UNIVERSITY INSTITUTE OF TECHNOLOGY
BARKATULLAH UNIVERSITY, BHOPAL

DEPARTMENT OF INFORMATION TECHNOLOGY

A MAJOR PROJECT REPORT ON

"DIAGNOSIS AND PREDICTION OF ADHD DISEASE USING MACHINE LEARNING METHOD ON EEG DATA"

SUBMITTED IN THE PARTIAL FULFILMENT OF REQUIREMENT FOR THE AWARD OF DEGREE OF
BACHELOR OF ENGINEERING IN INFORMATION TEHNOLOGY

YEAR 2022-23

SUBMITTED BY

Ashwini Kumar Mishra

UNDER THE GUIDANCE OF

Mrs. Kavita Rawat
Project Guide,
IT Department,
UIT-BU, Bhopal

Dr. Rachna Kulhare
Project Coordinator
IT Department,
UIT-BU, Bhopal

Dr. Sunanda Manke
Project Guide and
Head of department,
IT Department, UIT-BU, Bhopal

UNIVERSITY INSTITUTE OF TECHNOLOGY BARKATULLAH UNIVERSITY, BHOPAL

Department of Information Technology

CERTIFICATE

This is to certify that the work embodied in this Major project report entitled "DIAGNOSIS AND PREDICTION OF ADHD DISEASE USING MACHINE LEARNING METHOD ON EEG DATA", submitted by **ASHWINI KUMAR MISHRA** to the Department of IT, UIT, Barkatullah University, Bhopal is a bonafide work carried out by the candidate during the Session 2022-2023 under the guidance of undersigned. It is further certified that the work reported in the report partially fulfils the requirement for the award of the degree of Bachelor of Engineering in Information Technology from the University Institute of Technology, Barkatullah University, Bhopal.

Mrs. Kavita Rawat
Project Guide,
IT Department,
UIT-BU, Bhopal

Dr. Rachna Kulhare
Project Coordinator
IT Department,
UIT-BU, Bhopal

Dr. Sunanda Manke
Project Guide and
Head of department,
IT Department, UIT-BU, Bhopal

Prof. N.K Gaur
Director,
UIT, B.U. Bhopal

UNIVERSITY INSTITUTE OF TECHNOLOGY BARKATULLAH UNIVERSITY, BHOPAL

Department of IT

DECLARATION

We hereby declare that the work is being presented in the Major Project entitled "DIAGNOSIS AND PREDICTION OF ADHD DISEASE USING MACHINE LEARNING METHOD ON EEG DATA" in partial fulfilment of the requirement of the degree of Bachelor of Engineering in Information Technology, University Institute of Technology, BU, Bhopal is an authentic record of our work carried out in session 2022-2023 under the guidance of **Mrs. Kavita Rawat**, Project Guide, Department of IT, University Institute of Technology, BU, Bhopal. To the best of our knowledge, this is our original work and has not been submitted earlier for the award of any other degree, diploma or certificate.

Date: / /

Ashwini Kumar Mishra

ACKNOWLEDGEMENT

This gives us a great pleasure to express our deep sense of gratitude to our Project Guide **Mrs. Kavita Rawat** and **Dr. Sunanda Manke**, Head of Department of IT for guidance, suggestion, support, help and constructive criticisms throughout project work. Without his able guidance, it would not have been possible to complete the project in time.

We are greatly indebted to him for providing required facilities for completing the project work. We express our thanks to the Director, University Institute of Technology, Barkatullah University, Bhopal for extending his support and critical comments which improved the quality of this report. We wish to avail ourselves of this opportunity, express a sense of gratitude and love to our friends, our beloved parents and our family members for always cheering us for their moral support, strength, help and encouraging me with their wishes. Finally, to God, the father of all, we are thankful for the strength and blessings are given to us for the completion of this work and our studies. However, it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to them.

Ashwini Kumar Mishra
(R208237220005)

ABSTRACT

In this research work “DIAGNOSIS AND PREDICTION OF ADHD DISEASE USING MACHINE LEARNING METHOD ON EEG DATA” aims to develop a predictive model for Attention-Deficit/Hyperactivity Disorder (ADHD) using EEG (Electroencephalography) data taken from IEEE.

EEG signal refers to the electrical activity recorded from the brain using electrodes placed on the scalp. It represents the collective electrical behavior of brain cells, reflecting various cognitive and physiological processes. EEG signals are characterized by rhythmic patterns and oscillations, which can provide insights into brain states, neural activity, and mental processes. And with the help from EEG signals it will be easy to diagnose a child from ADHD. The study involved machine learning algorithms, namely Logistic Regression, K-Neighbors Classifier, Gaussian Naive Bayes, Decision Tree, and Random Forest Classifier, for a predictive task. Each algorithm was then implemented, trained, and evaluated using appropriate metrics. The experimental results revealed that the K-Neighbors Classifier outperformed the other algorithms in terms of accuracy because KNN performs well when the data has a clear separation between classes or when the decision boundary is relatively simple.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Attention deficit hyperactivity disorder (ADHD) is a disease group related to various dysfunctions. There are two different words in the name of this disorder, and each of them is responsible for a specific symptom of ADHD.

The first group of patients with attention deficit can rarely focus on a unique topic, and anything surrounding them in their environment can distract them quickly.

The second group is related to hyperactivity symptoms, where they can tolerate a static situation and intend so much to alter their posture and position, in some cases, they may shout or keep silence suddenly in a regular discussion. All these conditions can happen for children in different stages of growth.

Based on the Centre's for Disease Control and Prevention (CDC) statistics in 2016, over 6.1 million children in the age group of 4 to 17 years have been diagnosed with ADHD. Overall, patients can be categorised into three different stages of ages-

- The children in a range of 2–5 years are 388 thousand,
- The children in the age of 6– 11 are 4 million,
- The rest are in the range of 12–17 years old.

According to the National Institute of Mental Health (NIMH), around 70% of children with ADHD continue to have some hyperactivity and impulsivity during their adolescence and adulthood. Long-term follow-up studies from childhood to adulthood revealed that children with ADHD, compared with those without ADHD, were more involved in psychosocial, educational, and neuropsychological functioning and had higher risks for antisocial disorders, major depression, and anxiety disorders as adults.

With the development of medical technologies and the application of various analytical tools, research to extract the brain’s functionality during ADHD, its electrical patterns and connectivity are on the rise. In recent years, many researchers have focused their attention on using different neuroimaging modalities and biomedical signal processing techniques to determine the behaviour and origin of ADHD as early as possible. Although finding out the mechanism and causality seems slow, the recognisance techniques are improving fast, but it should be mentioned that some clinical assessments such as The Diagnostic and Statistical

Manual of Mental Disorders (DSM), and International Classification of Diseases (ICD) were introduced. Nevertheless, the diagnosis procedure is time-consuming and subjective therefore, the results can easily vary from one case to another.

Early diagnosis of this disorder is of the prime importance in preventing subsequent complications such as negative effects on children's social interactions. Usually, the diagnosis of ADHD is done based on diagnostic judgment using criteria of different editions of DSM (Diagnostic and Statistical Manual of Mental Disorders) or ICD (International Classification of Diseases), which are highly dependent on the parents and teachers understanding of the psychologists or psychiatrist's questions and honesty in their responses. To reduce these problems, several studies attempted to propose and use more objective methods such as EEG in the diagnosis of ADHD.

LITERATURE SURVEY

1. The author Mohammadi et al has proposed Article called “EEG classification of ADHD and normal children using non-linear features and neural network” in June 2016. He had used Preprocessing, nonlinear feature extraction (fractal dimension, LLE, ApEn), mRMR, and neural networks for the predictions. Data selected were 60 subjects (30 ADHD, 30 control) child and the result was having 93.65% accuracy.

2. The author Tenev et al has proposed Article called “Machine learning approach for classification of ADHD adults” in July 2014. He had used SVM and voting for the predictions. Data selected were 117 subjects (67 ADHD, 50 control) child and the result was having 82.30% accuracy.

3. The author Tosun has proposed Article called “Effects of spectral features of EEG signals recorded with different channels and recording statuses on ADHD classification with deep learning” in 2021. He had used Data augmentation, PSD, SE, and LSTM for the predictions. Data selected were 160 subject’s child and the result was having 92.15% accuracy.

4. The author Khoshnoud et al has proposed Article in 2015. He had used Preprocessing, LLE, ApEn, PNN networks for the predictions. Data selected were 22 subjects (12 ADHD, 10 control) child and the result was having 87.50% accuracy.

5. The author Chen et al has proposed Article called “A deep learning framework for identifying children with ADHD using an EEG-based brain network” in 2019. He had used EEG signal to image conversion, CNN for the predictions. Data selected were 101 subjects (51 ADHD, 50 control) child and the result was having 94.67% accuracy.

6. The author Saini et al has proposed Article in 2022. He had used PCA and KNN for the predictions. Data selected were 157 subjects (77 ADHD, 80 control) child and the result was having 86% accuracy.

7. The author Tor et al has proposed Article called “Automated detection of conduct disorder and attention deficit hyperactivity disorder using decomposition and nonlinear techniques with EEG signals” in June 2021. He had used Empirical mode decomposition, Discrete wavelet transform, KNN for the predictions. Data selected were 123 subjects (45 ADHD, 78 control) child and the result was having 97.88% accuracy.

8. The author Dubreuil-Vall et al has proposed Article called “Deep Learning Convolutional Neural Networks Discriminate Adult ADHD From Healthy Individuals on the Basis of Event-Related Spectral EEG” in June 2020. He had used Preprocessing, spectrogram conversion and CNN for the predictions. Data selected were 40 subjects (20 ADHD, 20 control) child and the result was having 88% accuracy.

OBJECTIVE

Following are the objectives of this research project-

- The main aim of our work is to classify ADHD and control patient from EEG data using machine learning. This system takes the value of electrodes and generates final output as a prediction of ADHD.
- It removes the dependencies on the doctors.
- It helps out the poor and helpless people with the normal medical checkup.
- It helps people avoid paying huge amount to the doctors unnecessarily.
- It extends the role of the technology in the medical field.

REQUIREMENTS

Hardware Requirements:

- **Hardware**: Processor Intel dual-core and above
- **Clock speed**: 3.0 GHz
- **RAM size**: 4 GB
- **Hard Disk Capacity**: 400 GB
- **Internet connection**: Required

Software Requirements:

- **Operating System**: Windows 7 and Above
- **Browser**: Google chrome latest version
- **Database**: SQLite or any other major database software
- **Documentation**: MS-Office
- **IDE:** Google Colab
- **Interpreter**: Python 3.4 or newer
- **Version-Control Management**: GIT and GitHub

TECHNICAL DESCRIPTION

GOOGLE COLAB:

Google Colab, short for Google Colaboratory, is an innovative cloud-based platform that has revolutionised the way data scientists, researchers, and developers collaborate on machine learning projects. Developed by Google, Colab provides a Jupyter Notebook environment that enables users to write and execute code, perform data analysis, and create machine learning models using popular libraries like TensorFlow, PyTorch, and scikit-learn. With its powerful features and seamless integration with other Google services, Google Colab has become a preferred choice for many in the data science community.

Features and Functionality:

- Cloud-Based Environment: Google Colab offers a cloud-based development environment, eliminating the need for local installations of software and dependencies. Users can access Colab notebooks through a web browser, making it convenient for collaboration and ensuring consistent environments across team members.

- Jupyter Notebook Integration: Google Colab is built on the Jupyter Notebook infrastructure, providing an interactive and flexible interface for data exploration, code development, and documentation. Users can create, edit, and run notebooks that combine code, text, equations, and visualisations, fostering reproducibility and sharing of research findings.

- Free GPU and TPU Support: One of the standout features of Google Colab is its provision of free Graphics Processing Unit (GPU) and Tensor Processing Unit (TPU) resources. Users can leverage the power of these accelerators to train deep learning models faster, enabling rapid experimentation and prototyping without the need for expensive hardware.

- Easy Collaboration and Sharing: With Google Colab, collaboration is effortless. Multiple users can work on the same notebook simultaneously, making it ideal for team projects. The platform supports real-time editing, comments, and discussions, enhancing teamwork and knowledge exchange. Colab notebooks can also be easily shared with others, either as view-only or editable, allowing for seamless collaboration within research communities.

- Preinstalled Libraries and Packages: Google Colab comes with a wide range of preinstalled libraries and packages commonly used in data science and machine learning.

This includes popular libraries like NumPy, Pandas, Matplotlib, and SciPy, as well as deep learning frameworks such as TensorFlow and PyTorch. The preinstalled packages save time and effort, enabling users to start working on their projects right away.

- Integration with Google Drive: Colab integrates seamlessly with Google Drive, allowing users to save and load datasets, models, and other files directly from their Google Drive accounts. This feature simplifies data management and ensures easy access to project resources across devices.

- External Data Sources: Google Colab provides easy access to external data sources, such as BigQuery, Google Cloud Storage, and GitHub repositories. This enables users to import and analyse large datasets without having to download them locally, reducing storage constraints and enhancing productivity.

- Interactive Visualisations: Colab supports interactive visualisations with libraries like Plotly and Bokeh. Users can create interactive charts, graphs, and dashboards directly within their notebooks, facilitating data exploration and presentation.

- Code Snippets and Examples: Google Colab offers a wide range of code snippets and example notebooks contributed by the community. These resources serve as valuable references for beginners and experienced practitioners, providing ready-to-use code for various tasks, from data preprocessing to advanced machine learning techniques.

PYTHON:

Python, a versatile and dynamic programming language, has become the language of choice for many machine learning practitioners and researchers. Its simplicity, readability, and vast ecosystem of libraries have made it the go-to language for developing and implementing machine learning algorithms and models. In this article, we will explore how Python plays a crucial role in contrast with machine learning, highlighting its strengths and benefits in this field.

- Ease of Use and Readability: Python's syntax is designed to be clean and readable, making it easier for developers to write and understand code. This characteristic is particularly advantageous in the context of machine learning, where complex algorithms and mathematical concepts are involved. Python's simplicity allows researchers and practitioners to focus more on the machine learning concepts rather than getting lost in intricate programming details.

- Extensive Libraries and Packages: Python's popularity within the machine learning community is primarily attributed to its rich ecosystem of libraries and packages. Libraries such as NumPy, Pandas, and Matplotlib provide powerful tools for data manipulation, analysis, and visualization, enabling efficient preprocessing and exploration of datasets. Additionally, libraries like scikit-learn, TensorFlow, and PyTorch offer a wide range of pre-implemented machine learning algorithms and frameworks, simplifying the implementation of complex models.

- Flexibility and Versatility: Python's versatility makes it an ideal choice for machine learning tasks. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming, allowing developers to choose the most suitable approach for their projects. Python's flexibility also enables seamless integration with other languages and platforms, facilitating the utilisation of specialised libraries or leveraging existing code bases.

- Strong Community Support: Python boasts a vibrant and supportive community of developers and researchers, particularly in the field of machine learning. The open-source nature of Python encourages collaboration, knowledge sharing, and the development of cutting-edge tools and frameworks. The community-driven development model ensures a constant stream of updates, bug fixes, and enhancements, making Python a reliable and up-to-date choice for machine learning projects.

- Rapid Prototyping and Experimentation: Python's ease of use and extensive libraries enable rapid prototyping and experimentation in machine learning. With Python, researchers can quickly implement and test new ideas, algorithms, and models. The availability of prebuilt modules and libraries accelerates the development cycle, allowing practitioners to iterate on their solutions efficiently. This agility is essential in the iterative process of refining and improving machine learning models.

- Integration with Other Technologies: Python's compatibility and integration capabilities extend beyond the machine learning realm. It seamlessly integrates with databases, web frameworks, cloud services, and visualisation tools. This integration enables machine learning models to be deployed as part of larger systems, such as web applications or data pipelines. Python's versatility ensures that machine learning algorithms can be seamlessly incorporated into a broader technological ecosystem.

- Education and Learning Resources: Python's simplicity and popularity have resulted in a wealth of educational resources and learning materials. Numerous online tutorials, courses, and books specifically focus on teaching machine learning using Python. These resources

cater to learners of all levels, making it easier for beginners to get started with machine learning using Python and enabling experienced practitioners to deepen their understanding and expand their skill set.

PICKLE:

Pickle is a module in Python's standard library that enables the serialization and deserialisation of Python objects. It allows objects to be converted into a byte stream, which can be stored in a file, transferred over a network, or shared between different Python processes. Pickle can serialize various data types, including simple built-in types, complex custom objects, and even functions.

Functionality and Usage:

- Object Serialisation: The primary purpose of Pickle is to serialise Python objects into a compact byte stream. The process of serialisation involves converting the object's state (data and attributes) into a series of bytes that can be easily stored or transmitted. Pickle uses a binary format to represent the serialised objects, making it efficient and compact.

- Object Deserialisation: Deserialisation is the reverse process of serialisation. With Pickle, serialised objects can be deserialised, i.e., the byte stream can be converted back into the original object with its original state. Deserialisation allows stored or transmitted objects to be reconstructed and utilised in the Python environment.

- Serialisation of Complex Objects: Pickle is not limited to simple data types; it can handle complex objects as well. This includes custom classes, instances, and objects that contain other objects as attributes. Pickle maintains the relationships and references between objects during serialisation and deserialisation, ensuring their integrity is preserved.

- File Persistence: Pickle provides a convenient way to save and load Python objects from files. Using the dump() function, objects can be serialised and saved to a file, while the load() function allows the retrieval and deserialisation of objects from the file. This capability is useful for persisting objects across multiple program executions or for sharing objects between different Python programs.

- Inter-Process Communication: Pickle facilitates inter-process communication by allowing objects to be serialised and transmitted between different Python processes. Objects can be serialised with the dumps() function to obtain a byte stream, which can then be transmitted over a network or shared using inter-process communication mechanisms like

pipes or sockets. The receiving process can then deserialise the byte stream to reconstruct the original object.

Best Practices and Considerations:

- Security Risks: Pickle can execute arbitrary code during deserialisation, making it potentially vulnerable to security risks if used with untrusted data sources. It is essential to exercise caution when deserialising Pickle data from untrusted or unknown sources to prevent code injection or other malicious activities. Consider using alternative serialisation libraries or implementing data validation mechanisms if security is a concern.

- Versioning and Compatibility: Pickle is version-specific, and deserialising objects from a different Python version may lead to compatibility issues. It is crucial to ensure that the Python version used for serialisation and deserialisation is compatible. Additionally, changes to the object's structure or code between different versions may result in deserialisation errors. Care should be taken to maintain backward compatibility or handle versioning explicitly.

- Large Objects and Performance: Serialising and deserialising large objects can have performance implications, as the entire object and its state need to be processed. Consider the trade-off between the convenience of Pickle and the performance impact for large datasets. For scenarios involving large objects or frequent serialisation, alternative serialisation formats, such as JSON or protocol buffers, may offer better performance.

TECHNOLOGY USED

MACHINE LEARNING:

Machine learning is a rapidly evolving field of study that focuses on the development of algorithms and models that enable computers to learn and make predictions or decisions without being explicitly programmed. It is a subset of artificial intelligence (AI) and has found applications in a wide range of industries, including healthcare, finance, transportation, and more. In this article, we will explore the fundamentals of machine learning, its main techniques, and its impact on various domains.

Machine learning is based on the idea that computers can learn from data and improve their performance over time without being explicitly programmed for every task. It involves the development and application of algorithms that can automatically learn patterns and relationships from data, enabling computers to make informed decisions or predictions.

Types of Machine Learning:

- Supervised Learning: In supervised learning, the algorithm learns from labeled data, where each input data point is associated with a corresponding target output. The algorithm learns to map inputs to outputs by finding patterns in the labeled examples. It can then make predictions on unseen data based on its learned knowledge.

- Unsupervised Learning: Unsupervised learning deals with unlabeled data, where the algorithm's objective is to find underlying patterns, structures, or relationships within the data. Clustering and dimensionality reduction are common tasks in unsupervised learning. By discovering patterns, the algorithm can uncover hidden insights or group similar data points together.

- Reinforcement Learning: Reinforcement learning involves an agent learning to interact with an environment to maximize a reward signal. The agent takes actions, and based on the feedback from the environment, it learns to make better decisions over time. Reinforcement learning has been successful in applications such as game playing, robotics, and autonomous systems.

LOGISTIC REGRESSION:

Logistic regression is a statistical algorithm used for binary classification, which predicts the probability of an event occurring based on input features. It is a supervised learning technique and is widely used in various fields, including medicine, finance, marketing, and social

sciences. Despite its name, logistic regression is a classification algorithm rather than a regression algorithm.

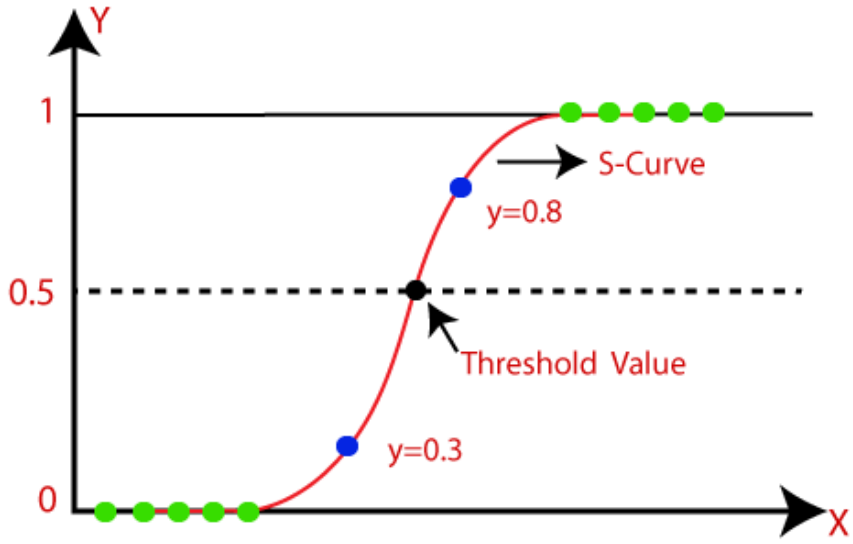

Fig.1 Logistic Regression

Working Principle:

Logistic regression models the relationship between the input features (predictor variables) and the probability of a binary outcome (dependent variable). The algorithm assumes a linear relationship between the features and the log-odds of the event occurring. The log-odds are then transformed using the logistic function, also known as the sigmoid function, to obtain the predicted probability.

The logistic function, represented by the equation:

p = 1 / (1 + e^(-z))

where p is the predicted probability, and z is the linear combination of the input features and their corresponding weights. The logistic function maps the log-odds to a value between 0 and 1, representing the probability of the positive class (event occurrence).

Model Training:

The logistic regression model is trained using a process called maximum likelihood estimation. The goal is to find the optimal weights that maximise the likelihood of the observed data. This is typically done by minimising the negative log-likelihood or using optimisation algorithms such as gradient descent.

During training, the algorithm iteratively adjusts the weights to minimise the difference between the predicted probabilities and the actual class labels. This process is guided by a cost function, such as the binary cross-entropy loss, which penalises large errors in prediction.

Model Interpretation:

One of the advantages of logistic regression is its interpretability. After training, the model provides estimates for the weights associated with each input feature. These weights represent the influence of the corresponding feature on the predicted probability. Positive weights indicate a positive association with the positive class, while negative weights indicate a negative association.

In addition to the weights, logistic regression models can provide other valuable information such as odds ratios and confidence intervals. Odds ratios quantify the change in the odds of the event occurring with a one-unit increase in the corresponding feature, allowing for meaningful comparisons between different features.

Extensions and Variations:

Logistic regression can be extended to handle multi-class classification problems through techniques like one-vs-rest or multinomial logistic regression. One-vs-rest creates multiple binary logistic regression models, each predicting one class against the rest. Multinomial logistic regression, on the other hand, directly models the probabilities of multiple classes using the SoftMax function.

Regularisation techniques, such as L1 or L2 regularisation, can also be applied to logistic regression to prevent overfitting and improve generalisation performance. Regularisation adds a penalty term to the cost function, discouraging large weights and promoting a more parsimonious model.

Limitations and Considerations:

While logistic regression is a versatile and widely used algorithm, it has certain limitations. It assumes a linear relationship between the features and the log-odds, which may not hold in complex datasets. Feature engineering and transformations may be necessary to capture non-linear relationships.

Additionally, logistic regression is sensitive to outliers and may not perform well in the presence of collinear features. It also assumes that the observations are independent, and the input features are unrelated to each other.

K-NEIGHBORS CLASSIFIER:

The K-Neighbors Classifier is a popular algorithm for classification tasks in machine learning. It is a type of instance-based or lazy learning algorithm, where the training data points are stored and used during the prediction phase. The algorithm classifies new instances by finding the K nearest neighbors in the training dataset and assigning the class label based on the majority vote of those neighbors.

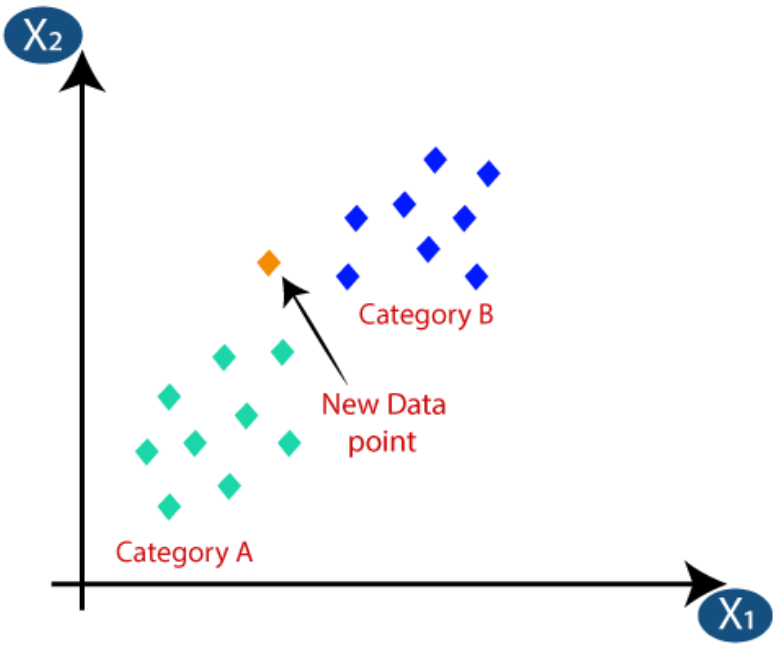

Fig.2 K-Neighbors Classifier

Working Principle:

The K-Neighbors Classifier algorithm classifies data points based on their proximity to other data points in the feature space. When a new instance is presented for prediction, the algorithm calculates the distances between the new instance and all the training instances. The distances are typically calculated using measures such as Euclidean distance, Manhattan distance, or other distance metrics.

The K parameter in K-Neighbors Classifier represents the number of neighbors to consider for the classification. It is an important parameter that needs to be carefully chosen based on the characteristics of the dataset. The larger the value of K, the smoother the decision boundaries become, but there is a trade-off in terms of increased computational complexity and potential loss of local details.

Prediction and Decision Rule:

Once the distances are calculated, the K-Neighbors Classifier algorithm selects the K nearest neighbors to the new instance. The class label for the new instance is determined by taking a majority vote among the class labels of the neighbors. In other words, the predicted class label is the class that occurs most frequently among the K nearest neighbors.

In cases where K is an even number, a common practice is to break ties by selecting the class label with the highest-class probability or by using a weighted voting scheme, where closer neighbors have a higher influence on the decision.

Model Training:
As a lazy learning algorithm, K-Neighbors Classifier does not involve an explicit training phase. Instead, it stores the training instances in memory to be used during the prediction phase. This characteristic allows the algorithm to adapt to new training data without the need for retraining the model.

During the prediction phase, the algorithm simply searches for the K nearest neighbors among the training instances based on the distance metric chosen. Therefore, the computational cost of the K-Neighbors Classifier algorithm depends on the number of training instances and the dimensionality of the feature space.

Considerations and Parameters:
There are several important considerations when using the K-Neighbors Classifier algorithm:

- Choice of K: The choice of K should be based on the nature of the problem and the characteristics of the dataset. A small value of K may lead to overfitting and sensitivity to noise, while a large value of K may lead to under fitting and loss of local details.

- Distance Metric: The choice of distance metric can have an impact on the algorithm's performance. The most common distance metrics are Euclidean distance and Manhattan distance, but other metrics can be used depending on the problem at hand.

- Data Pre-processing: K-Neighbors Classifier can be sensitive to the scale and distribution of the input features. It is often beneficial to normalize or standardise the feature values to ensure that no single feature dominates the distance calculations.

- Handling Imbalanced Classes: If the classes in the dataset are imbalanced, where some classes have significantly more instances than others, it may be necessary to balance the class distribution or adjust the weights during the majority voting process.

GAUSSIAN NAIVE BAYES:

Gaussian Naive Bayes, is a popular algorithm for classification tasks, particularly in situations where the features are continuous and assumed to have a Gaussian (normal) distribution. It is a variant of the Naive Bayes algorithm and is based on Bayes' theorem and the assumption of feature independence.

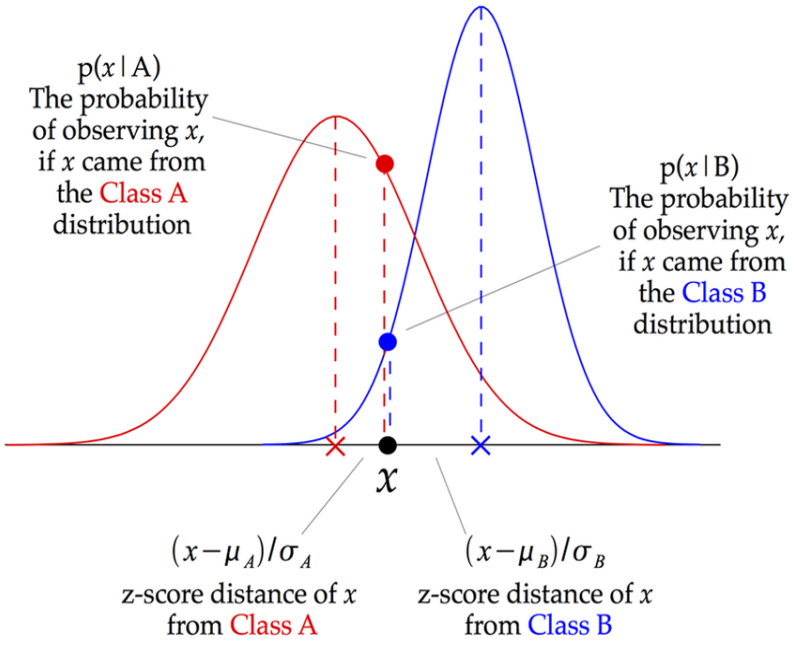

Fig.3 Gaussian Naive Bayes

Working Principle:

The Gaussian Naive Bayes algorithm is based on the principle of Bayes' theorem, which provides a way to calculate the probability of a certain event occurring based on prior knowledge. In the context of classification, Bayes' theorem calculates the probability of a class label given the observed features.

Gaussian Naive Bayes assumes that the features follow a Gaussian distribution within each class. This means that the algorithm models the probability density function (PDF) of each feature for Each class using a Gaussian distribution. It estimates the mean and standard deviation of each feature for each class from the training data.

Prediction and Decision Rule:

During the prediction phase, Gaussian Naive Bayes calculates the conditional probability of each class label given the observed features. It uses Bayes' theorem to combine the prior probabilities of the classes with the likelihoods of the observed features given each class.

The decision rule of Gaussian Naive Bayes is to assign the class label with the highest posterior probability. In other words, it selects the class that maximizes the product of the prior probability and the likelihood. This is equivalent to selecting the class that has the highest joint probability of the observed features and the class label.

Model Training:

The training phase of Gaussian Naive Bayes involves estimating the parameters of the Gaussian distribution for each feature in each class. This estimation is performed by calculating the mean and standard deviation of each feature within each class using the training data.

When a new instance is presented for prediction, Gaussian Naive Bayes uses the estimated parameters to calculate the likelihood of the observed features given each class. It then combines this likelihood with the prior probabilities of the classes to compute the posterior probabilities, which are used for classification.

Considerations and Assumptions:

- Feature Independence: Gaussian Naive Bayes assumes that the features are conditionally independent given the class label. This is known as the "naive" assumption, which simplifies the computation by considering each feature's contribution to the class probability independently. While this assumption may not hold in all cases, Gaussian Naive Bayes often performs well even when the independence assumption is violated.

- Gaussian Distribution: Gaussian Naive Bayes assumes that the features within each class follow a Gaussian distribution. If the features do not exhibit a Gaussian distribution, the performance of Gaussian Naive Bayes may be affected. In such cases, feature transformations or alternative algorithms may be more suitable.

- Handling Zero Variance: If a feature has zero variance within a class, it can cause issues during the calculation of probabilities in Gaussian Naive Bayes, as the denominator of the Gaussian distribution becomes zero. To handle this, techniques such as smoothing or adding a small constant to the variance can be employed.

- Handling Missing Values: Gaussian Naive Bayes does not naturally handle missing values in the features. Various imputation techniques can be applied to handle missing values before training the model.

DECISION TREE CLASSIFIER:

The Decision Tree Classifier is a popular algorithm for classification tasks in machine learning. It builds a decision tree model that makes predictions by recursively splitting the input features based on their values and the associated class labels.

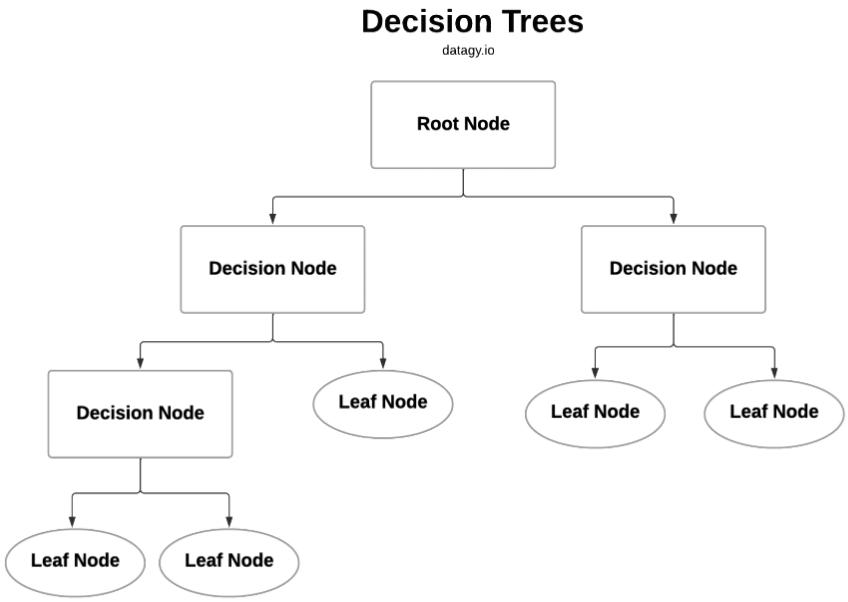

Fig.4 Decision Tree

Working Principle:

The Decision Tree Classifier algorithm creates a binary tree structure where each internal node represents a decision based on a specific feature and its threshold value. The tree's leaves represent the class labels or the decision outcome.

The algorithm selects the best feature and threshold value at each internal node based on a criterion such as information gain or Gini impurity. These criteria measure the homogeneity or impurity of the class labels within each branch of the tree. The goal is to find the feature and threshold value that maximize the separation between different classes.

Prediction and Decision Rule:

During the prediction phase, the Decision Tree Classifier algorithm follows the path down the tree based on the feature values of the input instance. At each internal node, it compares the feature value to the threshold value and proceeds down the left or right child node based on the comparison result. This process continues until a leaf node is reached.

The class label associated with the leaf node is then assigned as the predicted class label for the input instance.

Model Training:

The training phase of the Decision Tree Classifier algorithm involves growing the decision tree by recursively splitting the input features based on their values. This process starts with the root node, which contains all the training instances.

The algorithm selects the best feature and threshold value to split the current node, aiming to maximize the separation between classes. It measures the quality of a split using a criterion such as information gain or Gini impurity. The chosen feature and threshold value become the decision rule for the current node.

The splitting process continues recursively for each child node until a stopping criterion is met, such as reaching a maximum tree depth or a minimum number of instances in a leaf node.

Considerations and Parameters:

- Overfitting: Decision trees have a tendency to overfit the training data, meaning they may learn and memorize specific instances instead of generalizing patterns. This can lead to poor performance on unseen data. Regularization techniques such as pruning, limiting the tree depth, or setting a minimum number of instances in a leaf node can help combat overfitting.

- Feature Importance: Decision trees provide a measure of feature importance based on the number of times a feature is used for splitting and the improvement in impurity achieved by the splits. This information can be useful for feature selection or gaining insights into the dataset.

- Handling Categorical Features: By default, Decision Tree Classifier can handle both categorical and numerical features. However, categorical features need to be encoded or converted to numerical values before training the model.

- Handling Missing Values: Decision Tree Classifier can handle missing values by considering them as a separate category or by imputing the missing values before training the model.

RANDOM FOREST CLASSIFIER:

The Random Forest Classifier is an ensemble learning algorithm that combines multiple decision trees to make predictions. It is a powerful and widely used algorithm for classification tasks, known for its robustness and ability to handle complex datasets.

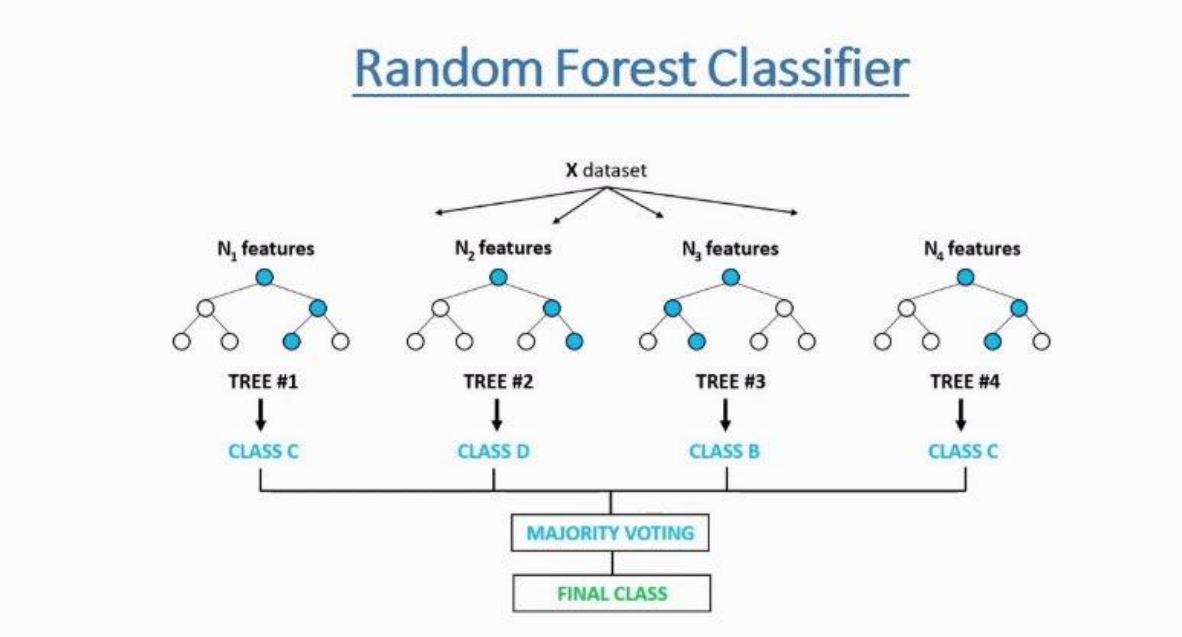

Fig.5 Random Forest Classifier

Working Principle:

The Random Forest Classifier algorithm builds an ensemble of decision trees, where each tree is trained on a random subset of the training data and a random subset of the input features. This process is known as bootstrap aggregating or bagging.

During the training phase, the algorithm creates a user-specified number of decision trees. Each tree is trained independently using a different subset of the training data, sampled with replacement. This means that each tree has the potential to see different instances and may learn different patterns.

Prediction and Decision Rule:

During the prediction phase, the Random Forest Classifier combines the predictions of all the individual decision trees to make the final prediction. For classification tasks, it uses the majority vote of the decision trees, where the class that receives the most votes is selected as the predicted class label.

The decision rule of the Random Forest Classifier is based on the aggregation of predictions from multiple trees. Each tree in the forest independently makes its prediction, and the final prediction is determined by the majority vote or the weighted average of the individual tree predictions.

Model Training:

The training phase of the Random Forest Classifier algorithm involves creating a forest of decision trees. Each tree is grown by recursively splitting the training data based on random subsets of the input features.

For each tree, the algorithm selects the best feature and threshold value at each internal node, similar to the Decision Tree Classifier. However, the selection process is limited to a random subset of features, rather than considering all features at each split. This randomness introduces diversity among the decision trees and helps reduce overfitting.

Considerations and Parameters:

- Number of Trees: The number of trees in the random forest is an important parameter to consider. Increasing the number of trees generally improves performance, but it also increases computational complexity.

- Feature Subset Size: Random Forest Classifier randomly selects a subset of features at each split. The size of this feature subset is another important parameter. A larger feature subset provides more diversity but may lead to increased correlation among the trees.

- Handling Imbalanced Classes: Random forests can handle imbalanced classes by adjusting the class weights during the training process or using sampling techniques such as oversampling or under sampling.

- Out-of-Bag Evaluation: An advantage of Random Forest Classifier is the ability to estimate the performance on unseen data without the need for a separate validation set. This is done using the out-of-bag samples, which are the instances not included in the bootstrap sample for each tree.

PROPOSED METHODOLOGY

This work focus on developing a model that identify attention deficit hyperactivity disorder (ADHD). For this input dataset is pass from the pre-processing step that cleans the data for finding the feature set. Feature identification is done by the dynamic approach with soft computing algorithms.

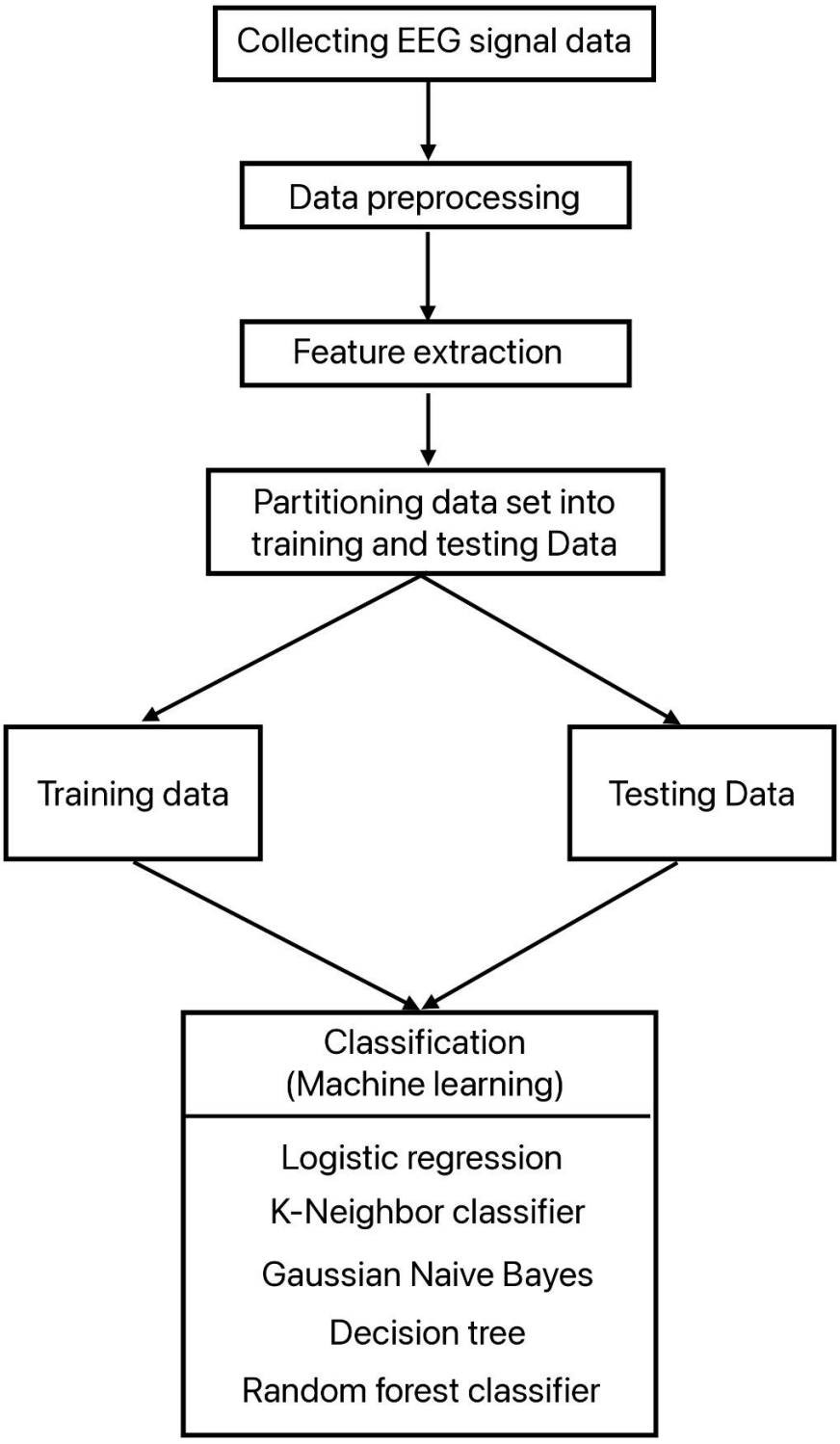

Fig. 6 Block Diagram

Dataset : ADHD dataset taken from IEEE.

```
[ ]  1  # Reading the ADHD data
     2  df_ADHD = pd.read_csv('/content/combined_ADHD.csv')
     3
     4  # # Print the first five row
     5  df_ADHD.head()
```

	Unnamed: 0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0	85	-407	200	191	420	457	310	310	16	1009	531	126	457	200	457	384	-90	473	121
1	1	-266	-55	-20	367	163	384	-20	310	494	1193	494	236	236	310	200	457	-195	543	15
2	2	-90	-19	126	437	420	568	347	457	-131	1156	384	384	494	384	494	531	-19	613	261
3	3	-90	-160	163	473	384	494	310	384	457	1340	494	420	310	420	273	531	-90	437	-19
4	4	-301	-336	-20	473	200	531	89	420	200	1156	310	494	273	457	236	568	-160	578	121

```
     1  # Print the dimension of data
     2  df_ADHD.shape
```

```
(63389, 20)
```

Fig. 7 ADHD Child data

```
[ ]  1  # Reading the Control data
     2  df_control = pd.read_csv('/content/combined_Control.csv')
     3
     4  # Print the first five row
     5  df_control.head()
```

	Unnamed: 0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	0	367.0	156.0	310.0	156.0	384.0	163.0	420.0	200.0	420.0	273.0	310.0	200.0	384.0	200.0	384.0	163.0	367.0	156.0	508.0
1	1	473.0	85.0	531.0	226.0	494.0	163.0	494.0	52.0	531.0	163.0	604.0	200.0	568.0	236.0	568.0	163.0	508.0	85.0	402.0
2	2	402.0	-19.0	531.0	85.0	457.0	-20.0	420.0	-241.0	347.0	-57.0	457.0	89.0	457.0	126.0	457.0	-57.0	332.0	15.0	226.0
3	3	332.0	50.0	457.0	191.0	384.0	-20.0	273.0	-388.0	310.0	-241.0	494.0	163.0	494.0	163.0	420.0	-131.0	473.0	50.0	15.0
4	4	437.0	121.0	568.0	261.0	494.0	52.0	384.0	-315.0	384.0	-57.0	494.0	126.0	568.0	200.0	568.0	-20.0	402.0	-19.0	85.0

```
[ ]  1  # Print the dimension of data
     2  df_control.shape
```

```
(41954, 20)
```

Fig. 8 Non-ADHD Child data

Pre-Processing : In this step of proposed model input dataset is transform into matrix format. Incomplete session is also removed in this step. For increasing the training efficiency of learning model type of session class in training dataset will be balance.

```
# Add a new column with a value of 0 in all rows of ADHD Data
df_ADHD['result'] = 0

# Write the updated DataFrame to the same CSV file
df_ADHD.to_csv('combined_ADHD.csv', index=False)

df_ADHD.head()
```

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	10	11	12	13	14	15	16	17	18	result
0	0	85	-407	200	191	420	457	310	310	16	...	531	126	457	200	457	384	-90	473	121	0
1	1	-266	-55	-20	367	163	384	-20	310	494	...	494	236	236	310	200	457	-195	543	15	0
2	2	-90	-19	126	437	420	568	347	457	-131	...	384	384	494	384	494	531	-19	613	261	0
3	3	-90	-160	163	473	384	494	310	384	457	...	494	420	310	420	273	531	-90	437	-19	0
4	4	-301	-336	-20	473	200	531	89	420	200	...	310	494	273	457	236	568	-160	578	121	0

5 rows × 21 columns

Fig. 9 ADHD Child data with result attribute having value '0'.

```
# Add a new column with a value of 1 in all rows Control healthy data
df_control['result'] = 1

# Write the updated DataFrame to the same CSV file
df_control.to_csv('combined_Control.csv', index=False)

df_control.head()
```

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	10	11	12	13	14	15	16	17	18	result
0	0	367.0	156.0	310.0	156.0	384.0	163.0	420.0	200.0	420.0	...	310.0	200.0	384.0	200.0	384.0	163.0	367.0	156.0	508.0	1
1	1	473.0	85.0	531.0	226.0	494.0	163.0	494.0	52.0	531.0	...	604.0	200.0	568.0	236.0	568.0	163.0	508.0	85.0	402.0	1
2	2	402.0	-19.0	531.0	85.0	457.0	-20.0	420.0	-241.0	347.0	...	457.0	89.0	457.0	126.0	457.0	-57.0	332.0	15.0	226.0	1
3	3	332.0	50.0	457.0	191.0	384.0	-20.0	273.0	-388.0	310.0	...	494.0	163.0	494.0	163.0	420.0	-131.0	473.0	50.0	15.0	1
4	4	437.0	121.0	568.0	261.0	494.0	52.0	384.0	-315.0	384.0	...	494.0	126.0	568.0	200.0	568.0	-20.0	402.0	-19.0	85.0	1

5 rows × 21 columns

Fig. 10 Non-ADHD Child data with result attribute having value '1'.

```
# Read in the data from both ADHD & Control Healthy CSV files
df7 = pd.read_csv('/content/combined_ADHD.csv')
df8 = pd.read_csv('/content/combined_Control.csv')

# Combine the DataFrames into a single DataFrame
combined_df_data = pd.concat([df7, df8])

# Shuffle the rows of the combined DataFrame
combined_df_data = combined_df_data.sample(frac=1, random_state=1234)

# Write the shuffled DataFrame to a new CSV file
combined_df_data.to_csv('combined_data.csv', index=False)
```

```
# Reading the Combine data
df = pd.read_csv('/content/combined_data.csv')

# Print the first ten row
df.head(10)
```

	Unnamed: 0	0	1	2	3	4	5	6	7	8	...	10	11	12	13	14	15	16	17	18	result
0	8059	-160.0	85.0	16.0	226.0	-20.0	347.0	52.0	310.0	89.0	...	-94.0	126.0	-57.0	163.0	-94.0	200.0	121.0	297.0	191.0	1
1	14168	613.0	437.0	163.0	261.0	-131.0	163.0	-167.0	200.0	-57.0	...	457.0	52.0	236.0	236.0	-20.0	347.0	-19.0	156.0	-55.0	0
2	6560	-583.0	437.0	-241.0	-55.0	-20.0	126.0	-278.0	52.0	-94.0	...	52.0	16.0	-57.0	126.0	52.0	126.0	-371.0	226.0	15.0	0
3	15506	85.0	50.0	16.0	-19.0	163.0	16.0	200.0	163.0	163.0	...	52.0	16.0	89.0	-20.0	89.0	89.0	-19.0	191.0	191.0	0
4	15750	367.0	85.0	310.0	-55.0	420.0	-315.0	236.0	-94.0	236.0	...	310.0	-57.0	347.0	-94.0	310.0	-57.0	261.0	-19.0	226.0	1
5	4454	121.0	156.0	89.0	121.0	126.0	-20.0	16.0	-131.0	163.0	...	273.0	494.0	273.0	236.0	163.0	-20.0	261.0	15.0	-19.0	1
6	21191	649.0	649.0	310.0	261.0	89.0	163.0	126.0	52.0	126.0	...	200.0	236.0	163.0	52.0	200.0	126.0	297.0	15.0	50.0	0
7	4031	-55.0	-125.0	16.0	-19.0	126.0	200.0	163.0	-131.0	126.0	...	-94.0	16.0	16.0	52.0	89.0	89.0	-90.0	191.0	85.0	1
8	410	-90.0	85.0	89.0	156.0	89.0	273.0	200.0	384.0	89.0	...	-131.0	310.0	-167.0	126.0	-20.0	236.0	-19.0	297.0	191.0	1
9	7522	-195.0	-19.0	-131.0	-125.0	-94.0	52.0	-131.0	52.0	-20.0	...	-20.0	52.0	-20.0	89.0	16.0	163.0	-125.0	121.0	-90.0	1

10 rows × 21 columns

Fig. 11 Combined ADHD Child data and Non-ADHD child data..

```
df = df.drop(['Unnamed: 0'], axis=1)
df.head(15)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	result
0	-160.0	85.0	16.0	226.0	-20.0	347.0	52.0	310.0	89.0	273.0	-94.0	126.0	-57.0	163.0	-94.0	200.0	121.0	297.0	191.0	1
1	613.0	437.0	163.0	261.0	-131.0	163.0	-167.0	200.0	-57.0	236.0	457.0	52.0	236.0	236.0	-20.0	347.0	-19.0	156.0	-55.0	0
2	-583.0	437.0	-241.0	-55.0	-20.0	126.0	-278.0	52.0	-94.0	457.0	52.0	16.0	-57.0	126.0	52.0	126.0	-371.0	226.0	15.0	0
3	85.0	50.0	16.0	-19.0	163.0	16.0	200.0	163.0	163.0	163.0	52.0	16.0	89.0	-20.0	89.0	89.0	-19.0	191.0	191.0	0
4	367.0	85.0	310.0	-55.0	420.0	-315.0	236.0	-94.0	236.0	-57.0	310.0	-57.0	347.0	-94.0	310.0	-57.0	261.0	-19.0	226.0	1
5	121.0	156.0	89.0	121.0	126.0	-20.0	16.0	-131.0	163.0	-241.0	273.0	494.0	273.0	236.0	163.0	-20.0	261.0	15.0	-19.0	1
6	649.0	649.0	310.0	261.0	89.0	163.0	126.0	52.0	126.0	52.0	200.0	236.0	163.0	52.0	200.0	126.0	297.0	15.0	50.0	0
7	-55.0	-125.0	16.0	-19.0	126.0	200.0	163.0	-131.0	126.0	163.0	-94.0	16.0	16.0	52.0	89.0	89.0	-90.0	191.0	85.0	1
8	-90.0	85.0	89.0	156.0	89.0	273.0	200.0	384.0	89.0	347.0	-131.0	310.0	-167.0	126.0	-20.0	236.0	-19.0	297.0	191.0	1
9	-195.0	-19.0	-131.0	-125.0	-94.0	52.0	-131.0	52.0	-20.0	200.0	-20.0	52.0	-20.0	89.0	16.0	163.0	-125.0	121.0	-90.0	1
10	191.0	226.0	384.0	332.0	310.0	273.0	273.0	273.0	163.0	236.0	273.0	236.0	200.0	200.0	200.0	163.0	297.0	297.0	226.0	0
11	-512.0	-336.0	16.0	-55.0	52.0	16.0	126.0	89.0	126.0	163.0	16.0	-57.0	163.0	16.0	200.0	163.0	-55.0	85.0	15.0	0
12	156.0	85.0	163.0	156.0	200.0	200.0	200.0	236.0	89.0	200.0	236.0	126.0	200.0	200.0	163.0	236.0	156.0	226.0	261.0	0
13	-195.0	965.0	-278.0	-90.0	-241.0	126.0	-351.0	126.0	-204.0	163.0	-167.0	200.0	-204.0	89.0	-278.0	163.0	-19.0	297.0	261.0	0
14	15.0	-55.0	-351.0	-301.0	-241.0	-20.0	-241.0	-167.0	-57.0	89.0	-94.0	16.0	-131.0	163.0	89.0	126.0	-442.0	-125.0	-160.0	0

```
df.shape
```

```
(105343, 20)
```

Fig. 12 Combined ADHD Child data and Non-ADHD child data (Dropped unnamed attribute)

Data Visualization : Data visualization is the graphical representation of information and data using visual elements such as charts, graphs, and maps. It aims to present complex data in a clear and intuitive manner, allowing viewers to understand patterns, trends, and relationships, facilitating better insights and decision-making.

Fig. 13 Pair Plot co-relation between attributes..

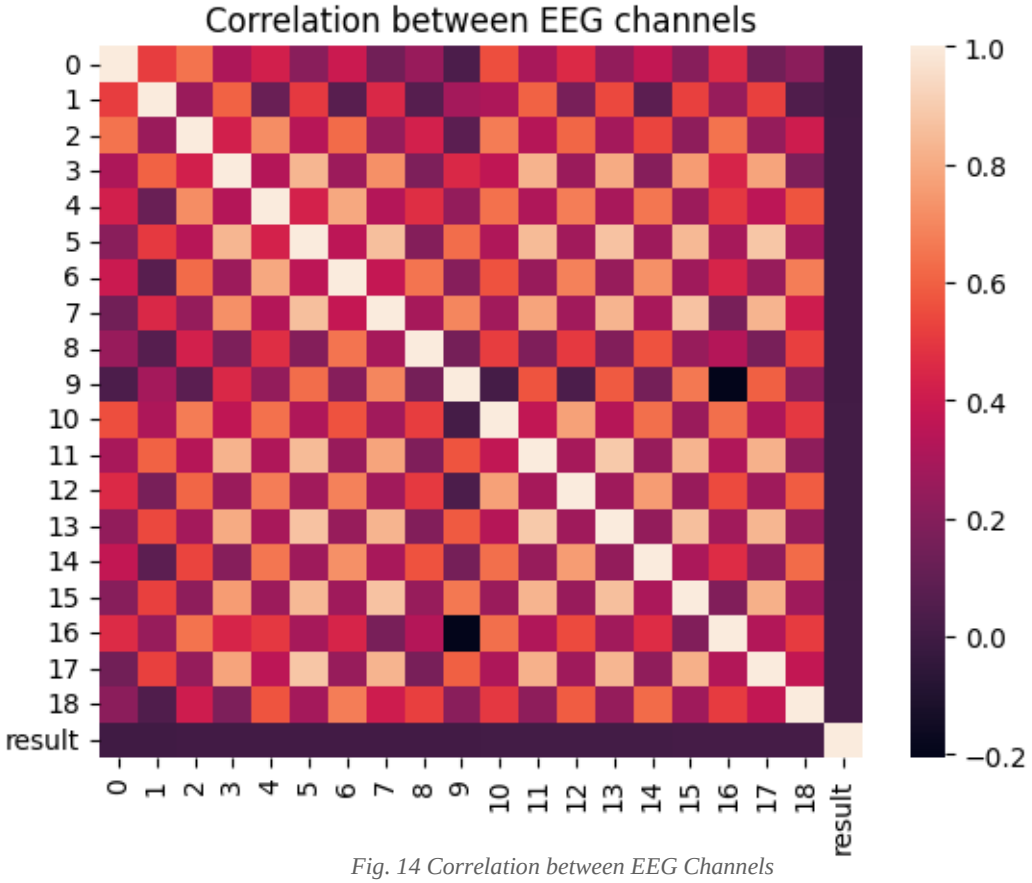

Fig. 14 Correlation between EEG Channels

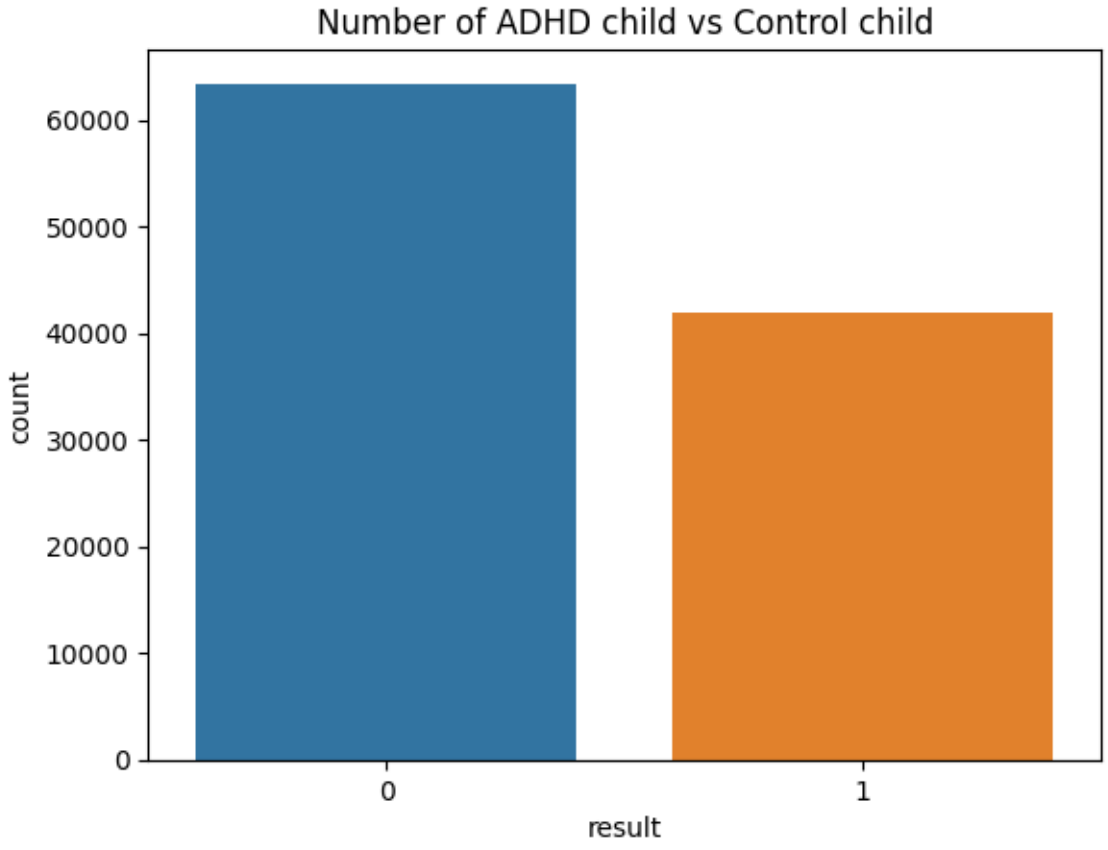

Fig. 15 Number of ADHD child vs Control child

Splitting Data : Splitting data into training and testing sets is a fundamental step in machine learning. It involves dividing the available data into two subsets: the training set, used to train the machine learning model, and the testing set, used to evaluate the model's performance and generalization ability on unseen data.

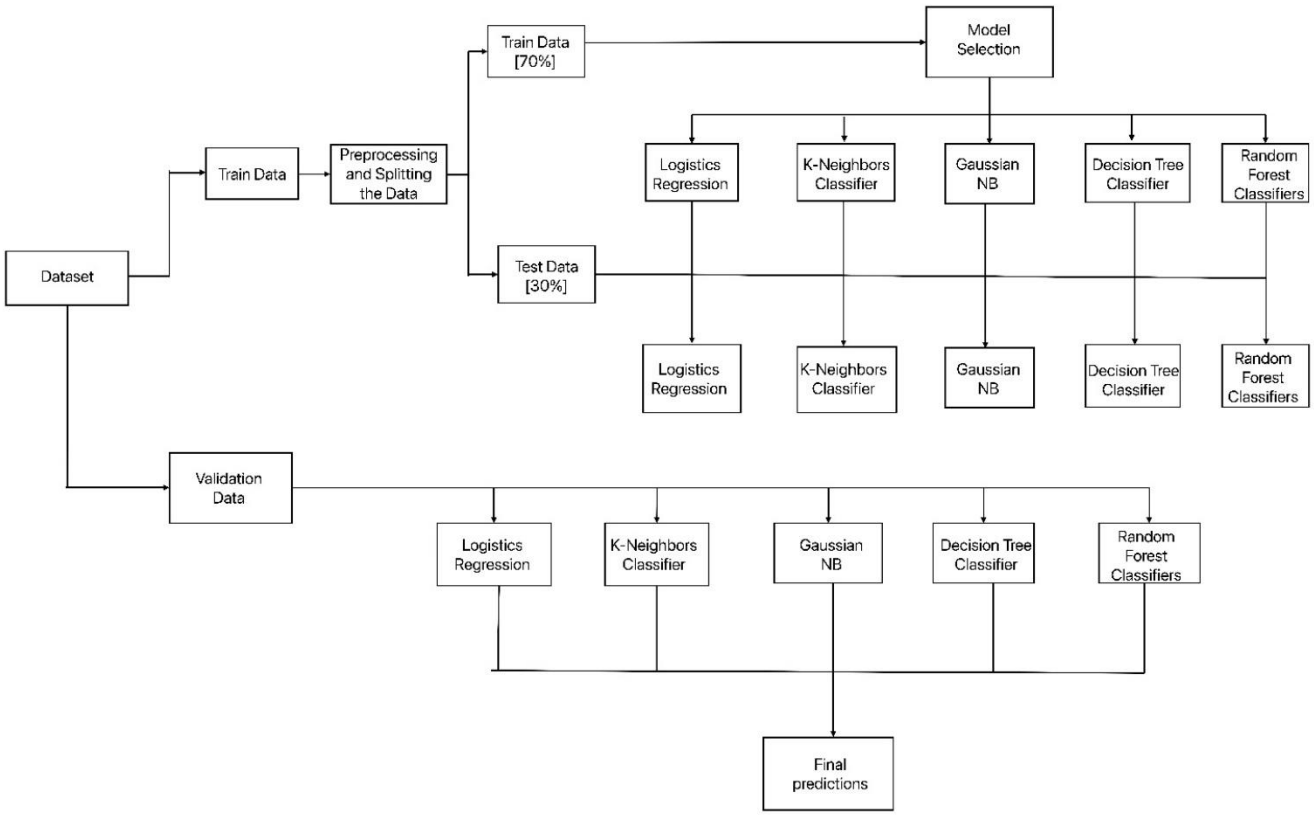

Fig. 16 Data flow Chart

```
[ ]  1  x = df.iloc[:, 0:-1]
     2  y = df.iloc[:, -1]
```

```
[ ]  1  # x_normalized = x / np.max(x)
     2  x.columns
```

```
Index(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12',
       '13', '14', '15', '16', '17', '18'],
      dtype='object')
```

```
     1  y=pd.DataFrame(y)
     2  y
```

	result
0	1
1	0
2	0
3	0
4	1
...	...
105338	1
105339	0
105340	0
105341	0
105342	1

105343 rows × 1 columns

```
[ ]  1  x.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
0	-160.0	85.0	16.0	226.0	-20.0	347.0	52.0	310.0	89.0	273.0	-94.0	126.0	-57.0	163.0	-94.0	200.0	121.0	297.0	191.0
1	613.0	437.0	163.0	261.0	-131.0	163.0	-167.0	200.0	-57.0	236.0	457.0	52.0	236.0	236.0	-20.0	347.0	-19.0	156.0	-55.0
2	-583.0	437.0	-241.0	-55.0	-20.0	126.0	-278.0	52.0	-94.0	457.0	52.0	16.0	-57.0	126.0	52.0	126.0	-371.0	226.0	15.0
3	85.0	50.0	16.0	-19.0	163.0	16.0	200.0	163.0	163.0	163.0	52.0	16.0	89.0	-20.0	89.0	89.0	-19.0	191.0	191.0
4	367.0	85.0	310.0	-55.0	420.0	-315.0	236.0	-94.0	236.0	-57.0	310.0	-57.0	347.0	-94.0	310.0	-57.0	261.0	-19.0	226.0

```
[ ]  1  y.head()
```

	result
0	1
1	0
2	0
3	0
4	1

```
[ ]  1  from sklearn.model_selection import train_test_split
     2  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=0)
```

```
     1  #Feature Scaling
     2  sc = StandardScaler()
     3  x_train = sc.fit_transform(x_train)
     4  x_test = sc.transform(x_test)
```

Fig. 17 Training and testing Data set and Feature scaling.

Model Development : Model development is an iterative process in which many models are derived, tested and built upon a model fitting the desired criteria is build.

```
[ ]  1  #Using Logistic Regression Algorithm to the Training Set
     2  from sklearn.linear_model import LogisticRegression
     3  lr_classifier = LogisticRegression(random_state = 0)
     4  lr_classifier.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143:
  y = column_or_1d(y, warn=True)
▾ LogisticRegression
LogisticRegression(random_state=0)
```

```
[ ]  1  y_pred = lr_classifier.predict(x_test)
     2  cm = confusion_matrix(y_test, y_pred)
     3  cm
```

```
array([[19035,     3],
       [12559,     6]])
```

```
[ ]  1  acs = accuracy_score(y_test, y_pred)
     2  acs*100
```

```
60.2506091193874
```

Fig. 18 Logistic Regression Model

```
1  #Using KNeighborsClassifier Method of neighbors class to use Nearest Neighbor algorithm
2  from sklearn.neighbors import KNeighborsClassifier
3  knn_classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
4  knn_classifier.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:215: DataConversionWarning:
  return self._fit(X, y)
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
[ ]  1  #confusion_matrix
     2  y_pred = knn_classifier.predict(x_test)
     3  cm = confusion_matrix(y_test, y_pred)
     4  cm
```

```
array([[18208,   830],
       [ 1694, 10871]])
```

```
[ ]  1  #accuracy_score
     2  acs = accuracy_score(y_test, y_pred)
     3  acs*100
```

```
92.01341644780558
```

Fig. 19 K-Neighbors Classifier Model

```
#Using GaussianNB method of naive_bayes class to use Naive Bayes Algorithm
from sklearn.naive_bayes import GaussianNB
gnb_classifier = GaussianNB()
gnb_classifier.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143: DataConversionWarning:
  y = column_or_1d(y, warn=True)
▾ GaussianNB
GaussianNB()
```

```
#confusion_matrix
y_pred = gnb_classifier.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[16790,  2248],
       [ 9319,  3246]])
```

```
#accuracy_score
acs = accuracy_score(y_test, y_pred)
acs*100
```

```
63.39904439451951
```

Fig. 20 Gaussian Naïve Bayes Model

```
#Using DecisionTreeClassifier of tree class to use Decision Tree Algorithm
from sklearn.tree import DecisionTreeClassifier
dt_classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
dt_classifier.fit(x_train, y_train)
```

```
▾                    DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

```
#confusion_matrix
y_pred = dt_classifier.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[16571,  2467],
       [ 2451, 10114]])
```

```
#accuracy_score
acs = accuracy_score(y_test, y_pred)
acs*100
```

```
84.438186248141
```

Fig. 21 Decision Tree Classifier Model

```
[ ]  1  #Using RandomForestClassifier method of ensemble class to use Random Forest Classification algorith
     2  from sklearn.ensemble import RandomForestClassifier
     3  rf_classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
     4  rf_classifier.fit(x_train, y_train)
```

```
<ipython-input-47-cc603be30f30>:4: DataConversionWarning: A column-vector y was passed when a 1d array was expected.
  rf_classifier.fit(x_train, y_train)
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

```
     1  #confusion_matrix
     2  y_pred = rf_classifier.predict(x_test)
     3  cm = confusion_matrix(y_test, y_pred)
     4  cm
```

```
array([[18296,   742],
       [ 2512, 10053]])
```

```
[ ]  1  #accuracy_score
     2  acs = accuracy_score(y_test, y_pred)
     3  acs*100
```

```
89.70350916052273
```

Fig. 22 Random Forest Classifier Model

RESULT

In this research work, the data set contained information worth 105343 rows and 20 columns. And the training and testing size were 70% and 30% respectively.

This project uses machine learning algorithms namely Logistic Regression, K-Neighbors Classifier, Gaussian Naïve Bayes, Decision Tree Classifier, Random Forest Classifier, having the accuracy 60.25%, 92.01%, 63.39%, 84.43% and 89.70% respectively. Among all those algorithms, K-Neighbors Classifier had a highest accuracy with 92.01%.

Comparative Study of K-Neighbors Classifier with other four algorithms:-

Algorithms	**Accuracy(%)**
Logistic Regression	60.25
K-Neighbours Classifier	92.01
Gaussian NB	63.39
Decision Tree Classifier	84.43
Random Forest Classifier	89.70

SIGNIFICANCE OF THE PROJECT

The significance of the project is to prove the following benefits:-

- It helps the people to easily check the chances of ADHD disease in case of any doubt if they are having solely with the help of their symptoms. They can easily check their disease by entering their symptoms.

- It saves user's time which they will spend in visiting the hospital or doctor. Because one just need to fill the given parameters given there and result will be shown within few seconds.

- The accurate analysis of medical database benefits in early disease prediction, patient care and community services. Since it is so easy and less time consuming to check is someone is suffering from some disease, one can use it frequently to analyze their health. In this process they can also come across the disease some disease at the early stage when it is easy to cure it.

- Disease prediction has the potential to benefit stakeholders such as the government and health insurance companies. It can identify patients at risk of disease or health conditions. Clinicians can then take appropriate measures to avoid or minimise the risk and in turn, improve quality of care and avoid potential hospital admissions.

SCOPE OF THE PROJECT

This project has a large scope as it has the following features which help in making it easy to use, understand and modify it:

- Saves money spent on checkups.
- No need to wait in the queue lines.
- No need to do the paper works.
- Suggest the patients to visit the doctor if needed.

This Project has the following components:

• Functional Requirements: The functional requirements specify relationship between the inputs and outputs. All the operations to be performed on the input data to obtain output are to be specified. This includes specifying the validity checks on the input and output data, parameters affected by the operations and the other operations, which must be used to transform the inputs into outputs.

• Performance Requirements: This section includes performance of the product that are set by user interaction and studying the existing system of the organization. These are stated in complete measurable terms, so that they can be verified during system evaluation phase.

Some of the performance requirements are stated below:

• User Friendly: The system produced is user friendly, understandable and easy to use so that the users of the system can easily learn to use the system. For this the system is made menu-driven with well-documented programs.

• Time Element (response and processing time): The response time of the system is very less and takes less time to execute queries and triggers.

• Maximum Throughput: The system gives maximum throughput with relevant output.

• Robustness: The system will be able to handle undesirable situations and errors encountered at various levels e.g. if the user supplies invalid input for processing, the system gracefully halts, displaying a message to the user indicating the cause of the error and prompting him it enter the correct input.

• Flexibility: The system is flexible in nature so that likely changes and alterations can easily be made.

• Information Security: Records in the system must be safe, confidential and must be prevented from unauthorised access.

CONCLUSION AND FUTURE WORK

Conclusion :

We developed a Prediction Engine which enables the user to check whether he/she has ADHD disease. The Prediction engine provides an optimal performance compared to other state of art approaches. The Prediction Engine makes use of five algorithms to predict the presence of a disease namely: Logistic Regression, K-Nearest Neighbours (KNN), GaussianNB, Decision Tree Classifier, Random Forest Classifier. Among all that five algorithm K-Nearest Neighbours had given the highest accuracy.

The reason to choose K-Nearest Neighbours algorithm among all are:

- It is effective, if the training data is large.
- Among all K-Nearest Neighbours had a highest accuracy.

Future Work:

- To enhance the accuracy deep learning can be used because deep learning gives a good accuracy on large dataset.
- To enhance the functionality of the prediction engine providing the details of 5 nearest hospitals or medical facilities to the user input location.
- Provide a user account which allows the user to keep track of their medical test data and get suggestions or support to meet the right specialists or the tests to be taken
- Provide admin controls to upload, delete the dataset which will be used to train the model.
- Automate the process of training the model and extracting pickle files of the trained models which will be consumed by the API's to predict the disease.
- Mail the detailed report of the prediction engine results along with the information of 5 nearest medical facilities details having location and contact information

REFERENCES

1. Mohammadi, M.R.; Khaleghi, A.; Nasrabadi, A.M.; Rafieivand, S.; Begol, M.; Zarafshan, H. EEG classification of ADHD and normal children using non-linear features and neural network. Biomed. Eng. Lett. 2016, 6, 66–73. [https://link.springer.com/article/10.1007/s13534-016-0218-2]

2. Tenev, A.; Markovska-Simoska, S.; Kocarev, L.; Pop-Jordanov, J.; Müller, A.; Candrian, G. Machine learning approach for classification of ADHD adults. Int. J. Psychophysiol. 2014, 93, 162–166. [https://www.sciencedirect.com/science/article/abs/pii/S0167876013000238?via=ihub]

3. Tosun, M. Effects of spectral features of EEG signals recorded with different channels and recording statuses on ADHD classification with deep learning. Phys. Eng. Sci. Med. 2021, 44, 693–702. [https://link.springer.com/article/10.1007/s13246-021-01018-x]

4. Khoshnoud, S.; Shamsi, M.; Nazari, M.A. Non-linear EEG analysis in children with attention-deficit/hyperactivity disorder during the rest condition. In Proceedings of the 2015 22nd Iranian Conference on Biomedical Engineering (ICBME), Tehran, Iran, 25–27 November 2015; pp. 87–92.

5. Chen, H.; Song, Y.; Li, X. A deep learning framework for identifying children with ADHD using an EEG-based brain network. Neurocomputing 2019, 356, 83–96. [https://www.sciencedirect.com/science/article/abs/pii/S0925231219306447?via=ihub]

6. Saini, S.; Rani, R.; Kalra, N. Prediction of Attention Deficit Hyperactivity Disorder (ADHD) using machine learning Techniques based on classification of EEG signal. In Proceedings of the 2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 25–26 March 2022; pp. 782–786.

7. Tor, H.T.; Ooi, C.P.; Lim-Ashworth, N.S.; Wei, J.K.E.; Jahmunah, V.; Oh, S.L.; Acharya, U.R.; Fung, D.S.S. Automated detection of conduct disorder and attention deficit hyperactivity disorder using decomposition and nonlinear techniques with EEG signals. Comput. Methods Programs Biomed. 2021, 200, 105941. [https://www.sciencedirect.com/science/article/abs/pii/S0169260721000158?via=ihub]

8. Dubreuil-Vall, L.; Ruffini, G.; Camprodon, J.A. Deep learning convolutional neural networks discriminate adult ADHD from healthy individuals on the basis of event-

related spectral EEG. Front. Neurosci. 2020, 14, 251. [https://www.frontiersin.org/articles/10.3389/fnins.2020.00251/full]

9. R.Y. Karimu, S. Azadi, Diagnosing the ADHD using a mixture of expert fuzzy models, Int. J. Fuzzy Syst. 20 (4) (2018) 1282–1296.

10. B. TaghiBeyglou, N. Hasanzadeh, F. Bagheri, M. Jahed, ADHD diagnosis in children using common spatial pattern and nonlinear analysis of filter banked EEG, in: 2020 28th Iranian Conference on Electrical Engineering, ICEE, IEEE, 2020, pp. 1–5.

11. A. Ekhlasi, A.M. Nasrabadi, M.R. Mohammadi, Analysis of effective connectivity strength in children with attention deficit hyperactivity disorder using phase transfer entropy, Iran. J. Psychiatry 16 (4) (2021) 374.

12. R. Catherine Joy, S. Thomas George, A. Albert Rajan, M. Subathra, Detection of adhd from eeg signals using different entropy measures and ann, Clin. EEG Neurosci. 53 (1) (2022) 12–23.

13. H. Chen, Y. Song, X. Li, A deep learning framework for identifying children with ADHD using an EEG-based brain network, Neurocomputing 356 (2019) 83–96.

14. M. Bakhtyari, S. Mirzaei, ADHD detection using dynamic connectivity patterns of EEG data and convlstm with attention framework, Biomed. Signal Process.Control 7(2022)103708.

15. H. Chen, Y. Song, X. Li, Use of deep learning to detect personalized spatial-frequency abnormalities in EEGs of children with ADHD, J. Neural Eng. 16 (6) (2019) 066046.

16. https://www.cdc.gov/datastatistics/index.html

17. https://ieee-dataport.org/open-access/eeg-data-adhd-control-children

18. ChatGPT