CHAPTER - 1

INTRODUCTION

Over the most recent couple of decades, brain-computer interface (BCI) turned into one of the most favorite fields of research due to its unlimited possible applications such as brain fingerprinting, detection and prevention of neurological diseases, adaptive e-learning, fatigue, stress, and depression monitoring. BCI establishes an effective communication link between a brain and a device by capturing the most relevant feature required for the establishment.

Among the applications of BCI given above, detection of neurological diseases has turned into an acute research field due to its growing importance which need not be mentioned. Due to the complex structure of the brain that varies with age and pathological history, it has always been very hard to detect neuro-degenerative diseases.

It is very much important to diagnose these diseases in early stages. Computer-aided mechanisms play a better role than conventional manual practices in detection of different brain diseases. However, the main focus of this study is to provide a brief review on recent ML and DL approaches to detect four different most common types of brain diseases such as **Alzheimer's, brain tumor, epilepsy, and Parkinson's.**

Brain is the controlling center of our body. With the advent of time, newer and newer brain diseases are being discovered. Thus, because of the variability of brain diseases, existing diagnosis or detection systems are becoming challenging and are still an open problem for research. Detection of brain diseases at an early stage can make a huge difference in attempting to cure them. In recent years, the use of artificial intelligence (AI) is surging through all spheres of science, and no doubt, it is revolutionizing the field of neurology. Application of AI in medical science has made brain disease prediction and detection more accurate and precise. In this study,

we present a review on recent machine learning and deep learning approaches in detecting four brain diseases such as Alzheimer's disease (AD), brain tumor, epilepsy, and Parkinson's disease. 147 recent articles on four brain diseases are reviewed considering diverse machine learning and deep learning approaches, modalities, datasets etc. Twenty-two datasets are discussed which are used most frequently in the reviewed articles as a primary source of brain disease data. Moreover, a brief

overview of different feature extraction techniques that are used in diagnosing brain diseases is provided.

ML is a process of training a computer to apply its past experience to solve a problem given to it. The concept of application of ML in different fields to solve problems faster than human has gained significant interest due to the current availability of cheaper computing power and inexpensive memory. This makes it possible to process and analyze a very large amount of data to discover insights and correlations amongst the data which are not so obvious to human eye. Its intelligent behavior is based on different algorithms which enables the machine to make abstractions based on experience, in order to produce salient judgments.

On the other hand, DL is a sub-field of ML, however, a more advanced approach which enables computers to automatically extract, analyze and understand the useful information from the raw data by imitating how humans think and learn. Precisely, deep learning is a group of techniques that is neural data driven and based on automatic feature engineering processes. The automatic learning of fe`atures from inputs is what makes it so accurate and of excellent performance. A quick overview of the difference between artificial intelligence (AI), ML, and DL is provided. Success in making the right decision in ML and DL relies on the classification algorithm. There are different classification algorithms available in ML which are specially designed for classification purposes and the performance is quite decent.

Even though performance of ML is quite up to rank, it is currently being replaced by DL in most classification applications. The Principle difference between ML and DL is in the technique of extracting the features on which the classifier works on. Extracted features of DL from several non-linear hidden layers makes its classification performance far better than ML’s classification which relies on handcrafted feature. In order to understand the difference between ML and DL.

CHAPTER - 2

LITERATURE SURVEY

2.1 TITLE: Machine Learning and Deep Learning Approaches for Brain Disease Diagnosis

AUTHOR: PROTIMA KHAN1, MD. FAZLUL KADER

Brain is the controlling center of our body. With the advent of time, newer and newer brain diseases are being discovered. Thus, because of the variability of brain diseases, existing diagnosis or detection systems are becoming challenging and are still an open problem for research. Detection of brain diseases at an early stage can make a huge difference in attempting to cure them. In recent years, the use of artificial intelligence (AI) is surging through all spheres of science, and no doubt, it is revolutionizing the field of neurology.

Application of AI in medical science has made brain disease prediction and detection more accurate and precise. In this study, we present a review on recent machine learning and deep learning approaches in detecting four brain diseases such as Alzheimer's disease (AD), brain tumor, epilepsy, and Parkinson's disease. 147 recent articles on four brain diseases are reviewed considering diverse machine learning and deep learning approaches, modalities, datasets etc. Twenty-two datasets are discussed which are used most frequently in the reviewed articles as a primary source of brain disease data. Moreover, a brief overview of different feature extraction techniques that are used in diagnosing brain diseases is provided.

Finally, key findings from the reviewed articles are summarized and a number of major issues related to machine learning/deep learning-based brain disease diagnostic approaches are discussed. Through this study, we aim at finding the most accurate technique for detecting different brain diseases which can be employed for future betterment.

Over the most recent couple of decades, brain-computer interface (BCI) turned into one of the most favorite fields of research due to its unlimited possible applications such as the associate editor coordinating the review of this manuscript and approving it for publication was K. C. Santosh. brain fingerprinting, detection and prevention of neurological diseases, adaptive e-learning, fatigue, stress, and depression monitoring and so on. BCI establishes an effective communication link between a brain and a device by capturing the most relevant feature required for the establishment.

2.2 TITLE: A Deep Learning Model Based on Concatenation Approach for the Diagnosis of Brain Tumor

AUHTOR: NEELUM NOREEN, SELLAPPAN PALANIAPPAN, ABDUL QAYYUM, IFTIKHAR AHMAD, MUHAMMAD IMRAN, AND MUHAMMAD SHOAIB

Brain tumor is a deadly disease and its classification is a challenging task for radiologists because of the heterogeneous nature of the tumor cells. Recently, computer-aided diagnosis-based systems have promised, as an assistive technology, to diagnose the brain tumor, through magnetic resonance imaging (MRI). In recent applications of pre-trained models, normally features are extracted from bottom layers which are different from natural images to medical images.

To overcome this problem, this study proposes a method of multi-level features extraction and concatenation for early diagnosis of brain tumor. Two pretrained deep learning models i.e. Inception-v3 and DensNet201 make this model valid. With the help of these two models, two different scenarios of brain tumor detection and its classification were evaluated. First, the features from different Inception modules were extracted from pre-trained Inception-v3 model and concatenated these features for brain tumor classification. Then, these features were passed to softmax classifier to classify the brain tumor. Second, pre-trained DensNet201 was used to extract features from various DenseNet blocks. Then, these features were concatenated and passed to softmax classifier to classify the brain tumor. Both scenarios were evaluated with the help of three-class brain tumor dataset that is available publicly.

The proposed method produced 99.34 %, and 99.51% testing accuracies respectively with Inception-v3 and DensNet201 on testing samples and achieved highest performance in the detection of brain tumor. As results indicated, the proposed method based on features concatenation using pre-trained models outperformed as compared to existing state-of-the-art deep learning and machine learning based methods for brain tumor classification.

This paper discussed the application of deep learning models for the identification of brain tumor. In this paper, two different scenarios were assessed. Firstly, pre-trained DensNet201 deep learning model was used, and the features were extracted from various DensNet blocks. Then, these features were concatenated and passed to softmax classifier to classify the brain tumor.

2.3 TITLE: Analysis of Brain Functional Network Based on EEG Signals for Early-Stage Parkinson's Disease Detection

AUTHOR: WEI ZHANG, XIAOXUAN HAN, SHUJUAN QIU, TENG LI

The early diagnosis of Parkinson's disease (PD) has always been a difficult problem to be solved clinically. At present, there is no clinical auxiliary diagnostic index for reference. We attempted to extract potential biomarkers for early PD from the currently used scalp EEG detection methods in clinical practice. We calculated the phase synchronization index to quantify the synchrony of EEG channels in various frequency bands (delta, theta, alpha and beta bands) of early PD.

The results showed that the synchronization of early PD in the delta band was significantly lower than the healthy level, and the brain region reflecting the lower synchronization was located in the temporal lobe, the posterior temporal lobe, the parietal lobe (the posterior center) and the occipital lobe. Moreover, this lower synchronicity is consistent with weaker brain functional connections. Besides, by constructing functional brain network, the graph theoretic topological features of each frequency band of early PD are presented.

We have found that early PD has characteristics of small world network in the delta and beta bands, and functional integration and separation characteristics of brain network in early PD are significantly abnormal in the delta, theta, alpha and beta bands. These results indicate that early PD has significant pathological changes from the perspective of brain function network analysis, and its characteristics can be described by multiple features, which may provide auxiliary guidance for the clinical diagnosis of early PD, and also provide theoretical support for the brain function changes of early PD.

In early PD, there was a significant decrease in brain synchronization and functional connectivity in the delta band, and the most significant brain regions were located in bilateral posterior temporal lobe, parietal lobe and occipital lobe. Interestingly, this lower synchronicity is consistent with weaker brain functional connections. Such early PD abnormalities may serve as an early warning of cognitive decline in PD patients. What's more, there are significance differences among specific frequency bands in the brain network structure between early PD patients and healthy subjects. Such specific brain network characteristics may reveal the specific brain activity of early PD. The graph theory derived features set based on the brain functional network could be used as an appropriate marker to assist in the diagnosis of early PD.

2.4 TITLE: Epileptic Seizures Prediction Using Deep Learning Techniques

AUTHOR: SYED MUHAMMAD USMAN, SHEHZAD KHALID, AND MUHAMMAD HASEEB ASLAM

Epilepsy is a very common neurological disease that has affected more than 65 million people worldwide. In more than 30 % of the cases, people affected by this disease cannot be cured with medicines or surgery. However, predicting a seizure before it actually occurs can help in its prevention; through therapeutic intervention. Studies have observed that abnormal activity inside the brain begins a few minutes before the start of a seizure, which is known as preictal state. Many researchers have tried to find a way for predicting this preictal state of a seizure but an effective prediction in terms of high sensitivity and specificity still remains a challenge.

The current study, proposes a seizure prediction system that employs deep learning methods. This method includes preprocessing of scalp EEG signals, automated features extraction; using convolution neural network and classification with the support of vector machines. The proposed method has been applied on 24 subjects of scalp EEG dataset of CHBMIT resulting in successfully achieving an average sensitivity and specificity of 92.7% and 90.8% respectively.

Epilepsy is a neurological disorder in which a patient undergoes frequent seizures. More than 1% of the world population is affected by this disease. Patients affected by this disease can be treated with medicines or by surgical treatments. However, it has been observed that if seizures have occurred then in more than 30% of the cases patient’s subsequent seizures cannot be controlled with current methods of treatments that include medicine or surgical procedures. Therefore, it is extremely important to predict the subsequent seizures before they occur so that seizure can be prevented with the help of medication.

Patients affected from epilepsy can live a healthy and risk-free life if effective prediction of seizures is ensured. Our proposed method combines feature extraction using CNN and classification with the help of machine learning classifier to achieve increased sensitivity and specificity compared with other methods. However, there is still room for improvement in many aspects. In future, if preprocessing is further enhanced for increasing signal to noise ratio. Our proposed method like other state of the art methods provides patient specific seizures’ prediction. In future, more research is required for non-patient specific epileptic seizures prediction methods.

2.5 TITLE: Studying the Manifold Structure of Alzheimer's Disease: A Deep Learning Approach Using Convolutional Autoencoders

AUTHOR: FRANCISCO J. MARTINEZ- MURCIA, ANDRES ORTIZ, JUAN-MANUEL GORRIZ, JAVIER RAMIREZ, AND DIEGO

Many classical machine learning techniques have been used to explore Alzheimer's disease (AD), evolving from image decomposition techniques such as principal component analysis toward higher complexity, non-linear decomposition algorithms. With the arrival of the deep learning paradigm, it has become possible to extract high-level abstract features directly from MRI images that internally describe the distribution of data in low-dimensional manifolds. In this work, we try a new exploratory data analysis of AD based on deep convolutional autoencoders.

We aim at finding links between cognitive symptoms and the underlying neurodegeneration process by fusing the information of neuropsychological test outcomes, diagnoses, and other clinical data with the imaging features extracted solely via a data-driven decomposition of MRI. The distribution of the extracted features in different combinations is then analyzed and visualized using regression and classification analysis, and the influence of each coordinate of the autoencoder manifold over the brain is estimated.

The imaging-derived markers could then predict clinical variables with correlations above 0.6 in the case of neuropsychological evaluation variables such as the MMSE or the ADAS11 scores, achieving a classification accuracy over 80% for the diagnosis of AD. ALZHEIMER'S Disease (AD) is the most common type of neurodegenerative disease in the world, affecting more than 5% of the population in Europe and with an incidence of 11.08 per 1000 person-years.

With a yet unknown aetiology, current diagnosis of AD often depend on clinical history and the outcomes of widely extended neuropsychological tests such as the Mini-Mental State Exam (MMSE) that, according to recent studies may add confounding information to the procedure of diagnosis. Therefore, understanding the disease progression as well as studying and standardizing new disease markers is paramount. Substantial advances in the technology

CHAPTER - 3

EXISITING SYSTEM

The existing system utilizes a variety of machine learning (ML) and deep learning (DL) algorithms to train for different diseases such as brain tumors, Alzheimer's, Parkinson's, and epilepsy. This approach enhances diagnostic accuracy by leveraging the strengths of these algorithms to identify patterns and characteristics specific to each condition.

By employing separate datasets for training each disease, the system can effectively capture the unique features and nuances associated with brain tumors, Alzheimer's, Parkinson's, and epilepsy. This tailored approach ensures that the models are finely tuned to recognize the specific markers of each disease, thereby improving the precision of diagnosis and treatment recommendations.

Advantages of the Existing system:

Early Detection: ML and DL models can detect subtle changes or abnormalities in medical imaging or patient data that may indicate the presence of a disease. Early detection is crucial for initiating timely interventions and improving treatment effectiveness.

Scalability: The existing system can scale to handle large volumes of data and accommodate new sources of information as they become available. This scalability ensures that the system remains effective and relevant in diverse healthcare settings and patient populations.

Drawbacks of the Existing system:

Application-specific Models: ML and DL techniques used in the existing system are typically tailored for specific diseases. For example, a model trained to detect brain tumors may not perform well when applied to diagnose Alzheimer's. This limitation arises because each model is designed to capture the intricacies of a particular disease, making it challenging to predict other conditions accurately.

Time-consuming Process: Training ML and DL algorithms for each disease can be a time-consuming process. It involves collecting and preprocessing data, selecting appropriate features, training the models, and fine-tuning parameters. This extensive process may delay the implementation of the system or updates to accommodate new data or advancements in the field.

CHAPTER - 4

PROPOSED SYSTEM

The proposed solution involves consolidating images of brain tumors, Alzheimer's disease, Parkinson's disease, and epilepsy into a single dataset. This consolidated dataset would then be used to train a unified machine learning or deep learning model capable of distinguishing between the different diseases and making predictions accordingly.

By training the model on a combined dataset, it is expected to learn the common features and unique characteristics of each disease. This approach can potentially improve the model's ability to generalize and make accurate predictions across multiple conditions. Instead of having separate models for each disease, which may not generalize well to other conditions, a unified model can leverage shared patterns and features among the diseases, leading to more robust and accurate predictions.

Overall, the proposed system aims to optimize the diagnostic process by leveraging a unified approach to training machine learning and deep learning models. By consolidating datasets and developing a unified model, the system seeks to improve diagnostic accuracy while addressing challenges associated with using separate datasets for each disease.

Problem Statement

- The problem statement revolves around optimizing the diagnostic process by consolidating the datasets and developing a unified model capable of accurately predicting the presence of each of the four diseases while mitigating the drawbacks associated with using separate datasets.
- The system aims to address the challenge of accurately diagnosing brain tumor, Alzheimer's disease, Parkinson's disease, and epilepsy using machine learning and deep learning algorithms.

CHAPTER - 5

REQUIREMENTS ANALYSIS

In the autonomous vehicle sector, computer-assisted learning is a rapidly increasing and dynamic area of research. The recent researchers in machine learning and artificial intelligence promise the improved accuracy of perception of Brian Diseases detection. Here, computers are enabled to think by developing intelligence by learning. There are many types of Machine Learning Techniques that are used to classify data sets.

- Functional Requirements.
- Non-Functional Requirements.

5.1 FUNCTIONAL REQUIREMENTS

Functional requirements are a crucial aspect of software development projects, as they outline the specific features and behaviors that a system must exhibit in order to satisfy its users' needs and requirements. These requirements serve as a foundation for the design, development, and testing phases of a project, providing a clear roadmap for building the desired system.

Features and Capabilities:

Functional requirements describe the core features and capabilities that the system must possess. These may include functionalities such as user authentication, data entry forms, search functionality, reporting tools, and so on.

User Interactions:

Functional requirements specify how users will interact with the system. This includes details about user interfaces, navigation flows, input methods (e.g., mouse, keyboard, touch), and feedback mechanisms (e.g., error messages, confirmation dialogs).

System Inputs and Outputs:

Functional requirements define the types of inputs that the system will accept and the corresponding outputs it will produce. This encompasses data formats, validation rules, processing logic, and output formats. For example, a system may require users to input data into specific fields on a form and then generate reports based on the entered data.

Business Rules and Logic:

Functional requirements outline the business rules and logic that govern the behavior of the system. These rules dictate how the system should process data, make decisions, perform calculations, enforce constraints, and ensure compliance with business policies and regulations.

Error Handling and Recovery:

Functional requirements specify how the system should handle errors, exceptions, and unexpected scenarios. This includes defining error messages, error codes, error logging mechanisms, and strategies for recovering from errors and restoring system functionality.

Performance Criteria:

Functional requirements may include performance criteria that the system must meet in terms of response times, throughput, scalability, and resource utilization. These criteria ensure that the system performs efficiently and effectively under various workload conditions.

Integration with External Systems:

Functional requirements describe how the system will integrate with external systems, services, or components. This includes defining data exchange formats, communication protocols, APIs, and mechanisms for synchronizing data between different systems.

Security Measures:

Functional requirements outline the security features and measures that the system must implement to protect data, prevent unauthorized access, and ensure compliance with security standards and regulations. This includes authentication mechanisms, access control policies, encryption methods, and audit trails.

5.2NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements, also known as quality attributes or constraints, define the criteria that specify how a system should behave rather than what it should do. These requirements focus on aspects such as performance, usability, reliability, security, and scalability. Unlike functional requirements, which describe specific features and functionalities, non-functional requirements describe the overall characteristics and qualities of the system.

Here are some key aspects of non-functional requirements:

Performance:

Non-functional requirements related to performance specify how the system should perform in terms of response times, throughput, and resource utilization. This includes requirements for maximum response time, minimum throughput, and acceptable levels of latency under various load conditions.

Usability:

Usability requirements define the ease of use and user-friendliness of the system. This includes requirements for intuitive user interfaces, clear navigation flows, accessibility features, and support for different user preferences and languages.

Reliability:

Reliability requirements specify the system's ability to perform consistently and predictably over time. This includes requirements for uptime, availability, fault tolerance, and mean time between failures (MTBF). The system should be able to recover gracefully from failures and maintain data integrity.

Security:

Security requirements outline the measures that the system must implement to protect against unauthorized access, data breaches, and other security threats. This includes requirements for authentication, authorization, encryption, data privacy, and compliance with security standards and regulations.

Scalability:

Scalability requirements define the system's ability to handle increasing workloads and growing user populations. This includes requirements for horizontal scalability (adding more nodes) and vertical

scalability (increasing resources on existing nodes) to accommodate future growth without compromising performance.

Maintainability:

Maintainability requirements specify how easy it is to maintain, update, and modify the system over time. This includes requirements for well-documented code, modular architecture, version control, and automated testing and deployment processes.

Compatibility:

Compatibility requirements define the system's ability to interoperate with other systems, platforms, and devices. This includes requirements for compatibility with different operating systems, web browsers, databases, and third-party software components.

Regulatory Compliance:

Regulatory compliance requirements specify the legal and regulatory standards that the system must adhere to. This includes requirements for data protection, privacy laws, industry regulations, and certification requirements.

Non-functional requirements complement functional requirements by ensuring that the system meets the desired level of performance, usability, reliability, security, scalability, maintainability, compatibility, and regulatory compliance. They provide a holistic view of the system's qualities and characteristics, guiding the design, development, and testing processes to ensure that the final product meets users' expectations and business needs.

CHAPTER - 6

DESIGN AND METHODOLOGY

6.1 TECHNOLOGIES

Deep Learning:

Deep learning is a subset of machine learning, which is a branch of artificial intelligence (AI) focused on training algorithms to learn from data and make predictions or decisions**.**

What sets deep learning apart is its use of artificial neural networks with many layers (hence the term "deep"). These neural networks are inspired by the structure and function of the human brain, where each layer of neurons processes different aspects of the input data, gradually extracting higher-level features.

Deep learning algorithms learn to perform tasks directly from data, without relying on explicit programming instructions. They are particularly well-suited for tasks such as image recognition, natural language processing, speech recognition, and many others.

In the context of deep learning and machine learning, a model refers to a mathematical representation of a real-world process or system. It's essentially an algorithm that has been trained on data to learn patterns, relationships, or features, which it can then use to make predictions or decisions about new data.

Models are created and trained using various algorithms and techniques, depending on the specific task and the nature of the data. These models can range from simple linear regression models to complex deep neural networks with multiple layers**.**

Deep Learning Algorithms:

Convolutional neural network:

A Convolutional Neural Network (CNN) is a type of deep neural network that is primarily used for analysing visual data. It's particularly effective for tasks such as image recognition, object detection, and image classification.

CNN have been widely used in various medical imaging tasks, including the detection and diagnosis of brain diseases from medical images such as MRI (Magnetic Resonance Imaging) scans. Here's a description of how CNNs can be applied in a brain disease detection project:

Data Collection and Preprocessing: The first step involves collecting a dataset of brain MRI scans. These scans may include images of healthy brains as well as brains affected by different diseases such as Alzheimer's disease, brain tumors, or multiple sclerosis. The images are typically pre-processed to standardize their resolution, orientation, and contrast, ensuring consistency across the dataset.

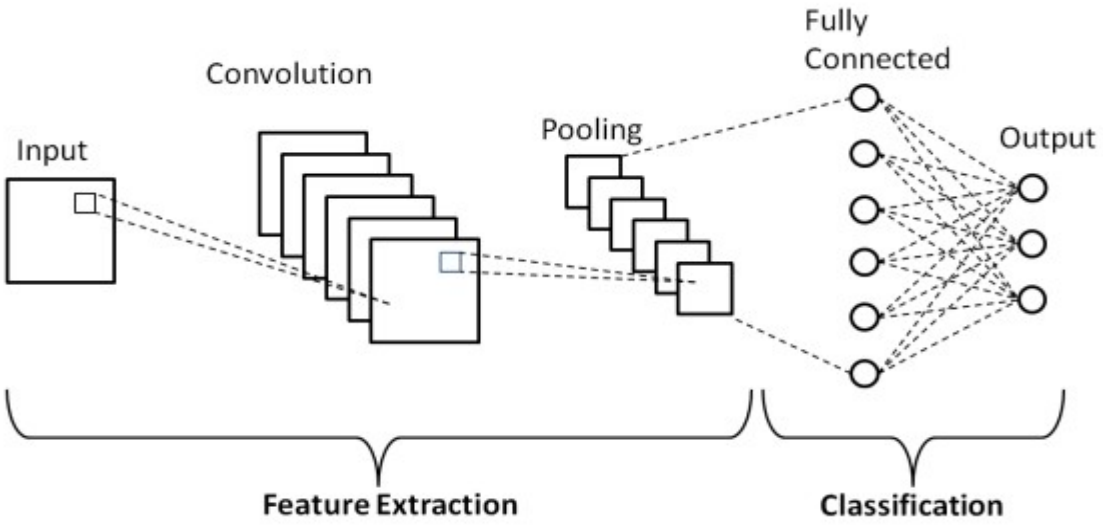
Model Architecture: The CNN architecture used in the project typically consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers are responsible for extracting features from the input images. These layers apply a series of filters or kernels to the input image, detecting patterns such as edges, textures, or shapes. Pooling layers then down sample the feature maps produced by the convolutional layers, reducing their spatial dimensions while retaining important information. Finally, fully connected layers integrate the extracted features and perform classification based on learned representations.

Training: The CNN is trained using a supervised learning approach, where it learns to differentiate between different brain diseases based on labeled MRI scans. During training, the model adjusts its parameters (weights and biases) using an optimization algorithm such as stochastic gradient descent, to minimize the difference between its predictions and the ground truth labels. The training process involves feeding batches of MRI images into the network, computing the prediction errors, and updating the model parameters iteratively.

Validation and Testing: After training, the performance of the CNN is evaluated using a separate validation dataset. This dataset contains MRI scans that were not seen by the model during training. The model's predictions are compared against the ground truth labels to assess its accuracy, sensitivity, specificity, and other performance metrics. Additionally, the model may be tested on an independent test set to further evaluate its generalization capability.

Fine-tuning and Optimization: The CNN architecture and hyperparameters may be fine-tuned and optimized to improve performance. Techniques such as transfer learning, data augmentation, dropout regularization, and hyperparameter tuning may be employed to enhance the model's robustness and generalization ability.

Deployment: Once trained and validated, the CNN can be deployed in clinical settings for automated brain disease detection. It can analyse new MRI scans and provide clinicians with diagnostic insights, helping to assist in early detection, treatment planning, and monitoring of brain diseases.



EfficientNetB0:

EfficientNetB0 is a convolutional neural network architecture known for its impressive performance and efficiency in image classification tasks. Introduced by researchers at Google AI in 2019, EfficientNetB0 represents the baseline model in the EfficientNet series.

EfficientNetB0 serves as the foundation upon which larger variants, such as EfficientNetB1, B2, B3, and so on, are built. Each variant increases in depth, width, and resolution, offering progressively higher performance levels.

EfficientNetB0 has found widespread use in various computer vision applications, including image classification, object detection, and semantic segmentation. Its balance between accuracy and efficiency makes it a popular choice for real-world deployments where computational resources may be limited.

EfficientNetB0, like other convolutional neural network architectures, excels in feature extraction, which is a crucial step in many computer vision tasks. Here's how EfficientNetB0 performs in terms of feature extraction:

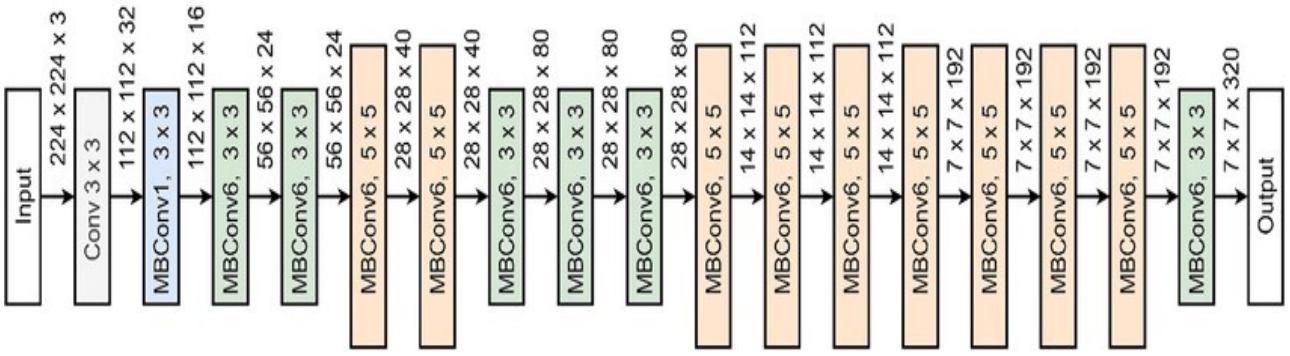
Hierarchical feature extraction: EfficientNetB0 consists of multiple layers, each designed to extract hierarchical features from input images. These layers typically start with low-level features like edges and gradients and gradually progress to higher-level features like textures, shapes, and object parts.

Multi-scale feature representation: Thanks to its compound scaling method, EfficientNetB0 can capture features at multiple scales. This is essential for handling objects of various sizes and complexities within images. The network can extract both local and global features effectively, allowing it to understand context and semantics.

Efficient feature representation: Despite its compact size, EfficientNetB0 can represent features efficiently. The network achieves this by leveraging depth-wise separable convolutions, which reduce the computational cost while maintaining expressive power. This efficiency makes EfficientNetB0 suitable for resource-constrained environments without sacrificing performance.

Transfer learning: Pre-trained EfficientNetB0 models are often used as feature extractors in transfer learning scenarios. By leveraging knowledge learned from large-scale datasets during pre-training, these models can effectively extract relevant features from new datasets with minimal fine-tuning. This approach saves computational resources and training time while still achieving competitive performance.

Overall, EfficientNetB0 offers an effective and efficient solution for feature extraction in various computer vision tasks, making it a popular choice among researchers and practitioners.

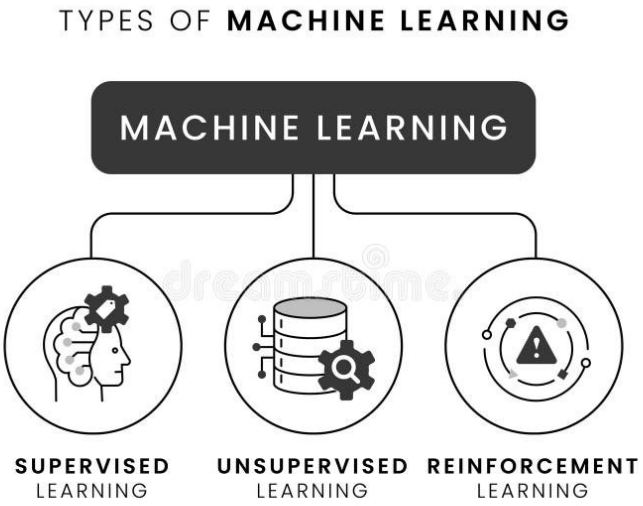


Machine Learning:

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of Machine Learning.

Machine learning is a subfield of artificial intelligence (AI) that focuses on developing algorithms and models that enable computers to learn from data and make predictions or decisions without being explicitly programmed to do so. It encompasses a variety of techniques and methodologies aimed at teaching computers to recognize patterns and extract insights from data, with the ultimate goal of improving their performance over time.

Machine learning has numerous applications across various domains, including but not limited to healthcare, finance, marketing, computer vision, natural language processing, and robotics. As the availability of data continues to grow and computational resources become more powerful, machine learning is expected to play an increasingly important role in driving innovation and solving complex problems.



Classification of Machine Learning

At a broad level, machine learning can be classified into three types:

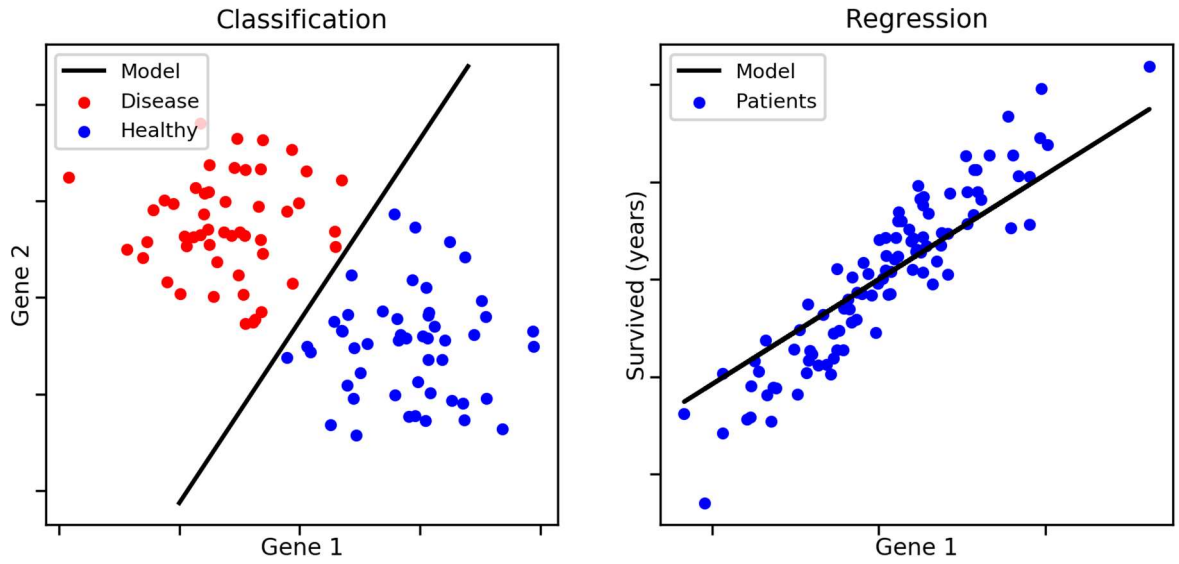
1. Supervised learning
2. Unsupervised learning
3. Reinforcement learning

1) Supervised Learning:

Supervised learning is a type of machine learning method in which we provide sample labeled data to the machine learning system in order to train it, and on that basis, it predicts the output. The system creates a model using labeled data to understand the datasets and learn about each data. Once the training and processing are done then we test the model by providing sample data to check whether it is predicting the exact output or not.

Supervised learning can be grouped further in two categories of algorithms:

- **Classification:** In classification tasks, the output variable (label) is categorical. The algorithm learns to classify input data into predefined categories or classes. Examples of classification tasks include spam email detection (classifying emails as spam or not spam), image recognition (classifying images into different categories), and sentiment analysis (classifying text as positive, negative, or neutral).
- **Regression:** In regression tasks, the output variable is continuous. The algorithm learns to predict a numeric value based on input features. Examples of regression tasks include predicting house prices based on features such as size, location, and number of bedrooms, forecasting stock prices based on historical data, and estimating the age of a person based on demographic information.



Supervised learning is widely used in various real-world applications, ranging from image and speech recognition to medical diagnosis, autonomous driving, recommendation systems, and more.

2) Unsupervised Learning:

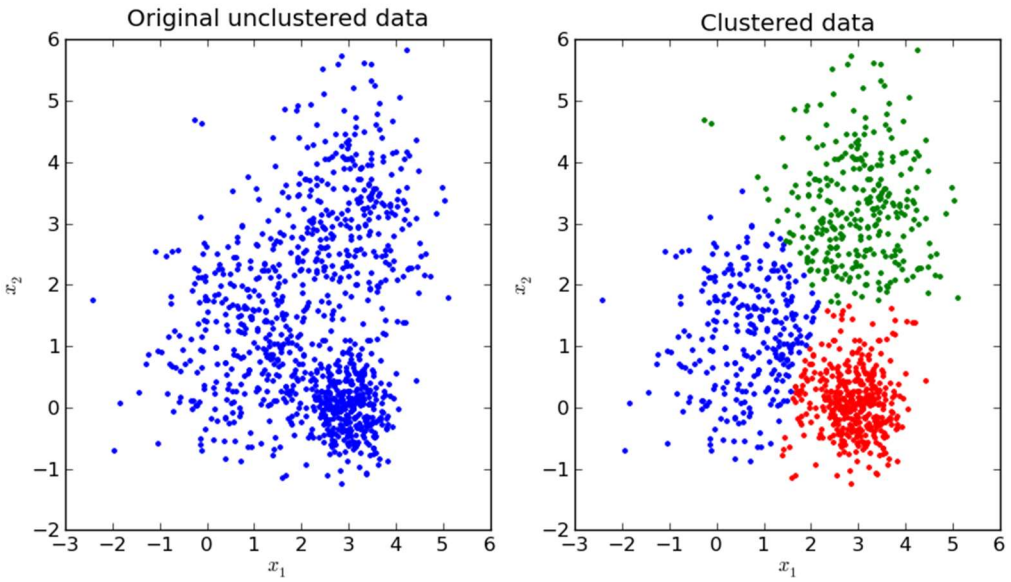
Unsupervised learning is a learning method in which a machine learns without any supervision. The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

It can be further classified into two categories of algorithms:

- **Clustering:** Clustering involves grouping similar data points together into clusters based on their features or characteristics. The algorithm aims to maximize the similarity within clusters

while maximizing the dissimilarity between clusters. Common clustering algorithms include K-means clustering, hierarchical clustering, and Gaussian mixture models.

- **Association:** Association involves discovering interesting relationships or associations between variables in large datasets. This is commonly used in market basket analysis to identify frequent item sets and discover patterns in consumer behavior. The Apriori algorithm is a well-known technique for association rule learning.



Unsupervised learning has numerous applications in various fields, including anomaly detection, customer segmentation, recommendation systems, image and text clustering, and more. It can be particularly useful in scenarios where labeled data is scarce or expensive to obtain, or when exploring and understanding the underlying structure of the data is of interest.

3) Reinforcement Learning:

Reinforcement learning (RL) is a type of machine learning paradigm where an agent learns to interact with an environment in order to maximize cumulative rewards. Unlike supervised learning, where the algorithm learns from labeled examples, and unsupervised learning, where the algorithm learns patterns from unlabeled data, reinforcement learning learns through trial and error by receiving feedback in the form of rewards or penalties based on its actions.

The fundamental components of reinforcement learning are:

Agent: The entity or system that interacts with the environment. The agent makes decisions based on the information it receives from the environment and its current state.

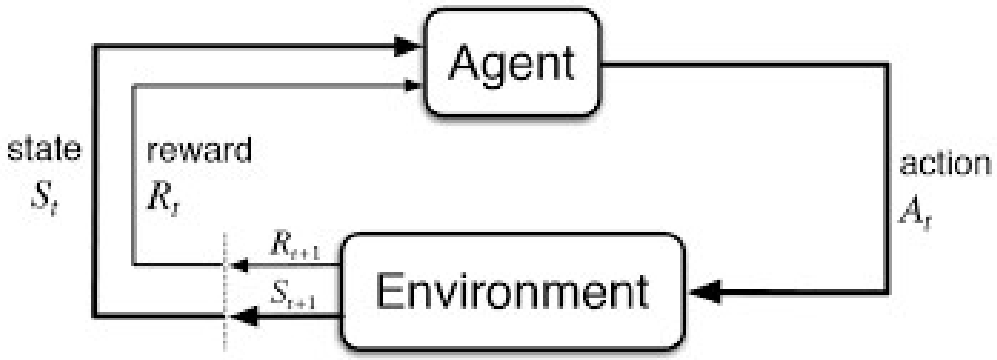
Environment: The external system or domain with which the agent interacts. The environment provides feedback to the agent based on its actions and may change in response to those actions.

State: The current situation or configuration of the environment. The state provides context for the agent's decision-making process.

Action: The choices available to the agent at each state. The agent selects actions based on its current state and the policy it follows.

Reward: The feedback signal provided by the environment to the agent after each action. Rewards indicate the desirability or utility of the agent's actions and are used to guide the learning process.

The goal of reinforcement learning is to learn a policy, which is a mapping from states to actions, that maximizes the cumulative reward over time. This is typically achieved through iterative trial and error, where the agent explores different actions in different states and learns from the resulting rewards.



Machine Learning algorithms:

1. Random forest:

Random Forest is a popular ensemble learning technique used in machine learning for both classification and regression tasks. It belongs to the family of decision tree-based algorithms and is known for its robustness and versatility.

Here's how Random Forest works:

Decision Trees: Random Forest is built upon the concept of decision trees. Decision trees are simple, tree-like structures where each internal node represents a decision based on a feature, each branch

represents an outcome of that decision, and each leaf node represents a class label (in classification) or a numeric value (in regression).

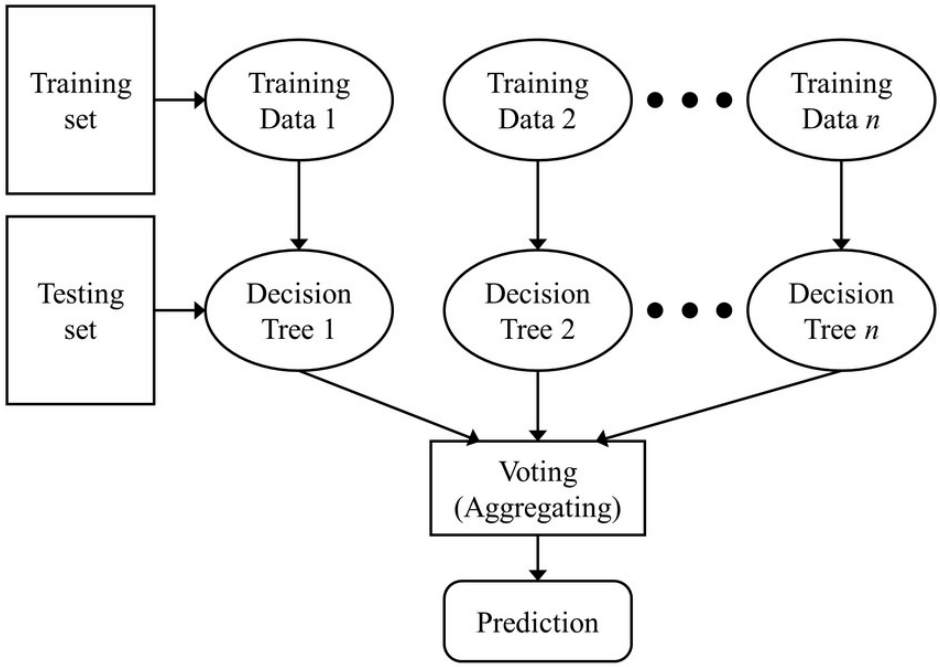
Ensemble Learning: Random Forest employs an ensemble learning approach, where multiple decision trees are trained independently and their predictions are combined to make the final prediction. This ensemble of decision trees helps to improve the overall performance and robustness of the model.

Random Subspace Sampling: The "random" in Random Forest refers to two main sources of randomness used during training:

Random sampling of the training data: Each decision tree in the Random Forest is trained on a bootstrap sample of the original training data, where samples are drawn with replacement. This means that each tree sees a slightly different subset of the training data.

Random selection of features: At each node of each decision tree, only a random subset of features (rather than all features) is considered for splitting. This helps to introduce diversity among the trees and reduces the risk of overfitting.

Voting or Averaging: For classification tasks, the final prediction of the Random Forest is typically determined by majority voting among the individual trees. For regression tasks, the final prediction is often computed as the average (or another aggregation function) of the predictions of all the individual trees.

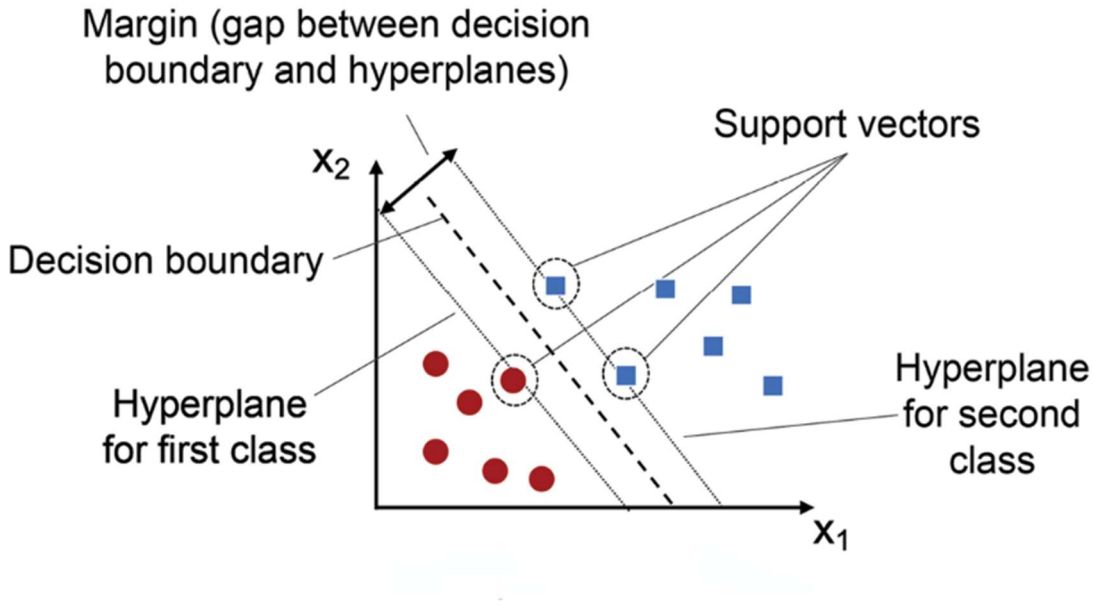


2. Support Vector Machine:

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It aims to find the hyperplane that best separates the data points of different classes while maximizing the margin, which is the distance between the hyperplane and the closest data points (support vectors).

SVM can handle both linear and nonlinear decision boundaries through the use of appropriate kernel functions. It is effective in high-dimensional spaces and is robust to overfitting. However, SVM also has limitations, such as the need for proper parameter tuning and the computational complexity of training large datasets. Overall, SVM is a powerful and versatile algorithm for classification tasks.

SVM is widely used in various applications, including text classification, image classification, bioinformatics, and finance. Its ability to handle high-dimensional data, robustness to overfitting, and effectiveness in dealing with both linearly and non-linearly separable data make it a popular choice in machine learning and pattern recognition tasks.

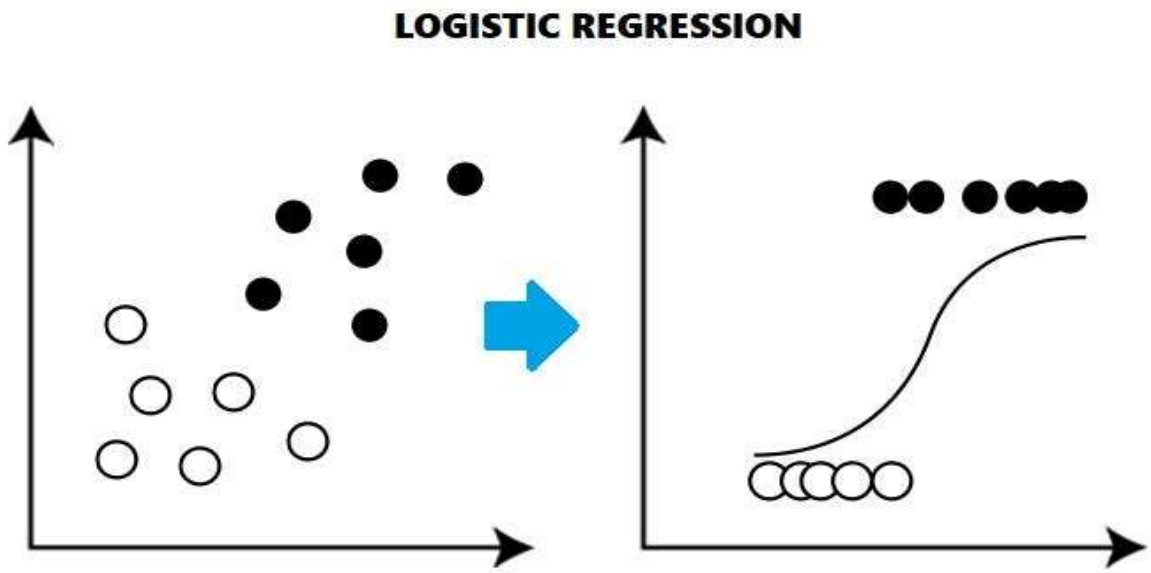


4. Logistic Regression:

Logistic Regression is a statistical model used for binary classification tasks. Unlike linear regression, which predicts continuous values, logistic regression predicts the probability that a given input belongs to a particular class. It models the relationship between the independent variables (features)

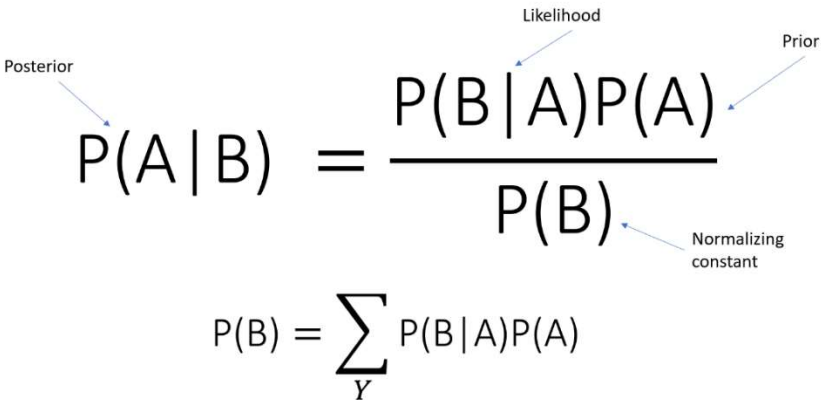
and the binary dependent variable (class label) using the logistic function, also known as the sigmoid function. The logistic function maps the output of a linear combination of the input features to a value between 0 and 1, representing the probability of belonging to the positive class. Logistic Regression is a widely used algorithm due to its simplicity, interpretability, and efficiency.

Logistic regression is widely used in various fields, including healthcare (e.g., disease diagnosis), finance (e.g., credit risk analysis), marketing (e.g., customer churn prediction), and more. Despite its simplicity, logistic regression is often favoured due to its interpretability, efficiency, and effectiveness in many real-world classification tasks.



5. Naive Bayes:

Naive Bayes is a simple yet powerful probabilistic classifier based on Bayes' theorem with the "naive" assumption of feature independence. It's commonly used for classification tasks, particularly in text categorization, spam filtering, and sentiment analysis. Naive Bayes is a simple yet effective probabilistic classifier based on Bayes' theorem with strong independence assumptions between features. It's commonly used for classification tasks, particularly in text classification.



6.2 MODULES

- TENSORFLOW
- NUMPY
- STREAMLIT
- MATPLOTLIB
- SCIKIT-LEARN

TENSORFLOW

TensorFlow is an open-source machine learning framework developed by Google Brain for building and training various types of machine learning models, particularly deep learning models. It provides a comprehensive ecosystem of tools, libraries, and resources that enable developers and researchers to design, train, deploy, and maintain machine learning models efficiently.

The TensorFlow ecosystem is vibrant and dynamic, with a large and active community of developers, researchers, and enthusiasts contributing to its development and evolution. TensorFlow Hub provides a repository of pre-trained models and modules that can be easily reused and integrated into new projects.

TensorFlow supports both CPU and GPU computation, allowing for efficient execution of models on different hardware architectures. It also offers support for distributed computing, enabling training of large-scale models across multiple devices or machines.

NUMPY

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the nd array object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences. NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of and array will create a new array and delete the original.

STREAMLIT

Streamlit is a Python library used for building interactive web applications for data science and machine learning projects. It allows you to create web apps directly from Python scripts, without having to write HTML, CSS, or JavaScript.

Here are some key points about Streamlit:

Simple and Intuitive: Streamlit provides a simple and intuitive way to create web apps. You can use familiar Python syntax and libraries to create interactive elements such as sliders, buttons, and plots.

Rapid Development: With Streamlit, you can quickly prototype and deploy web apps without the need for complex web development frameworks. This makes it ideal for data scientists and machine learning engineers who want to showcase their work or create user-friendly interfaces for their models.

Integration with Data Science Libraries: Streamlit seamlessly integrates with popular data science libraries such as Pandas, Matplotlib, and scikit-learn, allowing you to easily visualize data and build machine learning models within your app.

Automatic Updates: Streamlit automatically updates the app whenever the underlying Python script is modified. This makes the development process more efficient, as you can see the changes in real-time without having to manually reload the page.

Sharing and Deployment: Streamlit apps can be easily shared with others by simply sharing the Python script. Additionally, Streamlit provides built-in deployment options, including Streamlit Sharing, Heroku, and AWS, making it easy to deploy your app to the web

Overall, Streamlit simplifies the process of creating web applications for data science and machine learning projects, allowing you to focus on building and showcasing your models rather than worrying about web development intricacies.

MATPLOTLIB

Matplotlib is a powerful Python library for creating static, interactive, and publication-quality visualizations. It provides a wide range of plotting functions and customization options, making it suitable for various data visualization tasks, from simple line plots to complex 3D graphics.

Key features of Matplotlib include:

Versatile Plotting: Matplotlib offers a rich set of plotting functions for creating a wide variety of plots, including line plots, scatter plots, bar plots, histograms, pie charts, and more. It also supports advanced plot types such as contour plots, surface plots, and 3D plots.

Integration with NumPy and Pandas: Matplotlib seamlessly integrates with NumPy and Pandas, enabling users to visualize data stored in NumPy arrays, Pandas DataFrames, and other data structures. This integration simplifies the process of creating plots from data and facilitates data exploration and analysis.

Interactive Plotting: Matplotlib supports interactive plotting capabilities through the use of interactive backends such as Qt5, GTK, and Tkinter. Interactive plots allow users to zoom, pan, rotate, and interactively explore data in real-time, enhancing the data visualization experience.

Matplotlib is a versatile and powerful library for creating visually appealing and informative plots in Python. Whether you're visualizing data for exploratory analysis, communication, or presentation purposes, Matplotlib provides the tools and flexibility needed to create professional-quality visualizations with ease.

SCIKIT-LEARN

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib. Scikit-learn, often abbreviated as sklearn, is a popular open-source machine learning library for Python. It provides a simple and efficient toolkit for various machine learning tasks, including classification, regression, Clustering, Dimensionality Reduction, Model Selection and preprocessing.

key features and functionalities of scikit-learn:

Consistent API: Scikit-learn provides a consistent and easy-to-use API, making it accessible to both beginners and experienced users. The library follows a uniform interface across different algorithms, making it easy to experiment with different models and techniques.

Wide Range of Algorithms: Scikit-learn offers a comprehensive collection of machine learning algorithms, including linear models, support vector machines, decision trees, random forests, gradient boosting, k-nearest neighbors, clustering algorithms, and more. These algorithms cover a broad spectrum of tasks and are suitable for various types of data and problem domains.

Model Evaluation and Selection: Scikit-learn provides tools for model evaluation and selection, including cross-validation, grid search, and performance metrics such as accuracy, precision, recall, F1-score, and ROC curves. These tools help users assess the performance of their models and fine-tune hyperparameters to improve performance.

Preprocessing and Feature Engineering: Scikit-learn offers a wide range of preprocessing techniques for handling data preprocessing tasks such as scaling, normalization, imputation, encoding categorical variables, feature selection, and transformation. These techniques help prepare the data for training machine learning models and improve their performance.

Integration with NumPy and Pandas: Scikit-learn seamlessly integrates with popular Python libraries such as NumPy and Pandas, allowing users to work with data stored in NumPy arrays, Pandas Data frames, and other data structures. This integration simplifies the process of loading, manipulating, and preprocessing data for machine learning tasks.

6.3 UML DIAGRAMS

INTRODUCTION TO UML

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that has proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using UML helps project teams communicate, explore potential designs and validate the architectural design of the software. UML provides a rich set of graphical notations, enabling practitioners to represent various aspects of software systems in a clear and concise manner. UML encompasses diagrams for capturing the functional requirements and use cases of the system, including Use Case Diagrams and Activity Diagrams. ML enjoys robust tool support, with a plethora of modeling tools available to aid in the creation, editing, and analysis of UML diagrams

6.3.1 Use case Diagram

A use case diagram represents the functionality of the system. Use cases focus on the behavior of the system from an external point of view. Actors are external entities that interact with the system. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. Interaction among actors is not shown on the use case diagram.

Use cases

A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

Actors

An actor is a person, organization, or external system that plays a role in one or more interactions with the system.

The first use case often depends on the outcome of the included use case. This is useful for extracting truly common behaviors from multiple use cases into a single description.

Extend

In another form of interaction, a given use case (the extension) may extend to another. This relationship indicates that the behavior of the extension use case may be inserted in the extended use

case under some conditions. The notation is a dashed arrow from the extension to the extended use case, with the label "extend»".

Generalization

In the third form of relationship among use cases, a generalization/specialization relationship exists. A given use case may have common behaviors, requirements, constraints, and assumptions with a more general use case. In this case, describe them once, and deal with it in the same way, describing any differences in the specialized cases

Associations

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line.

Identified Use Cases

The “user model view “encompasses a problem and solution from the preservative of those individuals whose problem the solution addresses. The view presents the goals and objectives of the problem owners and their requirements of the solution. This view is composed of “use case diagrams”. These diagrams describe the functionality provided by a system to external integrators. These diagrams contain actors, use cases, and their relationships.

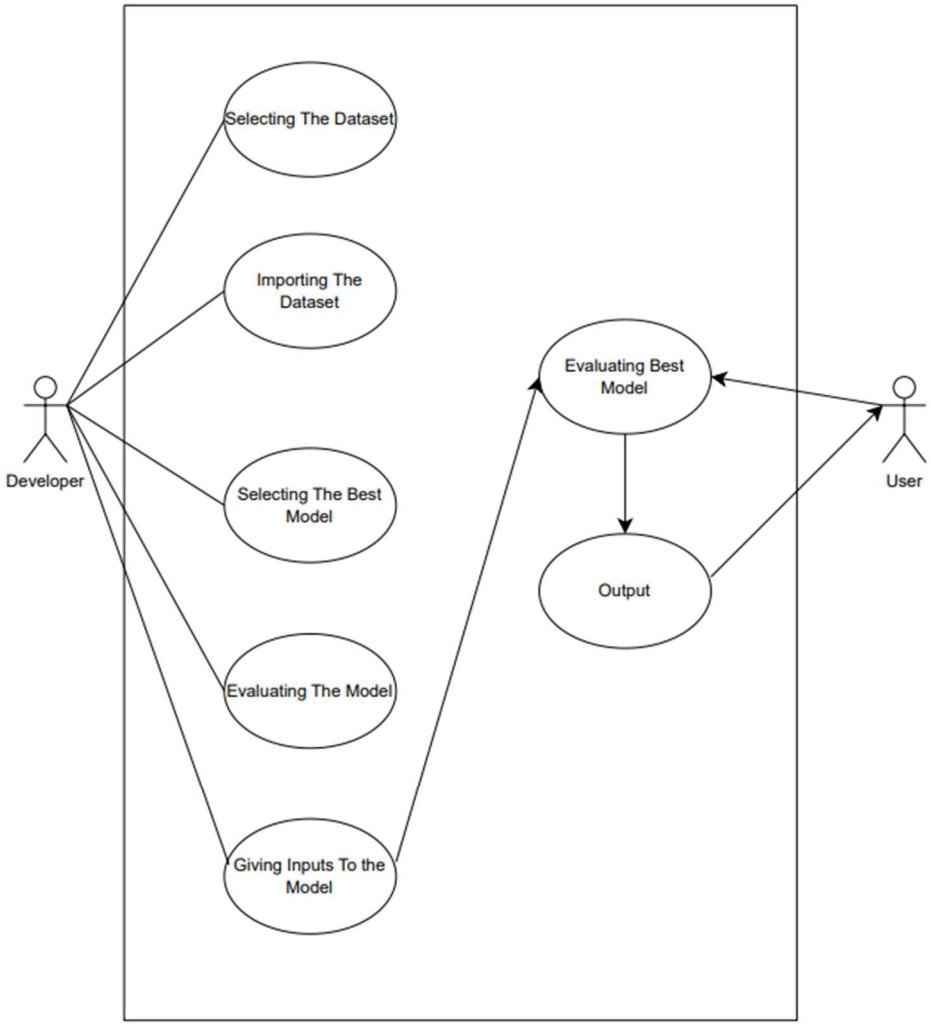


Fig 6.3.1 Use case diagram

6.3.2 CLASS DIAGRAM

The definition, or blueprint, of all objects of a specific type. An object must be explicitly created based on a class and an object thus created is an instance of that class. An object is like a structure, with the addition of method pointers, member access control, and an implicit data member which locates instances of the class (i.e., actual objects of that class) in the class hierarchy. In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes.

The class diagram is the main building block in object-oriented modeling. It is used both for general conceptual modeling of the semantics of the application, and for detailed modeling translating the models into programming code. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain the two parts:

- The upper part holds the name of the class.
- The middle part contains the attributes of the class.

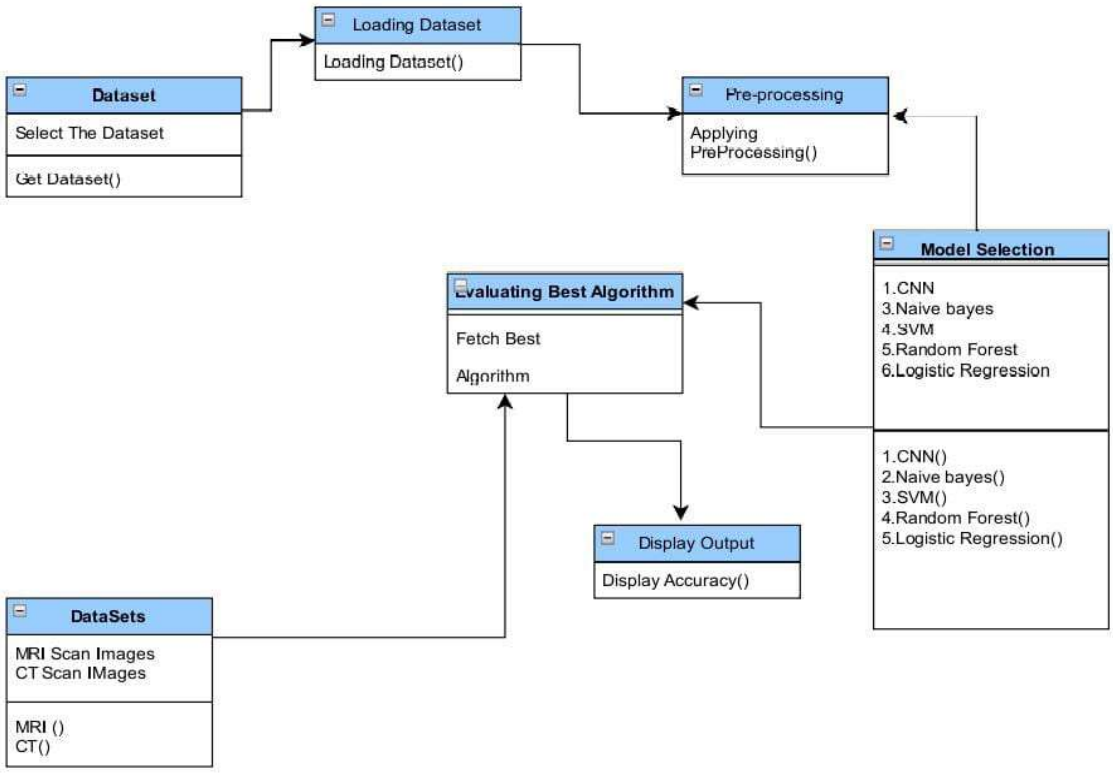


Fig 6.3.2 Class diagram

A class diagram is a visual representation within Unified Modeling Language (UML) that illustrates the static structure of a software system by depicting the classes, their attributes, methods, and relationships. It provides a high-level overview of the system's architecture and the interactions between its components.

6.3.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. If the lifeline is that of an object, it demonstrates a role. Note that leaving the instance name blank can represent anonymous and unnamed instances. In order to display interaction, messages are used. These are horizontal arrows with the message name written above them. Solid arrows with full heads are synchronous calls, solid arrows with stick heads are asynchronous calls and dashed arrows with stick heads are return messages. This definition is true as of UML 2, considerably different from UML 1.x.

Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent those processes that are being performed in response to the message (Execution Specifications in UML).

Objects calling methods themselves use messages and add new activation boxes on top of any others to indicate a further level of processing. When an object is destroyed (removed from memory), an X is drawn on top of the lifeline, and the dashed line ceases to be drawn below it (this is not the case in the first example though). It should be the result of a message, either from the object itself, or another. A message sent from outside the diagram can be represented by a message originating from a filled-in circle (found message in UML) or from a border of sequence diagram (gate in UML)

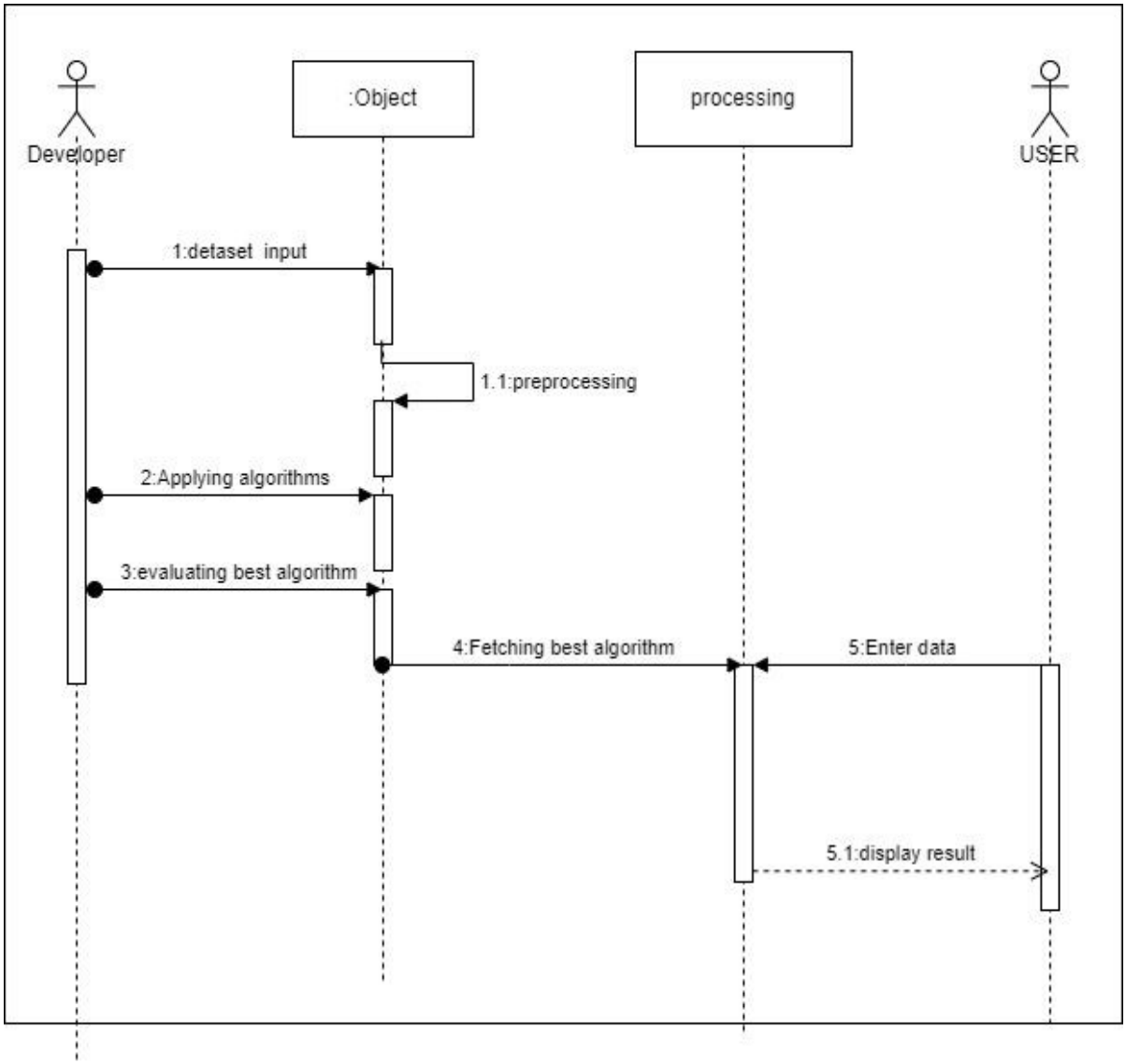


Fig 6.3.3 Sequence diagram

Sequence diagrams are valuable tools for understanding the runtime behavior of a system, identifying potential issues, and validating the correctness of system interactions. They facilitate communication among stakeholders by providing a clear and concise depiction of the system's dynamic behavior.

CHAPTER - 7

IMPLEMENTATION

7.1 CODING

IMPORTING THE LIBRARIES

```
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from keras.applications import EfficientNetB0
from keras.models import Model
from keras.layers import GlobalAveragePooling2D
from keras.preprocessing import image
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

PREPOCESSING

```
def load_images(directory):
    images = []
    labels = []
    for label, category in enumerate(os.listdir(directory)):
        category_path = os.path.join(directory, category)
```

```
    for filename in os.listdir(category_path):
        img = image.load_img(os.path.join(category_path, filename), target_size=(224, 224))
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array = img_array / 255.0
        images.append(img_array)
        labels.append(label)
  return np.vstack(images), np.array(labels)
```

LOADING THE DATASET

```
image_directory = r"C:\Users\Y HARSHITHA\Desktop\project\myenv\brain_mri_scan_images"
images, labels = load_images(image_directory)
```

FEATURE EXTRACTION

```
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape= (224,224,3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
feature_extractor = Model(inputs=base_model.input, outputs=x)


features = feature_extractor.predict(images)
```

SPLITTING THE DATA

```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
```

RANDOM FOREST MODEL

```
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
rf_accuracy = accuracy_score(y_test,y_pred)
print("Accuracy score of random forest model is",rf_accuracy*100)
```

NAIVE BAYES MODEL

```
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred = nb_classifier.predict(X_test)
nb_accuracy = accuracy_score(y_test,y_pred)
print("Accuracy score of guassianNB model is",nb_accuracy *100)
```

SVM MODEL

```
svclassifier = SVC(kernel='rbf')
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)
sv_accuracy = accuracy_score(y_test,y_pred)
print("Accuracy score of svm model is",sv_accuracy*100)
```

LOGISTIC REGRESSION MODEL

```
lr_classifier = LogisticRegression()
lr_classifier.fit(X_train, y_train)
y_pred = lr_classifier.predict(X_test)
lr_accuracy = accuracy_score(y_test, y_pred)
```

```
print("Accuracy score of logistic regression model is",lr_accuracy*100)
```

DATA VISUALIZATION

```
accuracies = {'Random Forest': rf_accuracy, 'Naive Bayes': nb_accuracy, 'SVM': sv_accuracy, 'Logistic Regression': lr_accuracy}

plt.bar(accuracies.keys(), accuracies.values())

plt.xlabel('Classifier')

plt.ylabel('Accuracy')

plt.title('Accuracy Comparison of Classifiers')

plt.savefig('bar_graph.png')

plt.show()
```

NEW IMAGE PREPROCESSING

```
def preprocess_image(image):
    img = image.convert('RGB')
    img = img.resize((224, 224))
    img_array = np.array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = img_array / 255.0
    return img_array


label = {
        0: 'alzheimer',
        1: "brainTumor",
        2: "epilepsy",
```

```
    3: 'normal',
    4: 'Parkinson',
    5: 'none of the above'
}
```

NEW IMAGE PREDICTION

```
def predict(image):
    new_image = preprocess_image(image)
    new_features = feature_extractor.predict(new_image)
    predicted_class = rf_classifier.predict(new_features)[0]
    return predicted_class
```

INTERFACE

```
import streamlit as st
from PIL import Image
st.title("Brain Diseases Detection")
uploaded_file = st.file_uploader("Upload an image", type=["jpg", "png", "jpeg"])
if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.image(image, caption='Uploaded Image', use_column_width=True)
    predicted_class = predict(image)
    st.write("Predicted Disease Class:", label[predicted_class]
```

CHAPTER - 8

TESTING

Introduction

After completing software development, the next complicated and time-consuming step is software testing. Only the development team knows how far the user requirements have been met during this process, and so on. This phase ensures software quality and provides the final review of specification, design, and coding. The increasing convenience of software as a system, as well as the cost associated with software failures, are driving forces for thorough testing.

Objectives of Testing:

These are the rules that account for testing objectives:

- Testing is the process of running a programme in order to find errors.
- A good test case is one that has a high chance of detecting an undiscovered error.
- The project's testing procedures are carried out in the order listed below.
- System testing is performed to verify the server's name of the machines that are linked between the customer and the executive.
- The product information provided by the company to the executive is validated using the centralised data store.
- System testing is also performed to ensure that the executive is available to connect to the server.
- The server's name authentication is checked, as is the customer's availability.
- Proper communication chat line viability is tested, and the chat system is made to work properly.
- Mail functions are validated against user concurrency and customer mail date.

Black box testing and white box testing are two common techniques used in software testing to ensure the quality and functionality of a software product.

WHITE BOX TESTING:

- This is unit testing method where is unit will be taken at a time and tested thoroughly at a statement level to find the maximum level errors.
- We have tested step wise every piece of code, taking care that every statement in the code
- Is executed at least once.

- The white box testing is also called glass box testing.
- In white box testing, the tester has access to the internal structure and code of the software. Test cases are designed based on the knowledge of the software's internal logic, code paths, and structure.
- The focus is on testing the internal workings of the software, including code coverage, branch coverage, and path coverage.
- Testers examine the structure of the code and its execution paths to ensure thorough testing.
- Examples of white box testing techniques include statement coverage, decision coverage, and condition coverage.

BLACK BOX TESTING:

This testing method a model a single unit and checks the unit at interface and communication with other models rather getting g into details levels. Here the model will be treated as a black box that take input and generates the output. Output of given input combinations are forwarded to other models.

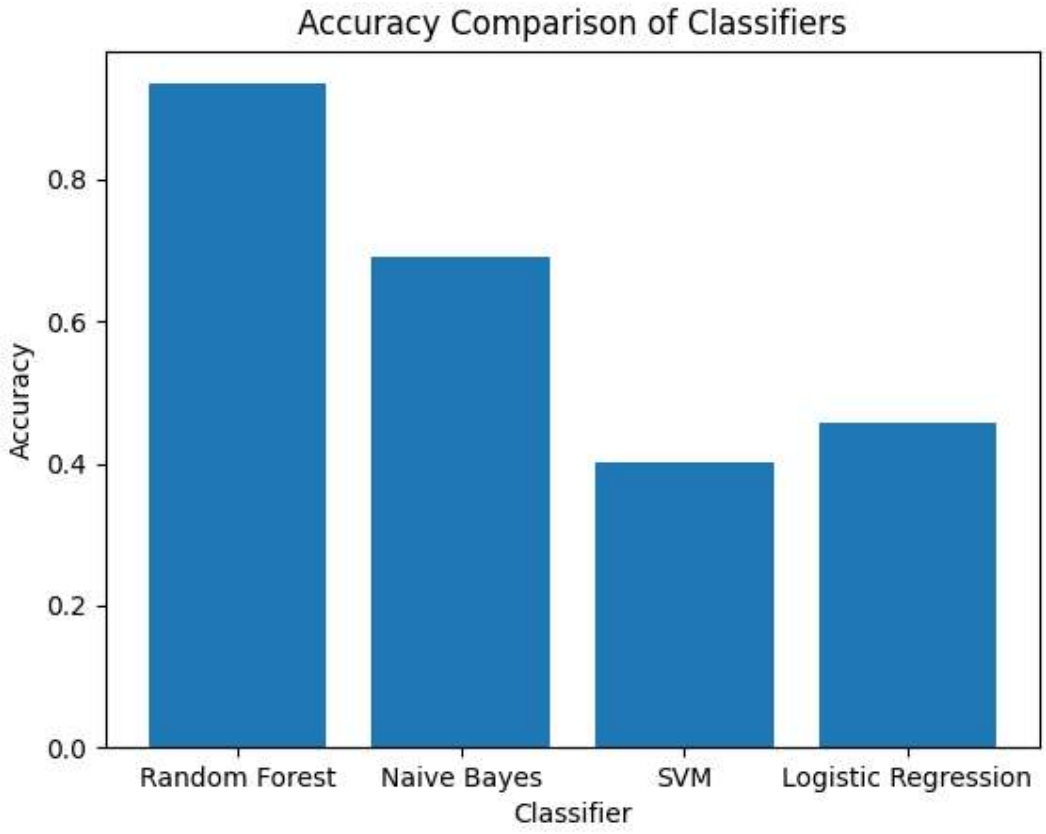
- In black box testing, the tester evaluates the functionality of the software without knowing its internal structure or code.
- Test cases are designed based on the specifications and requirements of the software.
- The focus is on testing the software's inputs and outputs, as well as its behavior in different scenarios.
- Testers treat the software as a black box, only interacting with its external interfaces and functionality.
- Examples of black box testing techniques include equivalence partitioning, boundary value analysis, and state transition testing.

Both black box and white box testing are important in software development, and they complement each other by providing different perspectives on testing the software. Black box testing ensures that the software meets the specified requirements and behaves correctly from the user's perspective, while white box testing verifies the internal correctness of the software's implementation.

CHAPTER - 9
RESULTS AND DISCUSSION

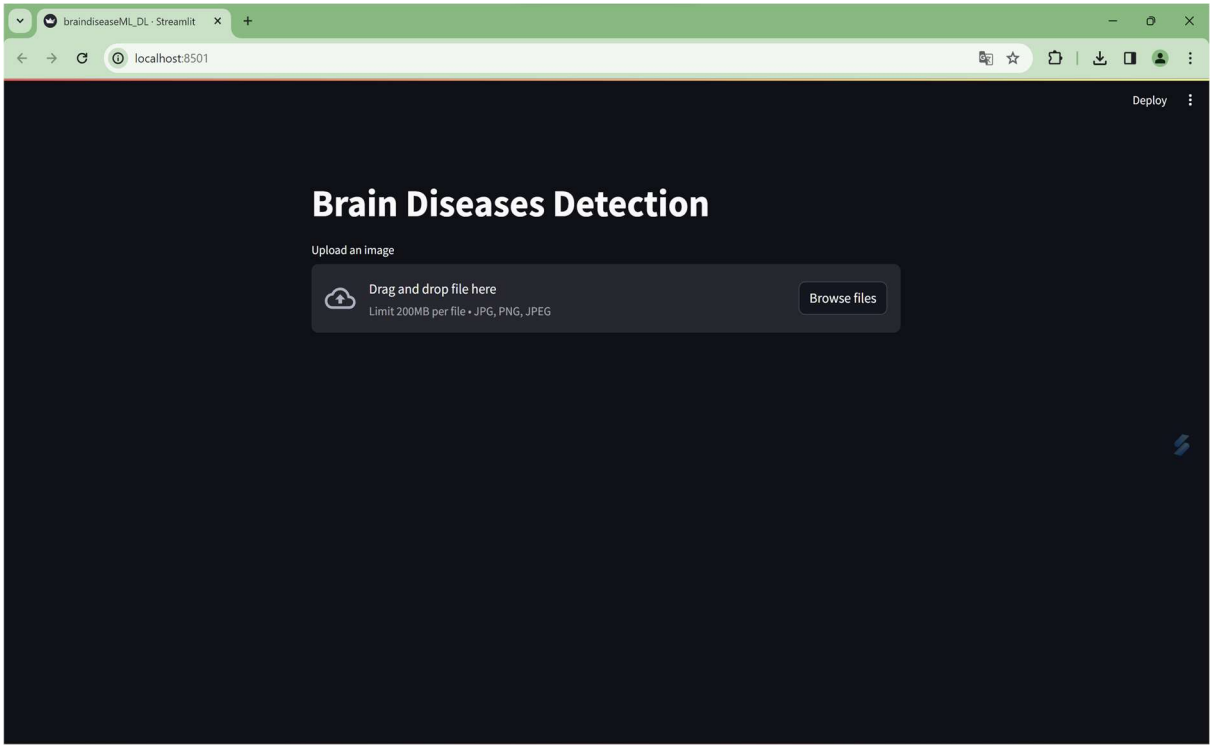
```
17/17 [==============================] - 15s 729ms/step
Accuracy score of random forest model is 93.45794392523365
Accuracy score of guassianNB model is 69.1588785046729
Accuracy score of svm model is 40.18691588785047
Accuracy score of logistic regression model is 45.794392523364486
```

Accuracies of different Machine Leaning Models which are used for brain diseases detection

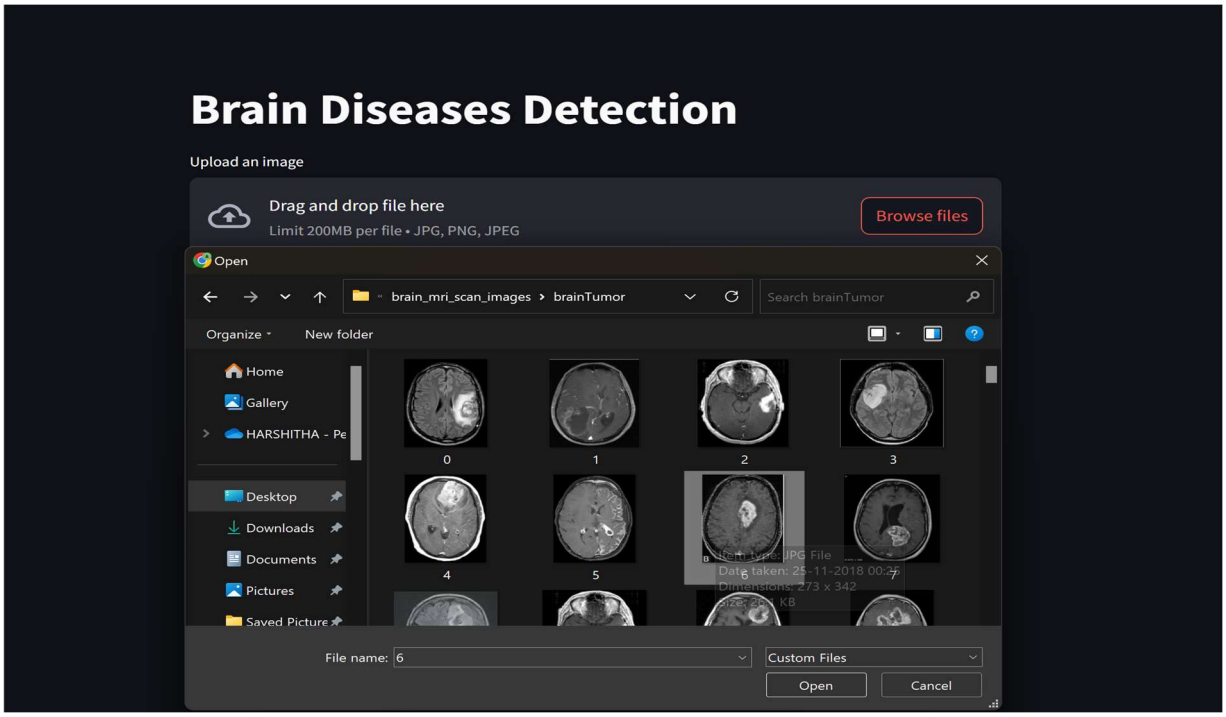


Graph shows the comparison between the different Machine Learning algorithms

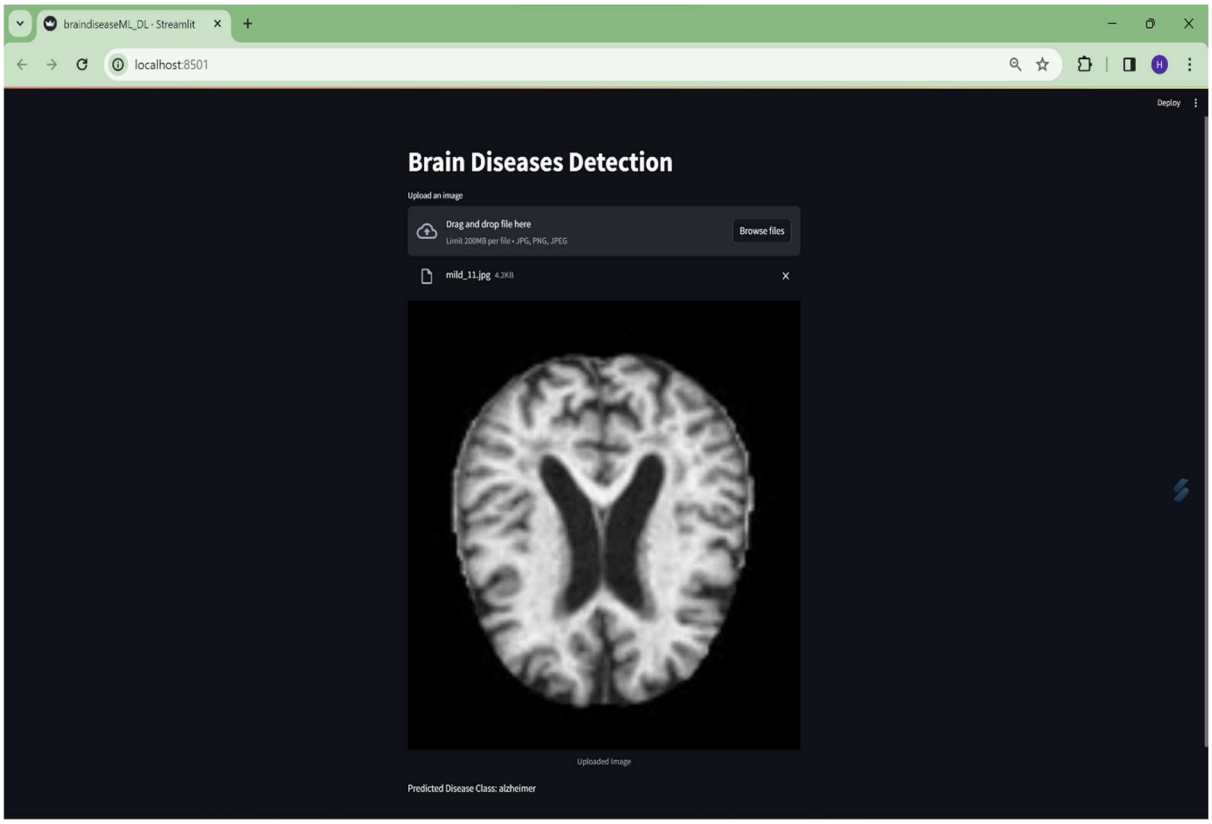
INTERFACE: This webpage offers a user-friendly interface for detecting brain diseases.



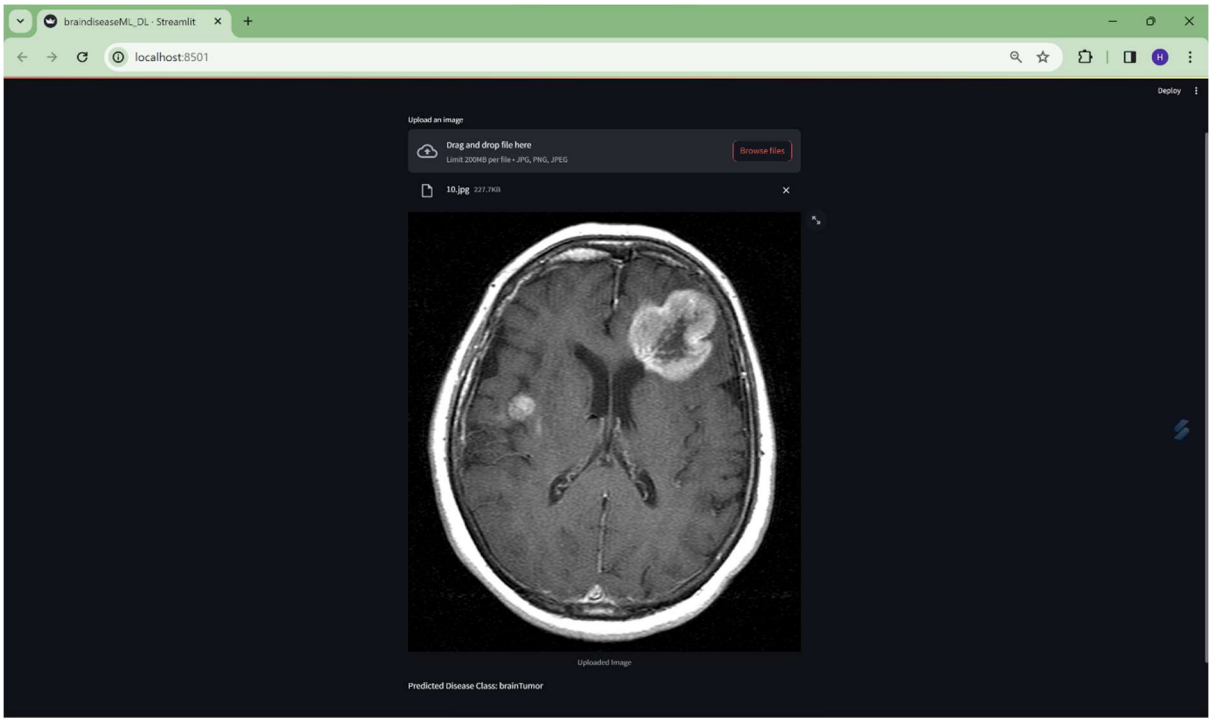
INPUT DATA: This allows users to input random image for predicting brain diseases, including brain tumor, Alzheimer's, Parkinson's, epilepsy, or normal brain function.



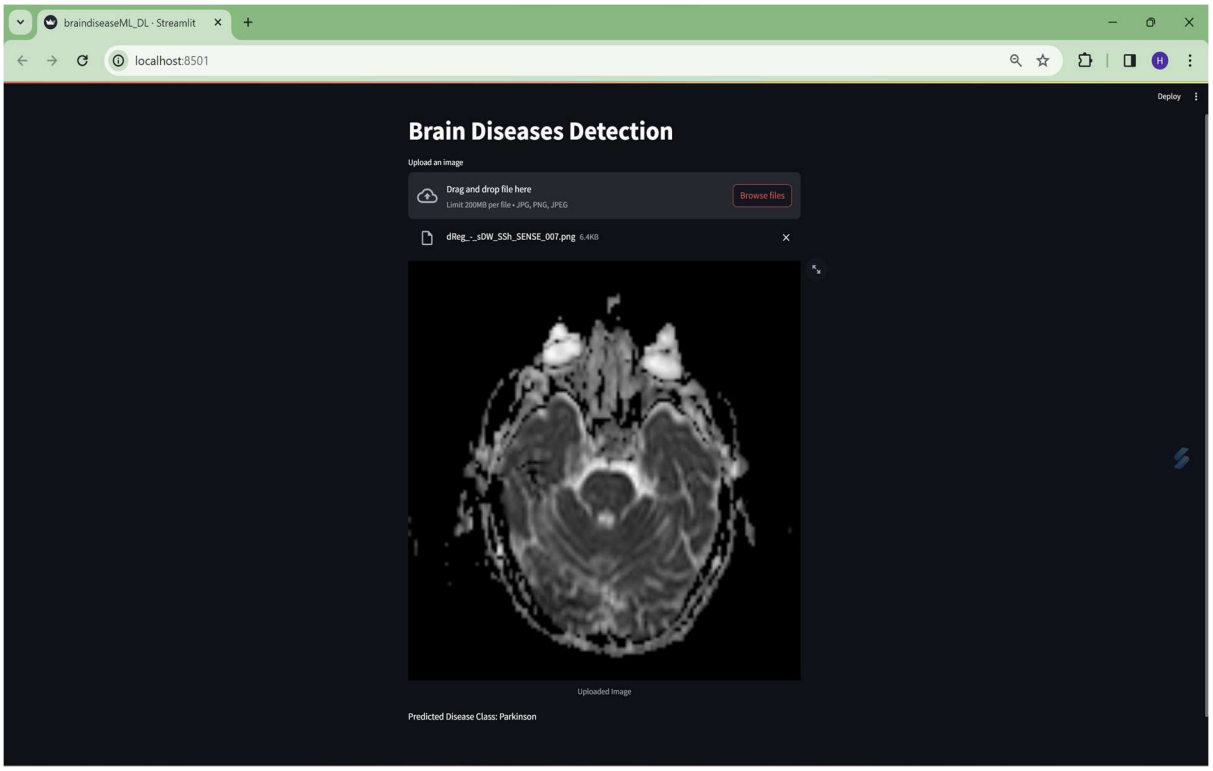
OUTPUT 1: On this screen, when the user submitted an image of a alzheimer, the prediction also identified it as a alzheimer.



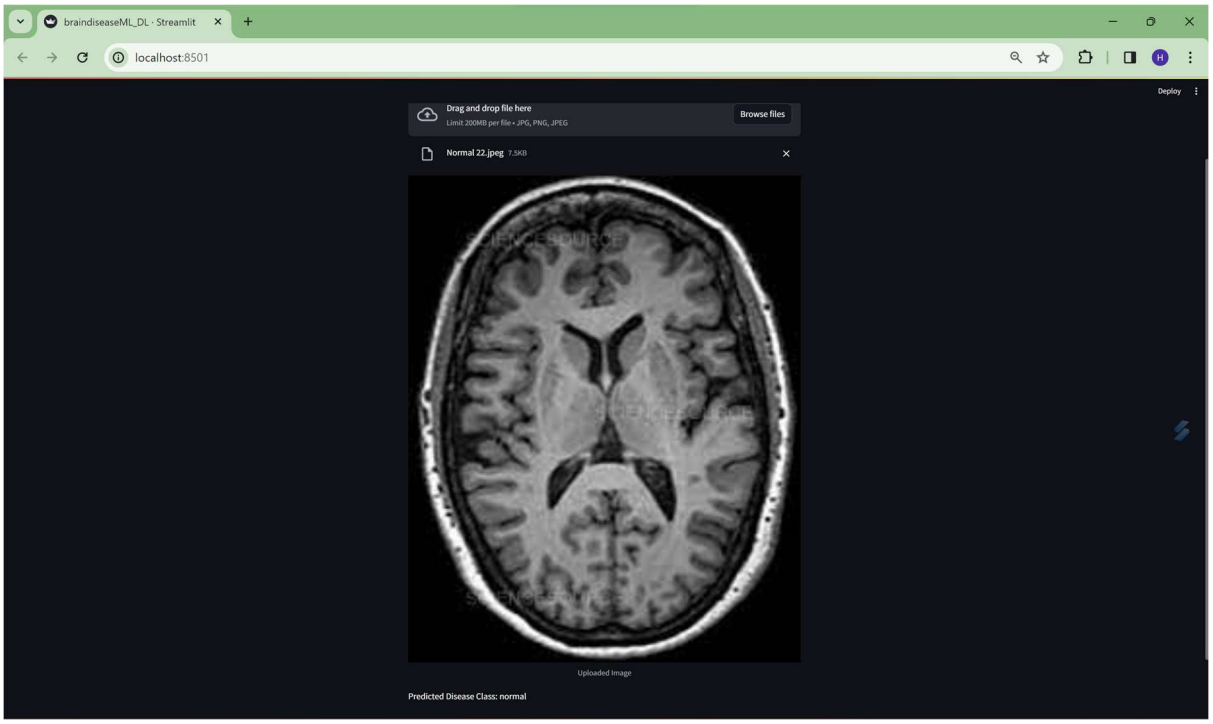
OUTPUT 2: On this screen, when the user submitted an image of a brain tumor, the prediction also identified it as a brain tumor.



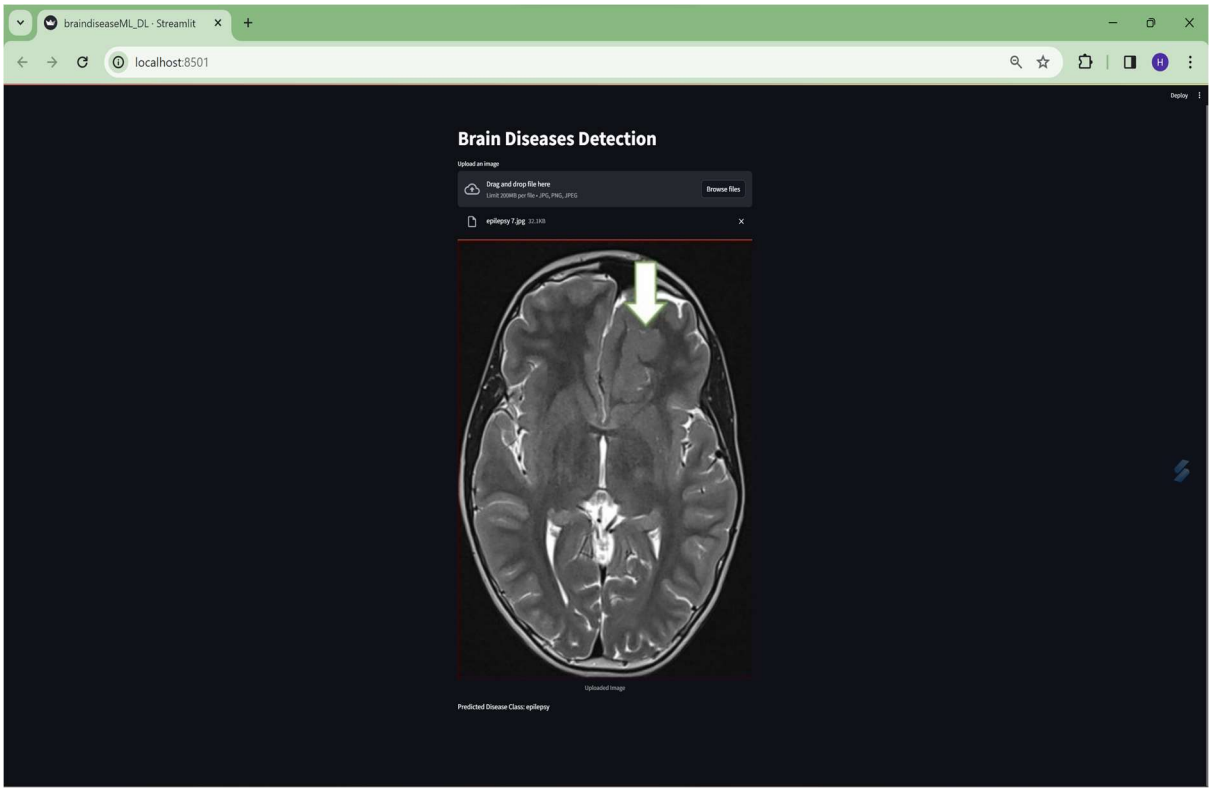
OUTPUT 3: On this screen, when the user submitted an image of a parkinson, the prediction also identified it as a parkinson disease.



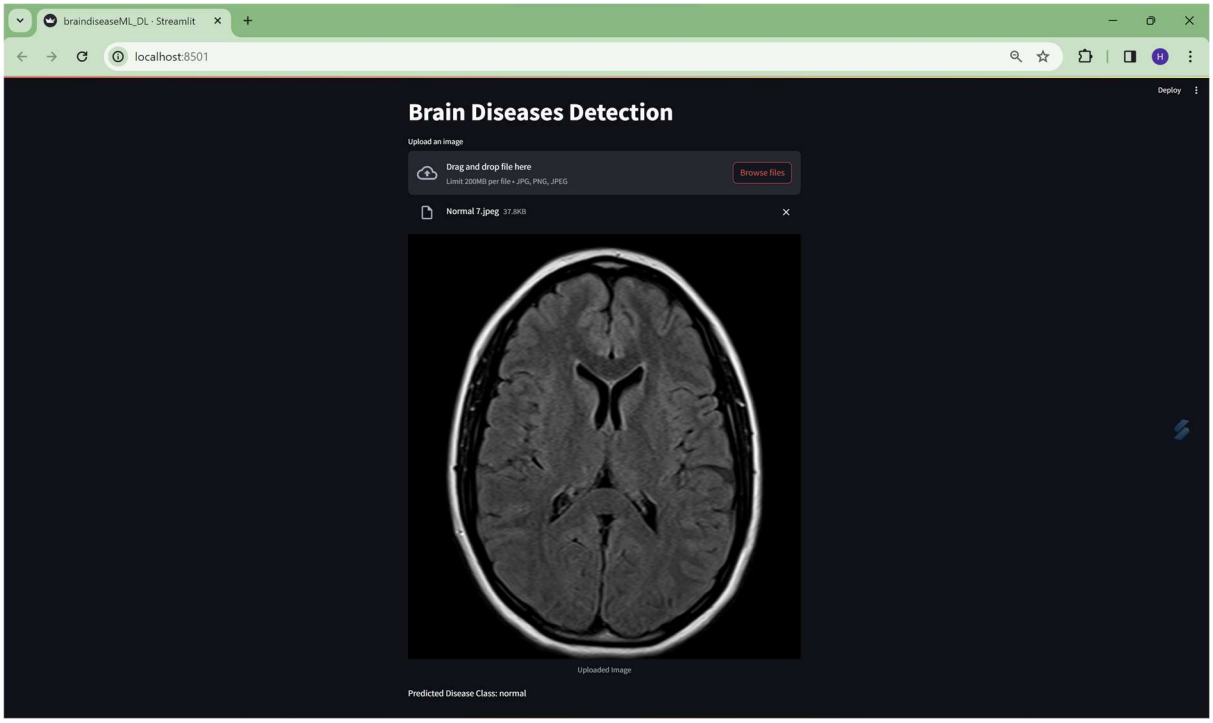
OUTPUT 4: On this screen, when the user submitted an image of a normal, the prediction also identified it as a normal



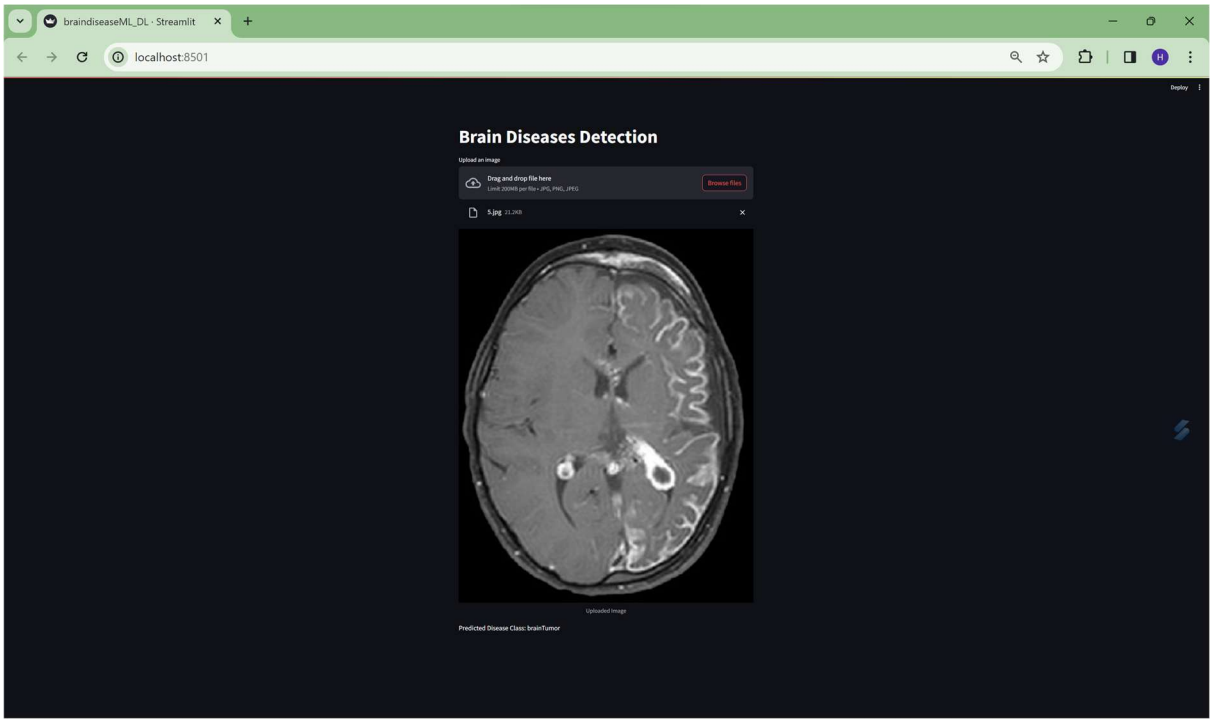
OUTPUT 5: On this screen, when the user submitted an image of a epilepsy disease, the prediction also identified it as a epilepsy disease.



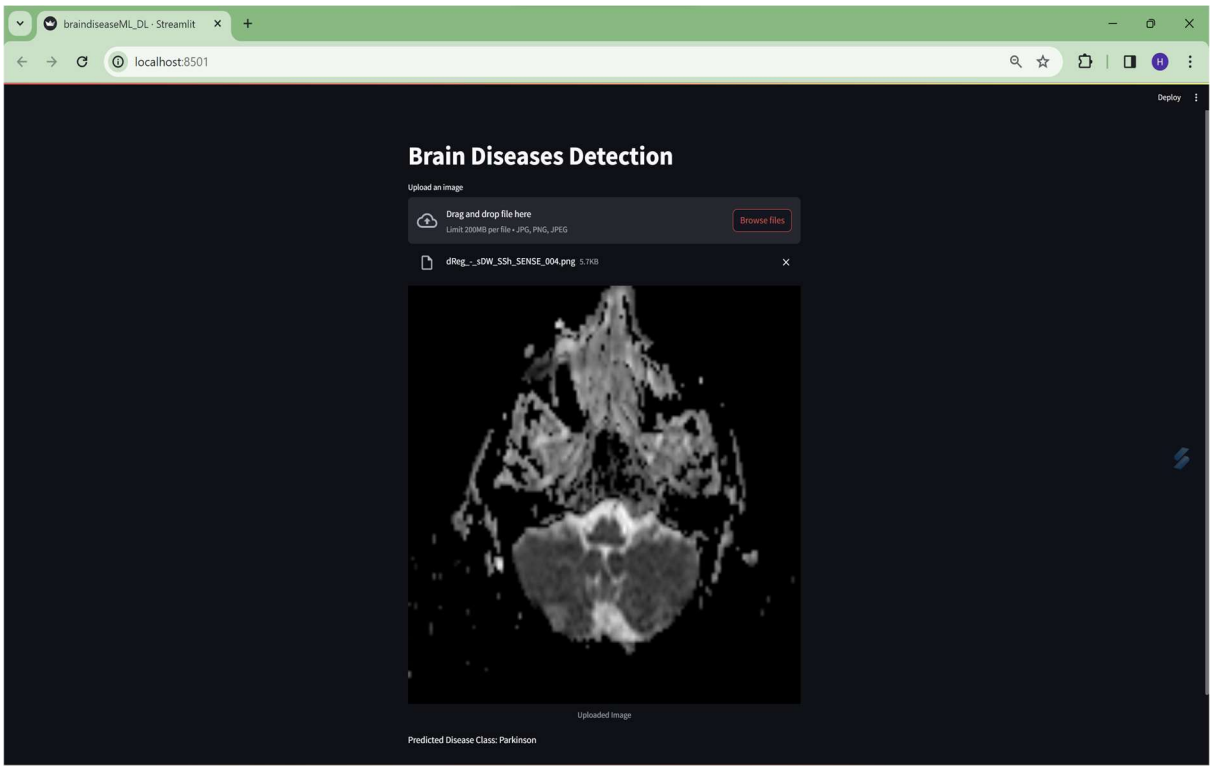
OUTPUT 6: On this screen, when the user submitted an image of a normal, the prediction also identified it as a normal



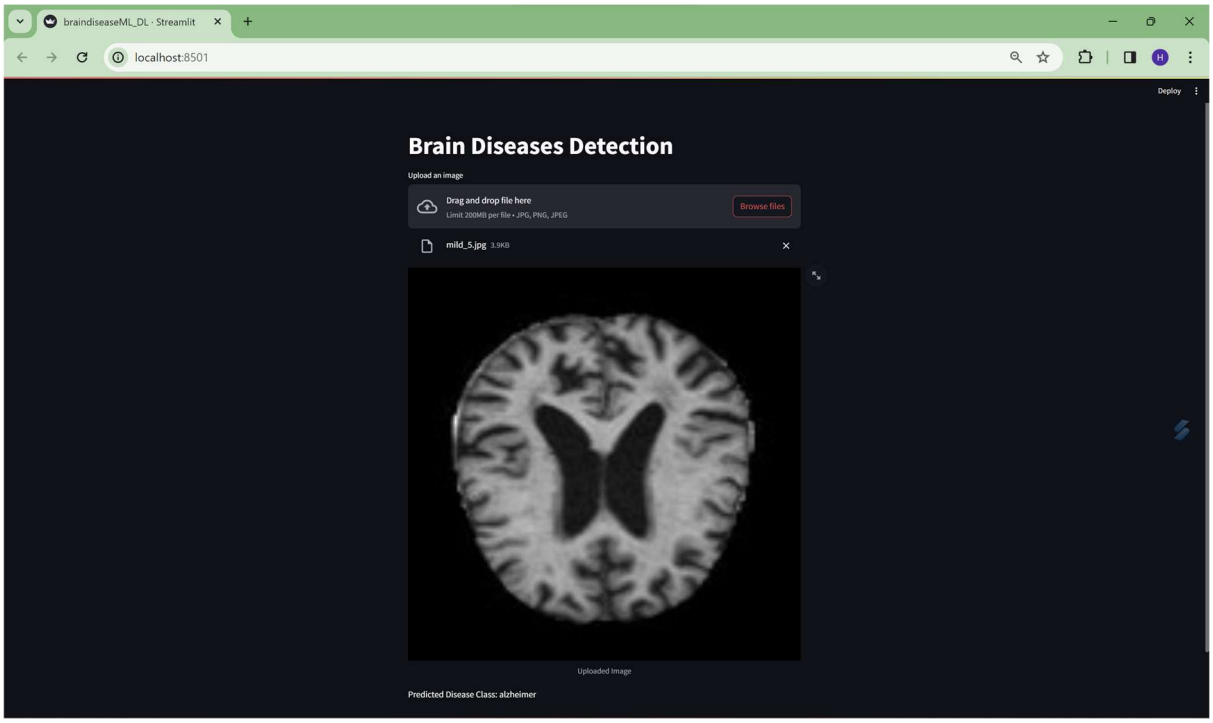
OUTPUT 7: On this screen, when the user submitted an image of a brain tumor, the prediction also identified it as a brain tumor.



OUTPUT 8: On this screen, when the user submitted an image of a parkinson, the prediction also identified it as a parkinson disease.



OUTPUT 9: On this screen, when the user submitted an image of a alzheimer, the prediction also identified it as a alzheimer.



It is already proven that Random Forest gave 93% but in this random analysis of algorithms we had new results and conclusion in which “Random Forest” performed best. This algorithm performed best while predicting the new results and also gave the best accuracy score compared to the other algorithms.

CHAPTER - 10
CONCLUSION

We have presented a survey on the four most dangerous brain disease detection processes using machine and deep learning. The survey reveals some important insights into contemporary ML/DL techniques in the medical field used in today's brain disorder research. With the passage of time, identification, feature extraction, and classification methods are becoming more challenging in the field of ML and DL. Researchers across the globe are working hard to improve these processes by exploring different possible ways. One of the most important factors is to improve classification accuracy. For this, the number of training data needs to be increased because the more the data is involved, the more accurate the results will be.

The use of hybrid algorithms and a combination of supervised with unsupervised and ML with DL methods are promising to provide better results. Even, various fine tunings can sometimes offer promising improvements, 3D-CNN is used first to extract primary features, and next, instead of the general FC layer, the FSBi-LSTM is used. This slight change in a part of the system eventually resulted in superior performances. Based on the discussion on different types of brain disease data sources and feature extractions methods, it is apparent that the accuracies differ based on different classifiers used and feature extraction processes applied in the systems.

To uncover the limitations of existing ML/DL-based approaches to detect various brain diseases, the paper provides a discussion focusing on a set of open research issues. To design effective AI systems for medical applications, the inclusion of XAI approaches is the ultimate necessity. This will help medical professionals to build their confidence and AI-based solutions will be transformed into clinical practice in the treatment of patients with brain disorders. We came to know that quality of training data and interoperability are also major concerns to develop ML and DL-based solutions. It is yet to be determined whether we will be able to have sufficient training data without compromising the performances of DL/ML algorithms. To make ML/DL-based solutions more practical, various other issues such as resource efficiency, large-scale medical data management, and security and privacy should be addressed well. This survey is expected to be useful for researchers working in the area of AI and medical applications in general and ML/DL-based brain disease detection in particular.

CHAPTER - 11

REFERENCES

[1] G. Dornhege, J. D. R. Millan, T. Hinterberger, D. McFarland, and K. Moller, Towards Brain-Computing Interfacing. Cambridge, MA, USA: MIT Press, 2007.

[2] J. Paul and T. S. Sivarani, ‘‘Computer aided diagnosis of brain tumor using novel classification techniques,’’ J. Ambient Intell. Humanized Comput., pp. 1–11, Jul. 2020.

[3] J. Godyń, J. Jończyk, D. Panek, and B. Malawska, ‘‘Therapeutic strategies for Alzheimer’s disease in clinical trials,’’ Pharmacol Rep., vol. 68, no. 1, pp. 127–138, Feb. 2016.

[4] R. T. Merrell, ‘‘Brain tumors,’’ Dis Mon., vol. 58, no. 12, pp. 678–689, Dec. 2012.

[5] Y. M. Hart, ‘‘Diagnosis and management of epilepsy,’’ Medical, vol. 44, no. 8, pp. 488–494, Aug. 2016.

[6] D. Calne, ‘‘Is idiopathic parkinsonism the consequence of an event or a process?’’ Neurology, vol. 44, no. 1, pp. 5–10, Jan. 1994.

[7] N. K. Chauhan and K. Singh, ‘‘A review on conventional machine learning vs deep learning,’’ in Proc. Int. Conf. Comput., Power Commun. Technol. (GUCON), Sep. 2018, pp. 347–352.

[8] Alzheimer’s Disease Neuroimaging Initiative (ADNI). Accessed: Oct. 15, 2020. [Online]. Available: http://adni.loni.usc.edu/

[9] D. L. Beekly, E. M. Ramos, G. van Belle, W. Deitrich, A. D. Clark, M. E. Jacka, and W. A. Kukull, ‘‘The national alzheimer’s coordinating center (NACC) database: An alzheimer disease database,’’ Alzheimer Disease Associated Disorders, vol. 18, no. 4, pp. 270–277, 2004.

[10] D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, and R. L. Buckner, ‘‘Open access series of imaging studies (OASIS): Crosssectional MRI data in young, middle aged,

nondemented, and demented older adults,'' J. Cognit. Neurosci., vol. 19, no. 9, pp. 1498–1507, Sep. 2007

[11] F. Ibrahim, S. Abd-Elateif El-Gindy, S. M. El-Dolil, A. S. El-Fishawy, E.-S.-M. El-Rabaie, M. I. Dessouky, I. M. Eldokany, T. N. Alotaiby, S. A. Alshebeili, and F. E. Abd El-Samie, ''A statistical framework for EEG channel selection and seizure prediction on mobile,'' Int. J. Speech Technol., vol. 22, no. 1, pp. 191–203, Jan. 2019.

[12] T. N. Alotaiby, S. A. Alshebeili, F. M. Alotaibi, and S. R. Alrshoud, ''Epileptic seizure prediction using CSP and LDA for scalp EEG signals,''Comput. Intell. Neurosci., vol. 2017, pp. 1–11, Oct. 2017.

[13] D. Cho, B. Min, J. Kim, and B. Lee, ''EEG-based prediction of epileptic seizures using phase synchronization elicited from noise-assisted multivariate empirical mode decomposition,'' IEEE Trans. Neural Syst. Rehabil. Eng., vol. 25, no. 8, pp. 1309–1318, Aug. 2017.

[14] M. H. Myers, A. Padmanabha, G. Hossain, A. L. de Jongh Curry, and C. D. Blaha, ''Seizure prediction and detection via phase and amplitude lock values,'' Frontiers Human Neurosci., vol. 10, p. 80, Mar. 2016.

[15] I. A. Illan, J. M. G´orriz, J. Ram´ırez, and A. Meyer-Base, "Spatial component analysis of MRI data for Alzheimer's disease diagnosis: A Bayesian network approach," Frontiers Comput. Neurosci., vol. 8, 2014, Art. no. 156.

[16] P. Markiewicz, J. Matthews, J. Declerck, and K. Herholz, "Robustness of multivariate image analysis assessed by resampling techniques and applied to FDG-PET scans of patients with Alzheimer's disease," NeuroImage, vol. 46, pp. 472–485, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6WNP-4VFK7X3-3/2/e7833cb1d62f8e28326352e45981d00

[17] F. Polatet al., "Computer based classification of MR scans in first time applicant Alzheimer patients," Current Alzheimer Res., vol. 9, no. 7, pp. 789–794, Sep. 2012.

[18] J. M. Gorriz et al., "A semi-supervised learning approach for model selection based on class-hypothesis testing," Expert Syst. Appl., vol. 90, pp. 40–49, 2017.