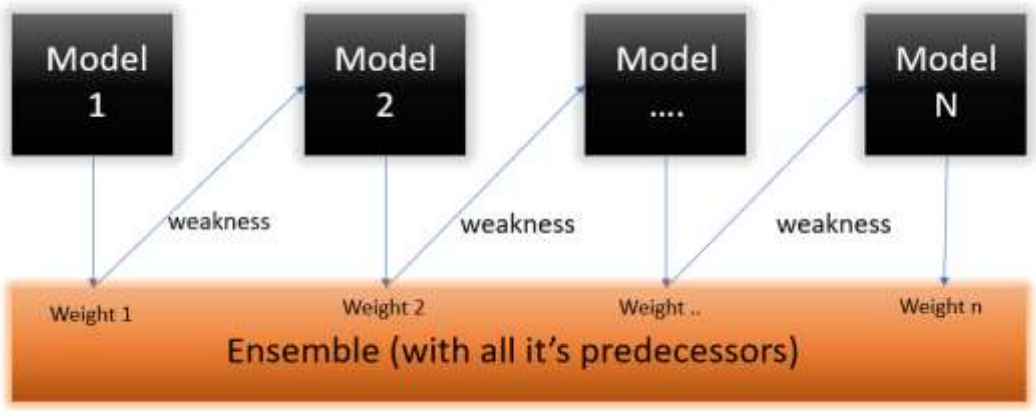

Fig. 3.7: Model of AdaBoost

The formula to calculate:

$$w(x_i, y_i) = \frac{1}{N}, \; i = 1, 2, \ldots . n$$

Where N is the total number of data points.

AdaBoost is adaptive in that it modifies future weak learners in favor of instances that prior classifiers incorrectly classified. It may be less prone to the over fitting issue than other learning algorithms in particular situations. It can be demonstrated that the final model converges to a strong learner even if the performance of each individual learner is just marginally better than random guessing. AdaBoost is frequently used to combine weak base learners (like decision stumps), but it has been demonstrated that it can also combine strong base learners (like deep decision trees) well, leading to an even more precise model.

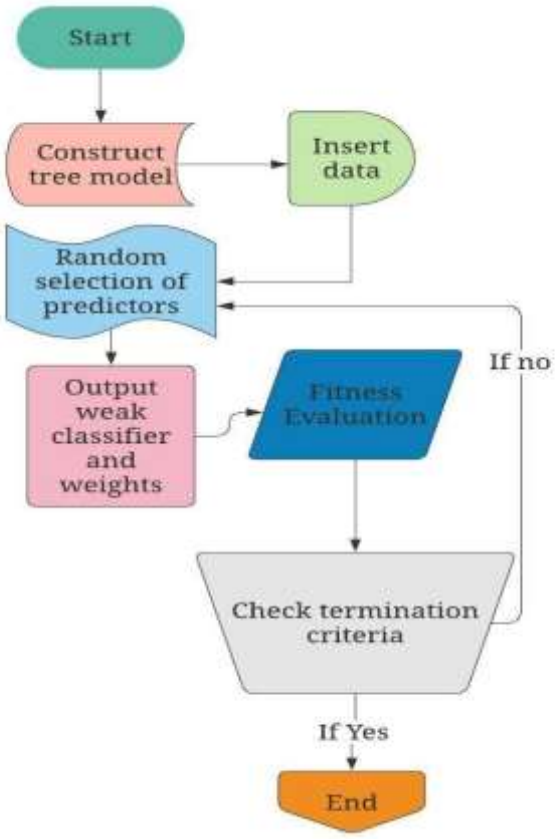

Fig. 3.8: Methodology flowchart of AdaBoost modelling

Chapter 4

Methodology

4.1 Overview

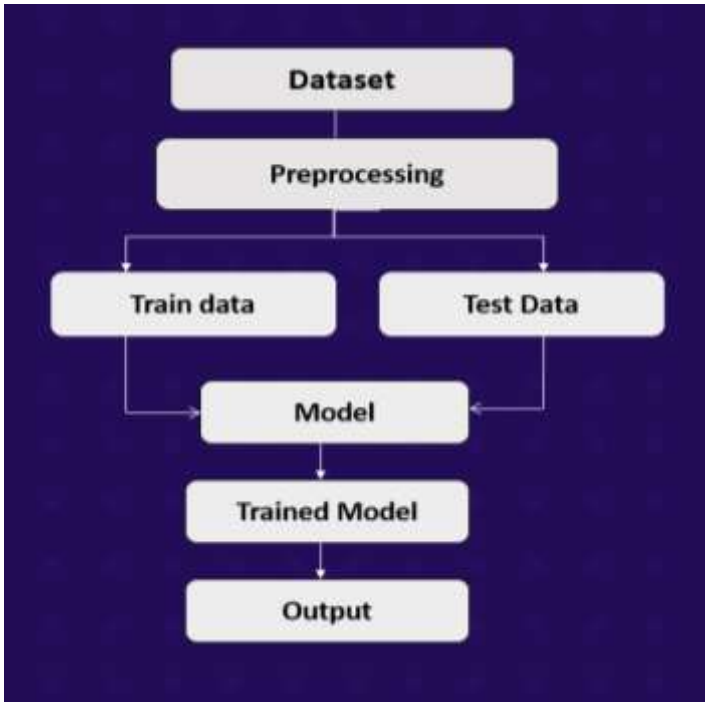

Fig. 4.1: Classification Framework

In Figure 4.1, our categorization strategy is displayed. For the purpose of developing our machine learning model, our first step would be to gather relevant data. Next step would be to prepare data. Data preprocessing, where we load our data into a suitable place and prepare it for use in our machine learning training. It includes data collection, cleansing, aggregation, augmentation, labeling, normalization and transformation as well as any other activities for structured, unstructured and semi-structured data. We'll also need to split the data in two parts. The first part, used in training our model, will be the majority of the dataset. The second part will be used for evaluating our trained model's performance. After that, the model Now We need to choose the model which determines the output we get after running a machine learning algorithm on the collected data. It is important to choose a model which is relevant to the task at hand. Now, after choosing this we need to train the model. In training, we pass the prepared data to our machine learning model to find patterns and make prediction. In results, in the model learning from the data so that it can accomplish the task set. Over time, with training the models get better at predicting. After training model we have to test the model. For checking the accuracy of the model by providing a test dataset to it.

4.2 Preprocessing

Before being utilized to train and test the models, input data are processed. Preprocessing is essentially the combination of a few different steps.

Fig. 4.2: Data Preprocessing

Obtaining the dataset is the first step in the data preparation process for machine learning. Before developing machine learning models, first obtain the necessary dataset. This dataset will be created using data that has been properly merged after being obtained from multiple and distinct sources. Depending on the use case, dataset formats change. Due to this, we obtained the epileptic seizure using EEG dataset. In the dataset, each folder refers has 100 files, each of which represents a particular topic or individual. Each file contains a 23.6-second recording of brain activity. 4097 data points from the associated time-series are sampled. Each data point represents the EEG recording's value at a particular time point. Thus, we have a total of 500 people and 4097 data points for each of them across 23.5 seconds. The dataset has Column 180 and Unique values 11500. We import Python libraries for data preparation in Machine Learning. Specific data pretreatment tasks can be carried out using the specified Python libraries. The second stage in machine learning's data preparation is importing all essential libraries. Numpy, pandas, seaborn, os, yellowbrick, sklearn, matplotlib and more Python libraries used for this data preparation in machine learning for importing the dataset you have amassed for the current ML project. Importing the dataset is one of the critical steps in preparing data for machine learning. Prior to importing the dataset, however, set the current directory as the working directory. We utilize Google Drive for the dataset's storage. It is critical to identify and handle missing values in a proper way while preparing data. The inference we get if we don't discover it will be incorrect. Obviously, this will make the ML project more difficult. Columns that aren't needed and null columns are deleted next. The following stage in machine learning's data

preparation is splitting the dataset. Each dataset must be divided into a training set and a test set before being used in a machine learning model. The subset of a dataset used to train a machine learning model is referred to as the training set. We are already aware of the result in this case. Contrarily, a test set is the portion of the dataset used to evaluate a machine learning model. The test set is used by the ML model to forecast results. The dataset is typically divided into 80:20 ratios. This means that either use 80 percent of the data for training the model and 30 percent or 20 percent of the remaining data is left out. The manner in which the dataset is divided differs depending on its size and form. We split into 80:20 ratio. Training and test sets are all present. Feature scaling in machine learning denotes the end of data preparation. It is a method for normalizing a dataset's independent variables inside a specific range. In other words, feature scaling narrows the scope of the variables so you can compare them. Data scaling is a helpful method since it makes it easier to comprehend the model. It improves training precision while accelerating training. The error surface's shape is improved, and training is expedited.

4.3 Machine Learning Algorithms

K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Logistic Regression, Gaussian Naive Bayes, Decision Tree, Random Forest, Extra Tree Classifier, Gradient Boosting and AdaBoost are used as models to find the best accuracy.

4.3.1 K-Nearest Neighbor

For Table- 4.1, creating a Class-label by the name Status which has binary set of values depending on the file contains a 23.6-second recording of brain activity. Each data point represents the EEG recording's value at a particular time point. Thus, we have a total of 500 people and 4097 data points for each of them across 23.5 seconds. At first we have found out the average value of 178 data points for 1 second, and each data point is the value of the EEG recording at a different point in time. In the dataset, they have done with the 23 x 500 = 11500 pieces of information(row), each information contains the 178 data points for 1 second(column), the last column represents the label y {1,2,3,4,5}. The response variable is y in column 179, the Explanatory variables X1, X2, …, X178. y contains the category of the 178-dimensional input vector. Specifically y in {1, 2, 3, 4, 5}. 5 - eyes open, means when they were recording the EEG signal of the brain the patient had their eyes open. 4 - eyes closed, means when they were recording the EEG signal the patient had their eyes closed. 3 - Yes they identify where the region of the tumor was in the brain and recording the EEG activity from the healthy brain area. 2 - They recorder the EEG from the area where the tumor was located. 1 - Recording of seizure activity. The value of average is greater than or equal then class-label will be "0". Otherwise the class label will be "1".

TABLE 4.1: K-Nearest Neighbor Result

Classification Report	Precision	Recall	F1 score	Support	Accuracy
0	0.92	1	0.96	1835	93.0000
1	0.99	0.66	0.79	465	
Accuracy			0.93	2300	
Macro avg	0.96	0.83	0.87	2300	
Weighted avg	0.94	0.93	0.92	2300	

4.3.2 Support Vector Machine

For **Table-4.2** this approach, creating a Class-label by the name Status which has binary set of values depending on the file contains a 23.6-second recording of brain activity. Each data point represents the EEG recording's value at a particular time point. The value of average is greater than or equal then class-label will be "0". Otherwise the class label will be "1". It speeds up training and improves the mistake surface form. Then split the dataset for train and test respectively 80 percent and 20 percent. After implementing the algorithm to the dataset our model shows the best accuracy which is shown in given table.

TABLE 4.2: Support Vector Machine result

Classification Report	Precision	Recall	F1 score	Support	Accuracy
0	0.99	0.99	0.99	1835	
1	0.97	0.94	0.96	465	
Accuracy			0.98	2300	98.2609
Macro avg	0.98	0.97	0.97	2300	
Weighted avg	0.98	0.98	0.98	2300	

4.3.3 Logistic Regression

For **Table-4.3** this approach, generating a Class-label called Status with a binary set of values based on a 23.6-second recording of brain activity in the file. We scaled the data to help the model learn more effectively. Scaling data is a good practice since it makes it simple to understand the model. Although it speeds up training, there are surface shape errors. It has the dataset's lowest accuracy.

TABLE 4.3: Logistic Regression result

Classification Report	Precision	Recall	F1 score	Support	Accuracy
0	0.81	1	0.90	1835	
1	0.96	0.10	0.18	465	
Accuracy			0.82	2300	81.6522
Macro avg	0.89	0.55	0.54	2300	
Weighted avg	0.84	0.82	0.75	2300	

4.3.4 Gaussian Naive Bayes

According to this method, a Class-label with the name Status and a binary set of values based on whether the file contains a 23.6-second recording of brain activity is created for **Table-4.4**. We used Gaussian Naive Bayes in this case. We scaled the data to help the model learn more effectively. Error surface form is made faster during training.

TABLE 4.4: Gaussian Naïve Bayes result

Classification Report	Precision	Recall	F1 score	Support	Accuracy
0	0.96	0.99	0.97	1865	
1	0.94	0.82	0.88	465	
Accuracy			0.95	2300	95.3478
Macro avg	0.95	0.90	0.92	2300	
Weighted avg	0.95	0.95	0.95	2300	

4.3.5 Decision Tree

For **Table-4.5**, with this method, we scaled the data to help the model learn more effectively. It improves the form of the error surface during training.

TABLE 4.5: Decision Tree result

Classification Report	Precision	Recall	F1 score	Support	Accuracy
0	0.92	0.96	0.94	1835	90.0000
1	0.81	0.66	0.73	465	
Accuracy			0.90	2300	
Macro avg	0.86	0.81	0.83	2300	
Weighted avg	0.90	0.90	0.90	2300	

4.3.6 Random Forest

For **Table-4.6,** we used the same way and method as "Decision Tree". It shows the better result than decision tree model.

TABLE 4.6: Random Forest result

Classification Report	Precision	Recall	F1 score	Support	Accuracy
0	0.97	0.99	0.98	1835	96.1739
1	0.94	0.87	0.90	465	
Accuracy			0.96	2300	
Macro avg	0.95	0.93	0.94	2300	
Weighted avg	0.96	0.96	0.96	2300	

4.3.7 Extra Tree Classifier

For **Table-4.7,** we scaled the data to help the model learn more effectively. Scaling data is a good practice since it makes it simple to understand the model. We followed the "Decision Tree" approach and methodology. There are numerous de-correlated decision trees produced as a result of this random sampling of features. It displays the dataset's second-best accuracy.

TABLE 4.7: Extra Tree Classifier result

Classification Report	Precision	Recall	F1 score	Support	Accuracy
0	0.98	0.99	0.98	1835	97.4883
1	0.97	0.91	0.94	465	
accuracy			0.97	2300	
Macro avg	0.97	0.95	0.96	2300	
Weighted avg	0.97	0.97	0.97	2300	

4.3.8 Gradient Boosting

Gradient Boosting trains many models in a gradual, additive and sequential manner. The term gradient boosting emerged because every case's target outcomes are based on the gradient's error with regards to the predictions. Every model reduces prediction errors by taking a step in the correct direction.

For **Table-4.8,** after implementing the algorithm to the dataset our model shows the accuracy which is shown in given table.

TABLE 4.8: Gradient Boosting result

Classification Report	Precision	Recall	F1 score	Support	Accuracy
0	0.95	0.99	0.97	1835	95.3043
1	0.96	0.80	0.87	465	
Accuracy			0.95	2300	
Macro avg	0.96	0.90	0.92	2300	
Weighted avg	0.95	0.95	0.95	2300	

4.3.9 AdaBoost

For **Table-4.9,** the most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps. AdaBoost builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now, all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a low error is received. It is the second lowest accuracy for the dataset.

TABLE 4.9: AdaBoost result

Classification Report	Precision	Recall	F1 score	Support	Accuracy
0	0.90	0.95	0.93	1835	87.8261
1	0.76	0.58	0.66	465	
Accuracy			0.88	2300	
Macro avg	0.83	0.77	0.79	2300	
Weighted avg	0.87	0.88	0.87	2300	

Chapter 5

Performance Matrix

5.1 Confusion matrix

Confusion matrix is one of the basic table layout that helps to visualize the performance of a classification and detection related machine learning algorithms. A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

TABLE 5.1: Confusion Matrix

	True Class Label		
Predicted Class Label		Positive (1)	Negative (0)
	Positive	***TP***	***FP***
	Negative	***FN***	***TN***

Understanding True Positive, True Negative, False Positive and False Negative in a Confusion Matrix

- True Positive (TP)

The predicted value matches the actual value

The actual value was positive and the model predicted a positive value

- True Negative (TN)

The predicted value matches the actual value

The actual value was negative and the model predicted a negative value

- False Positive (FP) – Type 1 error

The predicted value was falsely predicted

The actual value was negative but the model predicted a positive value

Also known as the Type 1 error

- False Negative (FN) – Type 2 error

The predicted value was falsely predicted

The actual value was positive but the model predicted a negative value

Also known as the Type 2 error

We can perform various calculations for the model, such as the model's accuracy, using this matrix. These calculations are given below:

Classification Accuracy: It is one of the important parameters to determine the accuracy of the classification problems. It defines how often the model predicts the correct output. It can be calculated as the ratio of the number of correct predictions made by the classifier to all number of predictions made by the classifiers. The formula is given below:

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

Misclassification rate: It is also termed as Error rate, and it defines how often the model gives the wrong predictions. The value of error rate can be calculated as the number of incorrect predictions to all number of the predictions made by the classifier. The formula is given below:

$$\text{Error rate} = (FP + FN)/(TP + FP + FN + TN)$$

Precision: It can be defined as the number of correct outputs provided by the model or out of all positive classes that have predicted correctly by the model, how many of them were actually true. It can be calculated using the below formula:

$$Precision = \frac{TP}{TP + FP}$$

Recall: It is defined as the out of total positive classes, how our model predicted correctly. The recall must be as high as possible.

$$Recall = \frac{TP}{TP + FN}$$

F-measure: If two models have low precision and high recall or vice versa, it is difficult to compare these models. So, for this purpose, we can use F-score. This score helps us to evaluate the recall and precision at the same time. The F-score is maximum if the recall is equal to the precision. It can be calculated using the below formula:

$$F - measure = \frac{2*Recall*Precision}{Recall + Precision}$$

5.2 Experiment and Evaluation

5.2.1 Dataset

Each folder in the original dataset from the reference has 100 files, each of which represents a particular topic or individual. Each file contains a 23.6-second recording of brain activity. 4097 data points from the associated time-series are sampled. Each data point represents the EEG recording's value at a particular time point. Thus, we know that total of 500 people and 4097 data points for each of them across 23.5 seconds. Every 4097 data points were separated and randomly jumbled into 23 chunks, with each chunk containing 178 data points for 1 second. Each data point represents the value of the EEG recording at a distinct moment. Therefore, now have 23 x 500 = 11 500 informational pieces (row), each of which comprises 178 data points for 1 second (column), and the last column is the label y [1, 2, 3, 4]. The response variable is y in column 179, the Explanatory variables X1, X2, …, X178

y contains the category of the 178-dimensional input vector. Specifically y in {1, 2, 3, 4, 5}:

5 - eyes open, means when they were recording the EEG signal of the brain the patient had their eyes open

4 - eyes closed, means when they were recording the EEG signal the patient had their eyes closed

3 – Yes, they identify where the region of the tumor was in the brain and recording the EEG activity from the healthy brain area

2 - They recorder the EEG from the area where the tumor was located

1 - Recording of seizure activity

All subjects falling in classes 2, 3, 4, and 5 are subjects who did not have epileptic seizure. Only subjects in class 1 have epileptic seizure. Our motivation for creating this version of the data was to simplify access to the data via the creation of a .csv version of it. Although there are 5 classes most authors have done binary classification, namely class 1 (Epileptic seizure) against the rest.

Using machine learning algorithm, we find out the positive data 2300 and negative data 9200, where we represent the positive data by 1 and negative data by 0

5.3 K-Nearest Neighbor Performance

Our first algorithm is K-Nearest Neighbor. From **Table 5.2**, we find the accuracy = 93.00%, with respect to the power parameter for the Minkowski metric. When p=1, this is equivalent to using manhattam distance (l1), and Euclidean distance (l2) for p=2.
For applying KNN, we used KNN Classifier = KNeighbors Classifier (n_neighbors = 3)
For this we find the classification report which is given below

TABLE 5.2: K-Nearest Neighbor Performance

Classification Report	Precision	Recall	F1 score	Support
0	0.92	1	0.96	1835
1	0.99	0.66	0.79	465
Accuracy			0.93	2300
Macro avg	0.96	0.83	0.87	2300
Weighted avg	0.94	0.93	0.92	2300

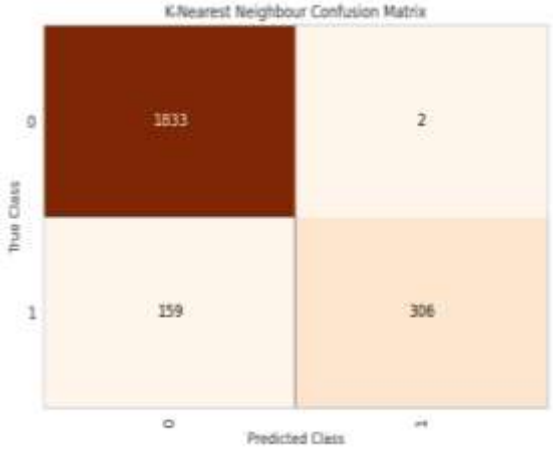

Fig. 5.1: Confusion Matrix (KNN)

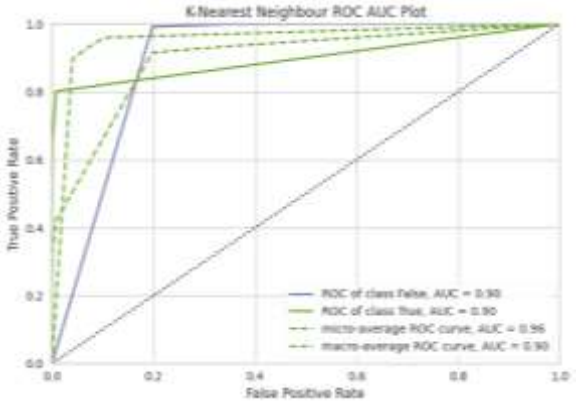

Fig. 5.2: ROC AUC Plot (KNN)

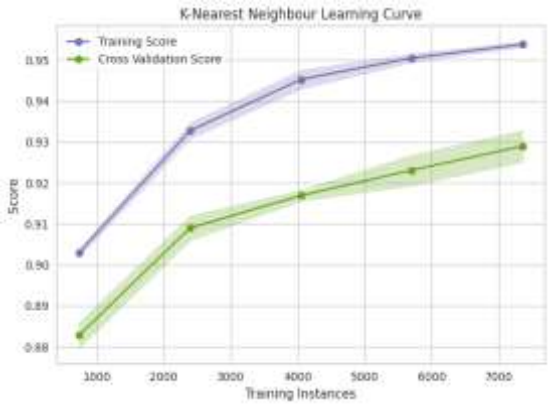

Fig. 5.3: F1 Score (KNN)

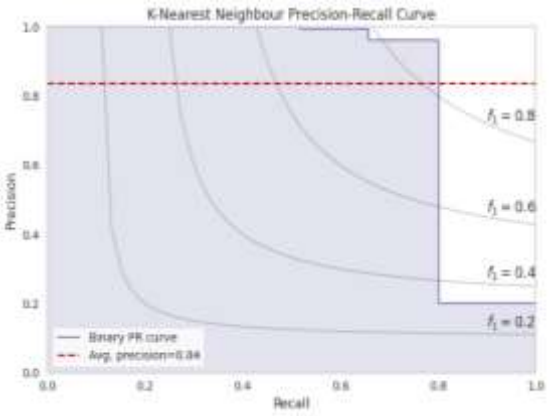

Fig. 5.4: Precision-Recall Curve (KNN)

Important information about the machine learning performance assessment metrics, including precision, recall, F1 Score, and support score, is provided in the table. These metrics are used to evaluate the classification model we constructed in terms of support score, precision, and recall. The values listed in the table for confusion matrix 0 are: precision is 0.92, Recall is 1, F1

score is 0.96, and support is 1835. And for confusion matrix 1 Precision = 0.99, recall = 0.66, F1 score = 0.79, and support value = 465.

5.4 Support Vector Machine Performance

Secondly, we applied Support Vector Machine, from **table 5.3**, we get the accuracy which is the best from other models accuracy, the accuracy 98.26%. With respect to the classifier is:
SVMclassifier = SVC(kernel='rbf', max_iter=1000, C=10, probability=True)

TABLE 5.3: Support Vector Machine Performance

Classification Report	Precision	Recall	F1 score	Support
0	0.99	0.99	0.99	1835
1	0.97	0.94	0.96	465
Accuracy			0.98	2300
Macro avg	0.98	0.97	0.97	2300
Weighted avg	0.98	0.98	0.98	2300

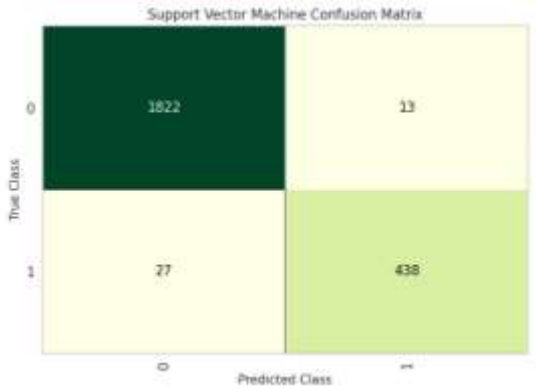

Fig. 5.5: Confusion Matrix (SVM)

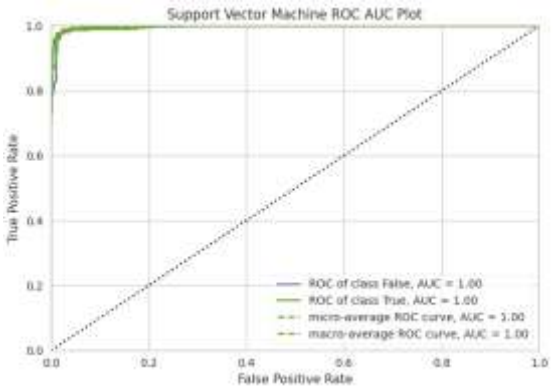

Fig. 5.6: ROC AUC Plot (SVM)

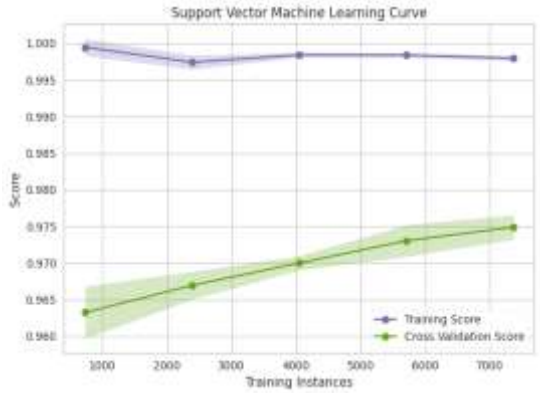

Fig. 5.7: F1 Score (SVM)

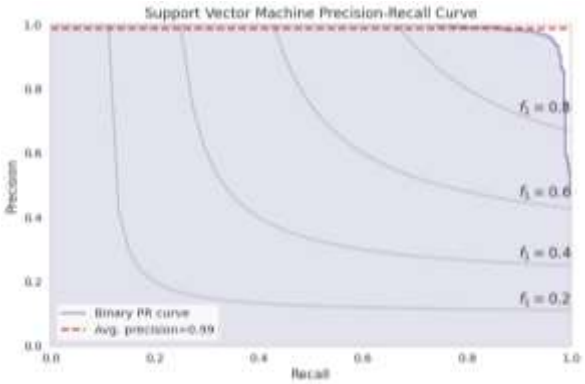

Fig. 5.8: Precision-Recall Curve (SVM)

In the classification table, displays some significant information about machine learning performance evaluation measures, which are used to reflect the accuracy, recall, F1 Score, and support score of our trained classification model. For confusion matrix 0, the following values are displayed in the table: precision = 0.99, recall = 0.99, F1 score = 0.99, and support = 1835. Confusion Matrix 1 has a precision = 0.97 recall = 0.94, F1 score = 0.96 and the support value is 465.

5.5 Logistic Regression Performance

The next algorithm is Logistic Regression. From Table 5.4, we find the accuracy = 81.65%, which is the lowest accuracy. The classifier is:
classifier_lr = Logistic Regression (random_state = 42)

TABLE 5.4: Logistic Regression Performance

Classification Report	Precision	Recall	F1 score	Support
0	0.81	1	0.90	1835
1	0.96	0.10	0.18	465
Accuracy			0.82	2300
Macro avg	0.89	0.55	0.54	2300
Weighted avg	0.84	0.82	0.75	2300

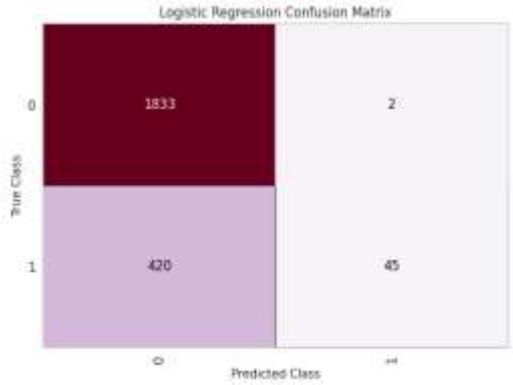

Fig. 5.9: Confusion Matrix (LR)

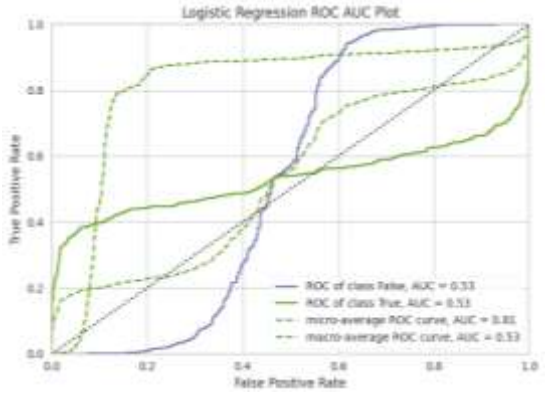

Fig. 5.10: ROC AUC Curve (LR)

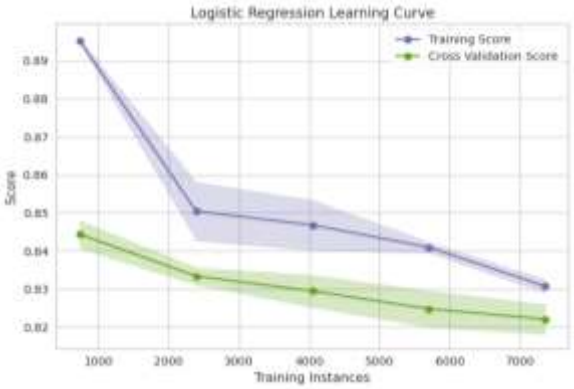

Fig. 5.10: F1 Score (LR)

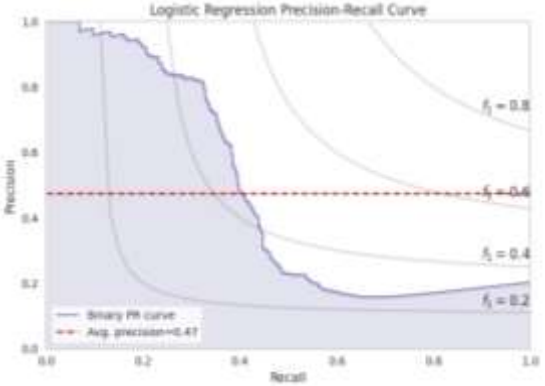

Fig. 5.12: Precision-Recall Curve (LR)

Important information about the machine learning performance assessment metrics, including precision, recall, F1 Score, and support score, is provided in the table. These metrics are used to evaluate the classification model we constructed in terms of support score, precision, and recall. The values listed in the table for confusion matrix 0 are: precision is 0.81, Recall is 1, F1 score is 0.90, and support is 1835. And for confusion matrix 1 Precision = 0.96, recall = 0.55, F1 score = 0.54, and support value = 465.

5.6 Gaussian Naive Bayes Performance

The next algorithm is Gaussian Naïve Bayes. From **Table 5.5**, we find the accuracy = 95.35%. The classifier is:
GNB classifier = Gaussian NB (var_smoothing=0.1).

TABLE 5.5: Gaussian Naïve Bayes Performance

Classification Report	Precision	Recall	F1 score	Support
0	0.96	0.99	0.97	1865
1	0.94	0.82	0.88	465
Accuracy			0.95	2300
Macro avg	0.95	0.90	0.92	2300
Weighted avg	0.95	0.95	0.95	2300

Fig. 5.13: Confusion Matrix (GNB)

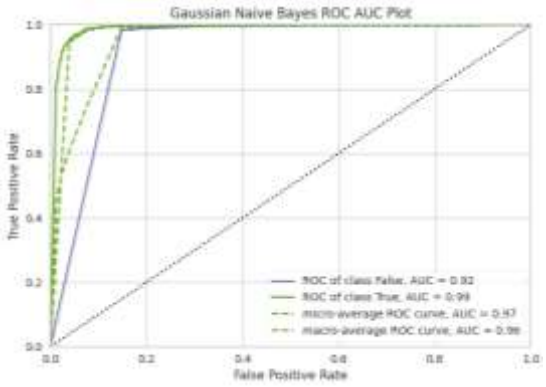

Fig. 5.14: ROC AUC Plot (GNB)

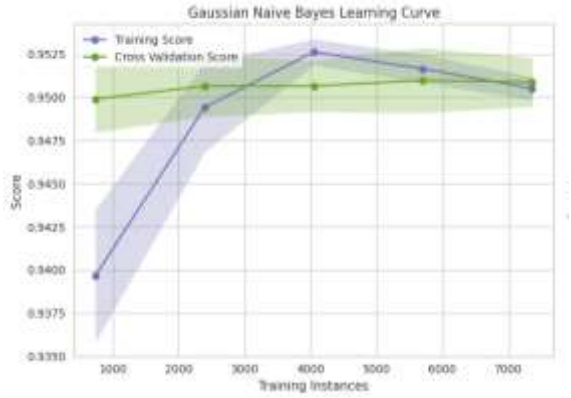

Fig. 5.15: F1 Score (GNB)

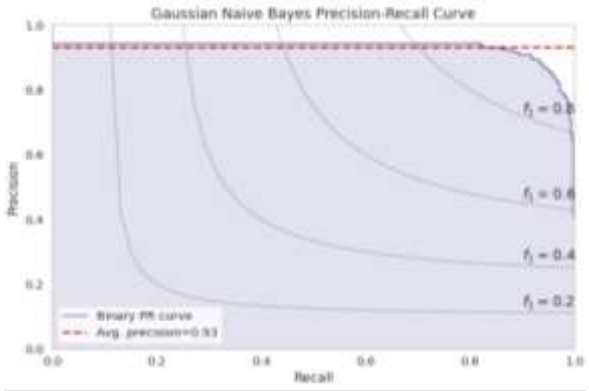

Fig. 5.16: Precision-Recall Curve (GNB)

The table displays some significant information about machine learning performance evaluation measures, which are used to reflect the accuracy, recall, F1 Score, and support score of our trained classification model. For confusion matrix 0, the following values are displayed in the table: precision = 0.96, recall = 0.99, F1 score = 0.97, and support

= 1835. Confusion Matrix 1 has a precision = 0.94 recall = 0.82, F1 score = 0.88 and the support value is 465.

5.7 Decision Tree Performance

The next algorithm is Decision Tree Performance. From **Table 5.6**, we find the accuracy = 90.00%, the classifier is:

DTCclassifier = DecisionTreeClassifier(max_depth=3, min_samples_leaf=5, criterion='entropy', m in_samples_split=5,splitter='random', random_state=1).

TABLE 5.6: Decision Tree performance

Classification Report	Precision	Recall	F1 score	Support
0	0.92	0.96	0.94	1835
1	0.81	0.66	0.73	465
Accuracy			0.90	2300
Macro avg	0.86	0.81	0.83	2300
Weighted avg	0.90	0.90	0.90	2300

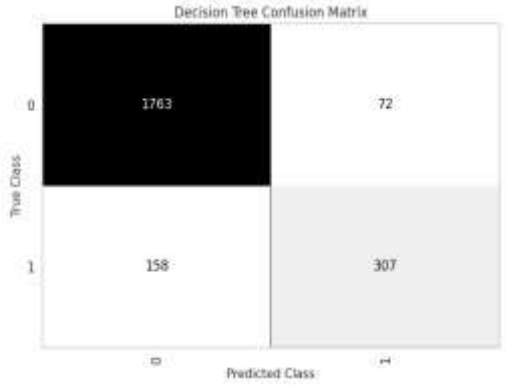

Fig. 5.17: Confusion Matrix (DT)

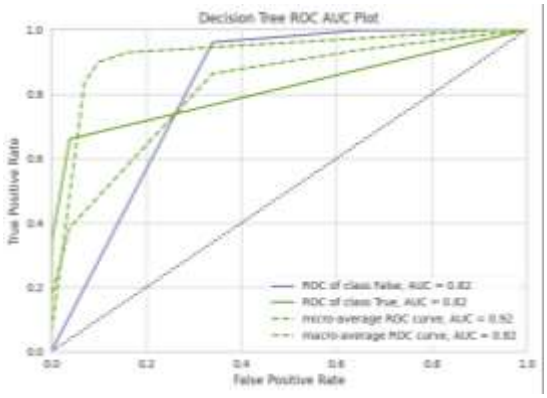

Fig. 5.18: ROC AUC Plot (DT)

Fig. 5.19: F1 Score (DT)

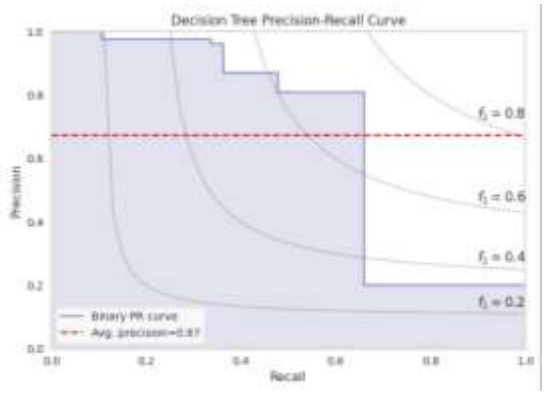

Fig. 5.20: Precision-Recall Curve (DT)

About the machine learning performance assessment metrics, including precision, recall, F1 Score, and support score, is provided in the table. These metrics are used to evaluate the classification model we constructed in terms of support score, precision, and recall. The values listed in the table for confusion matrix 0 are: precision is 0.92, Recall is 0.96, F1 score is 0.94, and support

is 1835. And for confusion matrix 1 Precision = 0.81, recall = 0.66, F1 score = 0.73, and support value = 465.

5.8 Random Forest Performance

Here we applied Random Forest Performance. From Table 5.7, we find the accuracy = 96.17%. The classifier is:

RFclassifier = RandomForestClassifier(n_estimators=1000, random_state=1, max_leaf_nodes=20, min_samples_split=15)

TABLE 5.7: Random Forest performance

Classification Report	Precision	Recall	F1 score	Support
0	0.97	0.99	0.98	1835
1	0.94	0.87	0.90	465
Accuracy			0.96	2300
Macro avg	0.95	0.93	0.94	2300
Weighted avg	0.96	0.96	0.96	2300

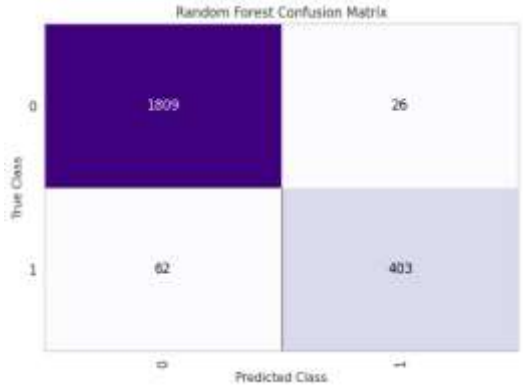

Fig. 5.21: Confusion Matrix (RF)

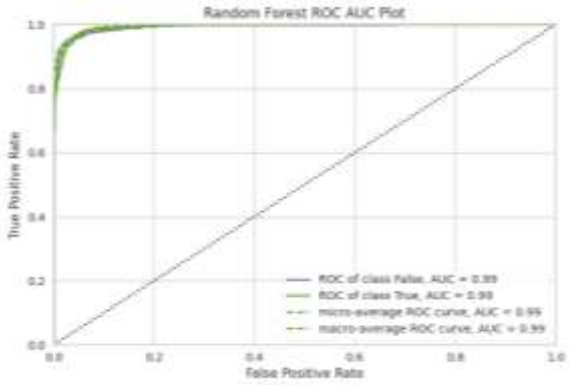

Fig. 5.22: ROC AUC Plot (RF)

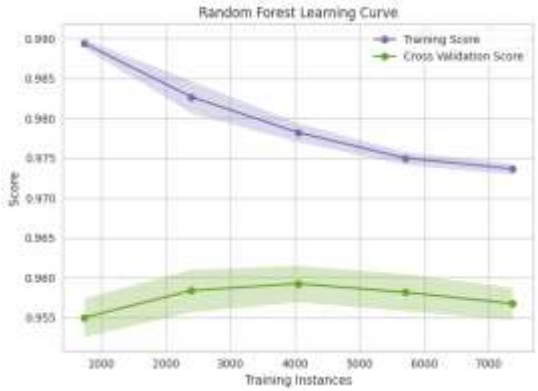

Fig. 5.23: F1 Score (RF)

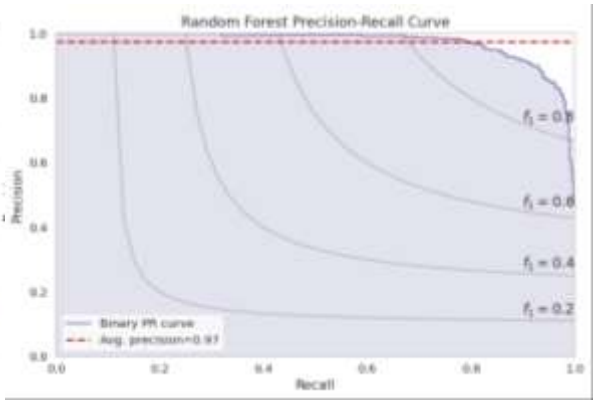

Fig. 5.24: Precision-Recall Curve (RF)

The table contains crucial information about the machine learning performance assessment metrics, which are used to evaluate the precision, recall, F1 Score, and support score of our created classification model. These metrics are used to evaluate the classification model we constructed in

terms of support score, precision, and recall. The values listed in the table for confusion matrix 0 are: precision is 0.97, Recall is 0.99, F1 score is 0.98, and support is 1835. And for confusion matrix 1 Precision = 0.94, recall = 0.87, F1 score = 0.90, and support value = 465

5.9 Extra Tree Classifier Performance

After that we applied Extra Tree Classifier Performance. From Table 5.8, we find the accuracy = 97.48%. The classifier is:
ETclassifier = Extra Trees Classifier (n_estimators=15, random_state=47)

TABLE 5.8: Extra Tree Classifier performance

Classification Report	Precision	Recall	F1 score	Support
0	0.98	0.99	0.98	1835
1	0.97	0.91	0.94	465
Accuracy			0.97	2300
Macro avg	0.97	0.95	0.96	2300
Weighted avg	0.97	0.97	0.97	2300

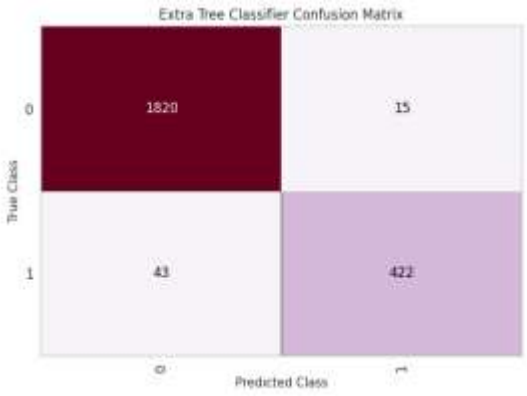

Fig. 5.25: Confusion Matrix (ETC)

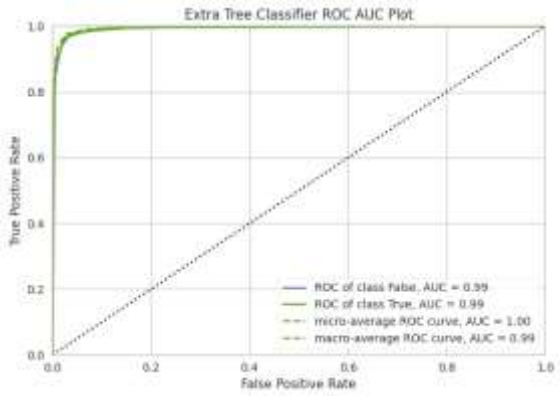

Fig. 5.26: ROC AUC Plot (ETC)

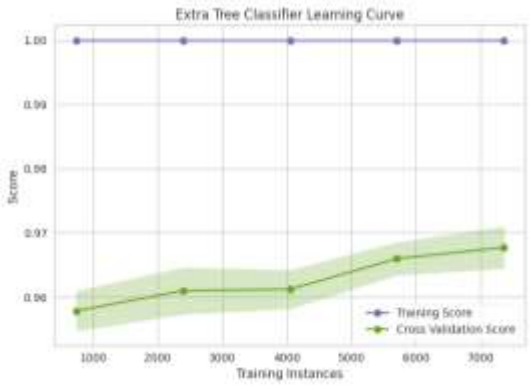

Fig. 5.27: F1 Score (ETC)

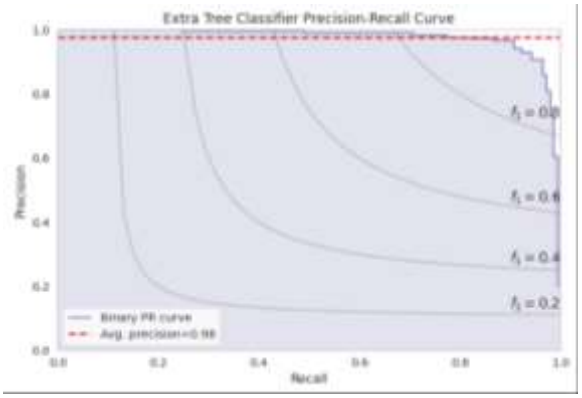

Fig. 5.28: Precision-Recall Curve (ETC)

In the table information about the machine learning performance assessment metrics, which are used to evaluate the precision, recall, F1 Score, and support score of our created classification

model. The values listed in the table for confusion matrix 0 are: precision is 0.98, Recall is 0.99, F1 score is 0.98, and support is 1835. And for confusion matrix 1 Precision = 0.97, recall = 0.91, F1 score = 0.94, and support value = 465.

5.10 Gradient Boosting Performance

In the Gradient Boosting Performance, we find out the accuracy is = 95.30% from **Table**5.9. The classifier is:

GB classifier = Gradient Boosting Classifier
(random_state=1, n_estimators=100, max_leaf_nodes=3, loss='exponential', min_samples_leaf=2).

TABLE 5.9: Gradient Boosting performance

Classification Report	Precision	Recall	F1 score	Support
0	0.95	0.99	0.97	1835
1	0.96	0.80	0.87	465
Accuracy			0.95	2300
Macro avg	0.96	0.90	0.92	2300
Weighted avg	0.95	0.95	0.95	2300

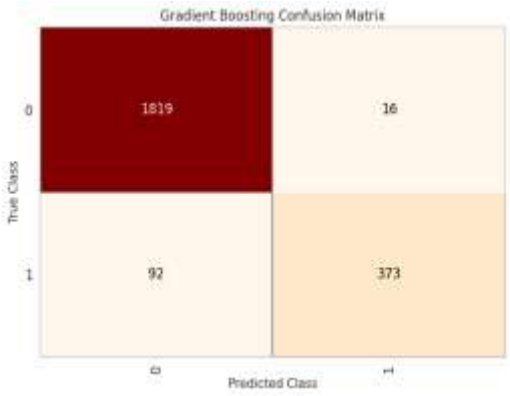

Fig. 5.29: Confusion Matrix (GB)

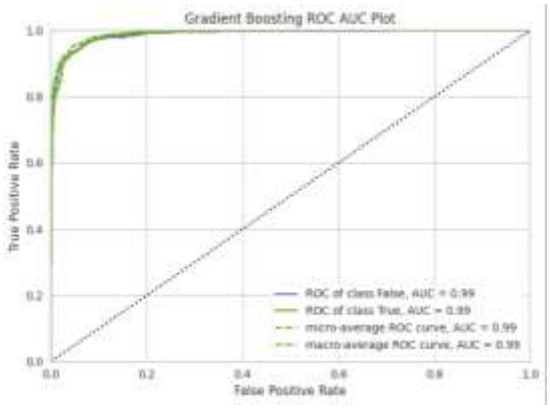

Fig. 5.30: ROC AUC Plot (GB)

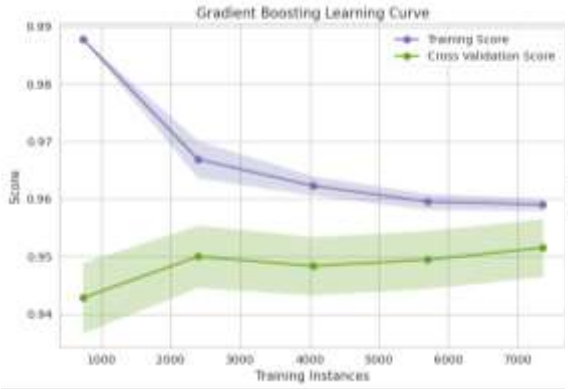

Fig. 5.31: F1 Score (GB)

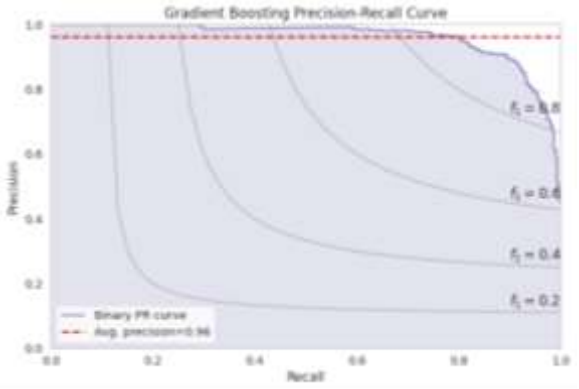

Fig. 5.32: Precision-Recall Curve (GB)

Essential information about the machine learning performance assessment metrics, including precision, recall, F1 Score, and support score, is provided in the table. These metrics are used to evaluate the classification model we constructed in terms of support score, precision, and recall. The values listed in the table for confusion matrix 0 are: precision is 0.95, Recall is 0.99 , F1 score is 0.97, and support is 1835. And for confusion matrix 1 Precision = 0.96, recall = 0.80, F1 score = 0.87, and support value = 465.

5.11 AdaBoost Performance

The last algorithm is AdaBoost. From **table 5.10**, we find the accuracy 87.83%. The classifier is: AB classifier = AdaBoost Classifier(n_estimators=3).

TABLE 5.10: AdaBoost Performance

Classification Report	Precision	Recall	F1 score	Support
0	0.90	0.95	0.93	1835
1	0.76	0.58	0.66	465
Accuracy			0.88	2300
Macro avg	0.83	0.77	0.79	2300
Weighted avg	0.87	0.88	0.87	2300

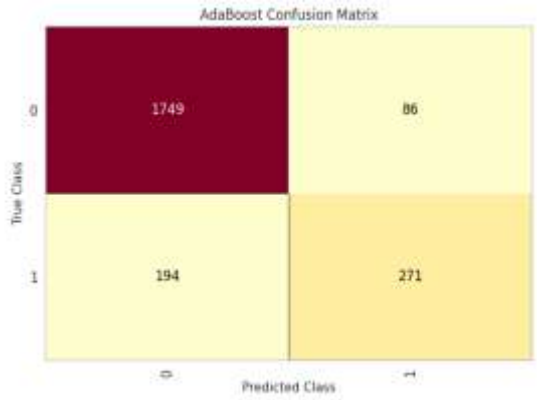

Fig. 5.33: Confusion Matrix (AdaBoost)

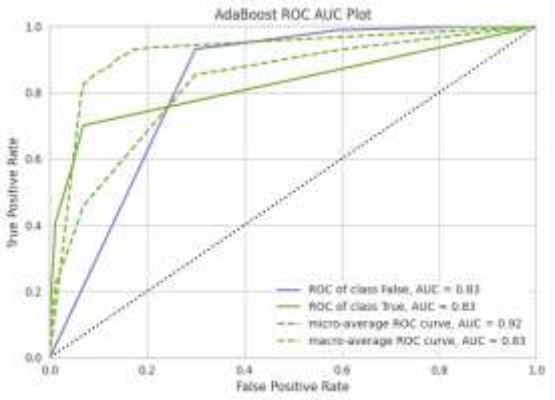

Fig. 5.34: ROC AUC Plot (AdaBoost)

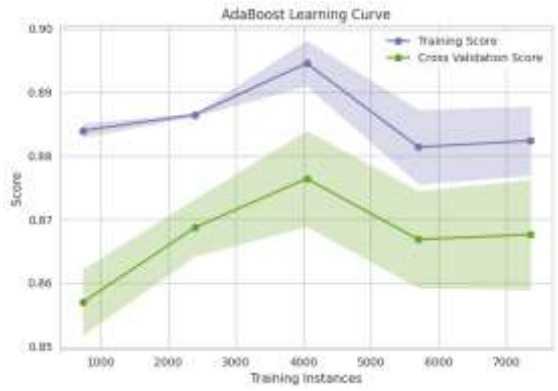

Fig. 5.35: F1 Score (AdaBoost)

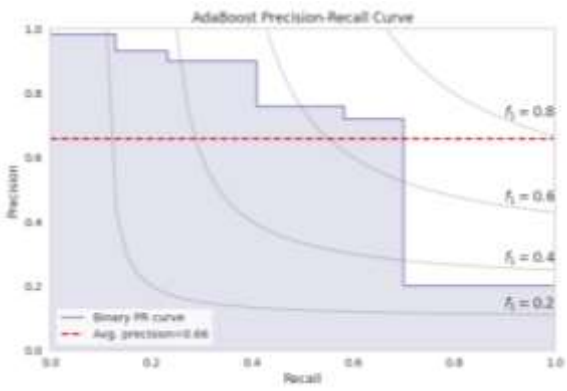

Fig. 5.36: Precision-Recall Curve (AdaBoost)

The table contains crucial information about the machine learning performance assessment metrics, which are used to evaluate the precision, recall, F1 Score, and support score of our created

classification model. The values listed in the table for confusion matrix 0 are: precision is 0.90, Recall is 0.95, F1 score is 0.93, and support is 1835. And for confusion matrix 1 Precision = 0.76, recall = 0.58, F1 score = 0.66 and support value = 465.

5.12 Comparison

From figure shows that the Support Vector Machine has the highest accuracy of all. Extra Tree Classifier and Random Forest has almost same accuracy. Gradient Boosting and Gaussian Naive Bayes are almost equally accurate. KNN's accuracy is lower than Gradient Boosting. Decision Tree and AdaBoost's accuracy is greater than Logistic Regression. The accuracy of Logistic Regression is the lowest of all. In addition, Support Vector Machine performs better with this dataset because it changes more quickly than other classifiers. However, the Support Vector Machine yields 98.2609 accuracy rates, which is the highest accuracy to date.

Index	Model	Accuracy
1.	Support Vector Machine	98.2609
2.	Extra Tree Classifier	97.4783
3.	Random Forest	96.1739
4.	Gaussian Naive Bayes	95.3478
5.	Gradient Boosting	95.3043
6.	K-Nearest Neighbour	93.0000
7.	Decision Tree	90.0000
8.	AdaBoost	87.8261
9.	Logistic Regression	81.6522

Fig. 5.37: Result

Fig. 5.38: Accuracy Comparison with Bar Graph

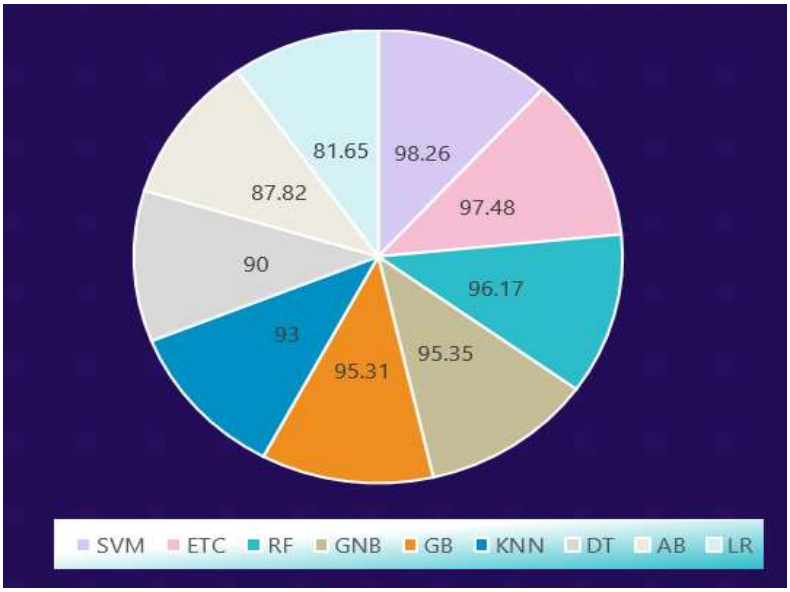

Fig. 5.39: Accuracy Comparison with Pie Graph

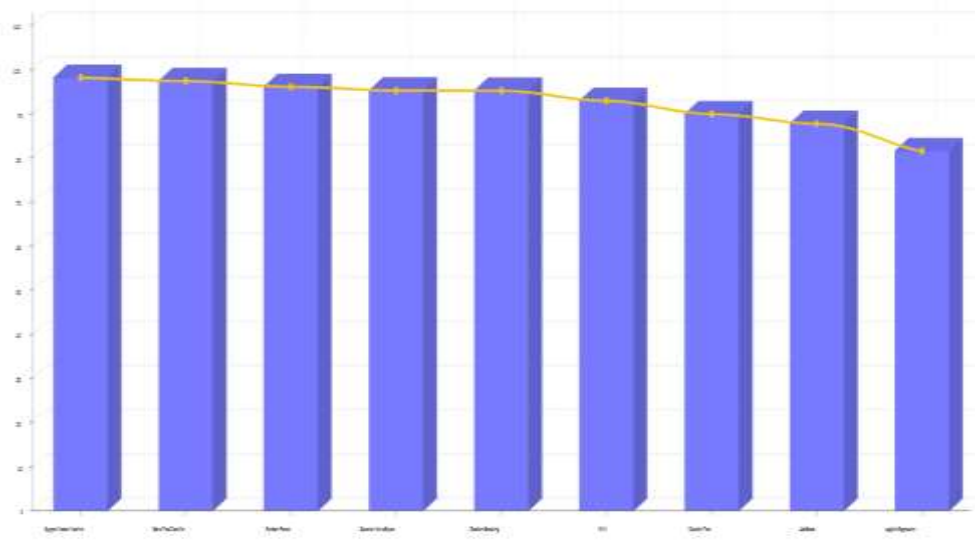

Fig. 5.40: Combination of Bar Graph & Line Graph

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this paper, with the dataset it was an attempt to implement nine different algorithms. We just tried to help the doctors who determine to the epileptic seizure disease by making a prediction of the EEG signals which must be good enough. Here we tried to Build various ML models that can predict whether patients have disease or not by using EEG recording's value at a particular time point. Our nine models show different levels of accuracy, which helps us to get an idea about how "Machine Learning" can be applied in the field of the epileptic seizure, which is very important to investigate in the medical science. Otherwise, it could not be easy to find out the epileptic seizure in a short time.

6.2 Future Work

The field of seizure detection, prediction has the undergone a lot of significant breakthroughs. The creation of seizure-treating gadgets offers some hope for patients who are drug-resistant. Only two sub-types of the two main types of epilepsy, generalized and partial, were represented by the epileptic seizure types that were the subject of this study. Future research could try to predict seizures in patients with the remaining forms of generalized epilepsy, as well as partial epilepsy, such as simple and complex partial seizures. By doing this, we can investigate the possibility of generalizing these prediction models to all varieties of epileptic seizure.
In the future, we will work with new data for increasing the accuracy. And also we will work with the real time data. As in this work, we have used only supervised machine learning. So that, we will use unsupervised machine learning for better result.

Bibliography

[1] Abdalla Gabara, Retaj Yousri, Darine Hamdy, Michael H. Zakhari, & Hassan Mostafa. (2021). Patient Specific Epileptic Seizures Prediction based on Support Vector Machine. *IEEE Xplore, 32nd International Conference on Microelectronics (ICM)*.

[2] Ahmet Alkan, Etem Koklukaya, & Etem Koklukaya. (2005). Automatic seizure detection in EEG using logistic regression and artificial neural network. *Elsevier, Journal of Neuroscience Methods.*

[3] Ali H. Shoeb, & John V. Guttag. (2010). Application of Machine Learning To Epileptic Seizure Detection. *Proceedings of the 27th International Conference on Machine Learning*.

[4] Bekir Karlik, & Şengül Bayrak. (2014). Comparison Machine Learning Algorithms for Recognition of Epileptic Seizures in EEG. *International Work-Conference on Bioinformatics and Biomedical Engineering* .

[5] Cher Hau Seng, R. Demirli, Lunal Khuon, & Donovan Bolger. (2012). Seizure detection in EEG signals using support vector machines. *IEEE, 38th Annual Northeast Bioengineering Conference (NEBEC)*.

[6] Dwi Sunaryono, Riyanarto Sarno, & Joko Siswantoro. (2021). Gradient boosting machines fusion for automatic epilepsy detection from EEG signals based on wavelet features. *Elsevier, Journal of King Saud University - Computer and Information Sciences*.

[7] Fauzia P. Lestari, Mohammad Haekal, Rizki Edmi Edison, & Fikry Ravi Fauzy. (2020). Epileptic Seizure Detection in EEGs by Using Random Tree Forest, Naïve Bayes and KNN Classification. *Journal of Physics Conference Series*.

[8] Garineh Sarkies Ohannesian, & Esraa Jasim Harfash. (2022). EPILEPTIC SEIZURES DETECTION FROM EEG RECORDINGS BASED ON A HYBRID SYSTEM OF GAUSSIAN MIXTURE MODEL AND RANDOM FOREST CLASSIFIER. *Informatica An International Journal of Computing and Informatics*.

[9] Harikumar Rajaguru, & Sunil Kumar Prabhakar. (2017). Analysis of adaboost classifier from compressed EEG features for epilepsy detection. *IEEE, International Conference on Computing Methodologies and Communication (ICCMC)*.

[10] Khaled Mohamad Almustafa. (2020). Classification of epileptic seizure dataset using different machine learning algorithms. *Elsevier, Informatics in Medicine Unlocked*.

[11] Mohammad Khubeb Siddiqui, Ruben Morales-Menendez, Xiaodi Huang, & Nasir Hussain. (2020). A review of epileptic seizure detection using machine learning classifiers. *Brain Inform.*

[12] Poomipat Boonyakitanont, Apiwat Lek-uthai, Krisnachai Chomtho, & Jitkomut Songsiri. (2019). A review of feature extraction and performance evaluation in epileptic seizure detection using EEG. *eess*.

[13] Rekha Sahu, Satya Ranjan Dash, Lleuvelyn Areglado Cacha, & R. R. Poznansky. (2020). Epileptic seizure detection: a comparative study between deep and traditional machine learning techniques. *Journal of Integrative Neuroscience*.

[14] Sarthak Gupta, Siddhant Bagga, Vikas Maheshkar, & M. P. S. Bhatia. (2020). Detection of Epileptic Seizures using EEG Signals. *IEEE, International Conference on Artificial Intelligence and Signal Processing*.

[15] Wail Mardini, Muneer Masadeh Bani Yassein, Rana Al-Rawashdeh, Shadi Aljawarneh, & Yaser M. Khamayseh. (2020). Enhanced Detection of Epileptic Seizure Using EEG Signals in Combination With Machine Learning Classifiers. *IEEE Access*.