International Journal of Image and Graphics
© World Scientific Publishing Company

Simplifying MRI Tumor Classification: The Effectiveness of Basic Machine Learning Algorithms in High-Dimensional Data

Nadimpalli Ujwal Srimanth Varma
Department of Computer Science and Engineering, Amrita School of Computing, Coimbatore,Amrita Vishwa Vidyapeetham 64112, India[*]
first_author@university.edu

SECOND AUTHOR
Group, Laboratory, Address
City, State ZIP/Zone, Country
second_author@group.com

SECOND AUTHOR
Group, Laboratory, Address
City, State ZIP/Zone, Country
second_author@group.com

SECOND AUTHOR
Group, Laboratory, Address
City, State ZIP/Zone, Country
second_author@group.com

SECOND AUTHOR
Group, Laboratory, Address
City, State ZIP/Zone, Country
second_author@group.com

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Image classification plays a pivotal role in various fields, including healthcare, agriculture, monitoring, and safety. While deep learning (DL) methods are frequently employed for this purpose, traditional machine learning (ML) techniques can also yield effective results. This paper explores the challenges and findings related to MRI scan tumor classification. Our approach involves converting image data into numerical form by extracting the RGB values of each pixel. Given that standard ML techniques are not equipped to handle 3D or 4D data, we simplify the process by reducing the dimensionality of the data, which is often high. Through meticulous experimentation, we address

[*]

class imbalance using the Synthetic Minority Over-sampling Technique (SMOTE), fine-tune hyperparameters, and fit the models effectively to the data. The results reveal several significant patterns, including the variability of hyperparameter tuning, the limitations of probabilistic models, the advantages of ensemble and tree-based methods, and the unexpectedly strong performance of K-Nearest Neighbors (KNN). We attribute KNN's success to its lazy learning nature. This study highlights the potential of basic ML algorithms in image classification tasks and elucidates the factors influencing their performance, providing valuable insights for future research in medical image analysis.

Keywords: Image Classification, Machine Learning, Tumor Classification, MRI Scans, Dimensionality Reduction, K-Nearest Neighbors

1. Introduction

Tumors are clusters of divergent cells that grow erroneously. Such cells grow inside or near the brain, forming a brain tumor. They may begin in the brain or spread from other parts of the body to the brain. The cells that cause these brain tumors primarily categorize their type.While some of them are not dangerous, others can be cancerous and can affect an individual's memory, speech, thought process, mobility, and different organs' functionalities.[1]It is important to diagnose and treat these tumors promptly. With a plethora of brain tumor types, it is grueling to classify them precisely. Given this prospect and the costs involved in diagnosis, shrewd ML and DL models established themselves as utilitarians. While trained over large data sets, these models help determine patterns that might not be visible to radiologists. These models have demonstrated their ability to detect early-stage brain tumors and assist in timely treatment.

Image processing is the process of converting image data into numerical digital data in order to perform various tasks such as recognition, classification, and so on. The transformed digital data makes it easier to apply a wide range of algorithms and returns more accurate results for extracting useful information or improving overall quality. Grayscale and color are the two broad categories into which images fall. To make the algorithm suitable, we should pre-process the data based on the type and resolution of the image. Because it is unstructured, image recognition in ML is an arduous task. We refer to data without a predefined structure or format as unstructured data. Unlike structured data, image data is high-dimensional, increasing the complexity and making it difficult for the ML algorithms to train on. This formidable challenge led to the emergence of a need for transformation. The transformation into numerical data is cardinal, allowing for effective interpretation and prediction by the ML models.

A collection of magnetic resonance imaging (MRI) scan images supports multiclass classification in the used dataset. The dataset consists of 4 types of classes; refer to Figure 1.MRI, in particular, is frequently used for brain imaging because it can produce detailed images in three views: the axial, coronal, and sagittal views .[2]We represent the images as matrices of pixel intensities. We used a resolution of 224 x 224 pixels for the grayscale MRI scan images, which included a 3D shape and nearly 150,000 columns after flattening, resulting in high computational demands.

Such high-dimensional data is a deterrent as it can lead to overfitting of the model, where the model learns more from the noise than from the underlying patterns, causing the curse of dimensionality.

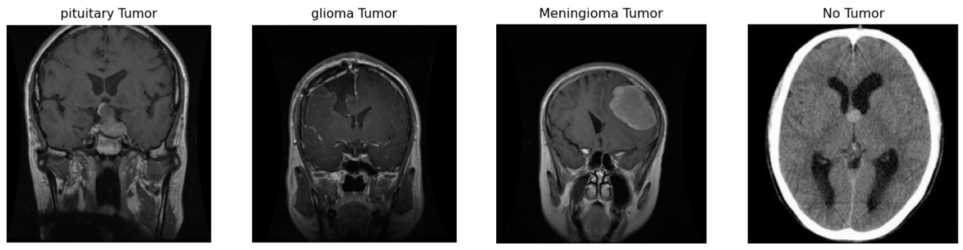


Fig. 1. Sample MRI scan images depicting different types of tumors.

We use dimensionality reduction methods to represent the same data in fewer columns. In the context of MRI scans, these transformations are cardinal because these images have a wealth of information in the form of subtle variations in the pixel intensities. Extracting useful information from these images is a critical step for the ML models' efficiency in detecting tumors. People often overlook the computational power of ML algorithms in the context of image classification. The misconception that these algorithms are not the optimal choice for image data is widespread. This research aims to broaden the trade-off between choosing ML algorithms over DL algorithms for the image classification task. The KNN algorithm with optimized parameters can achieve outstanding accuracy. The section that precedes the rest of the article is the literature review, followed by data preprocessing and class imbalance handling. The article further elaborates on model selection and training, fine tuning, and hyperparameter optimization. The article's conclusion signifies the end.

2. Related Works

Researchers have conducted several studies on image recognition, specifically focusing medical image classification such as brain tumors and breast cancer classification.Coming to Breast cancer ,Globally the death rate of women is more due to this disease.On Breast Cancer detection Many methodologies were proposed. A paper which solely discusses of improving the performance of breast cancer detection using mammogram images was proposed [3].where the preprocessing is done using the techniques like image denoising and enhancement.Edge segmentation is used with the help of Gabor cut and canny edge algorithms. The results shown an improved performance among the images.[3]

Similarly for Brain Tumor Classification,Both ML and DL techniques have proposed numerous solutions. This section summarizes the key findings and approaches from various research papers in this field. Generally, pure ML models are not sufficient for achieving state-of-the-art performance in image recognition tasks. The reason is that traditional ML algorithms, such as SVMs, decision trees, or random forests, typically require hand-crafted feature extraction methods to transform raw pixel data into a suitable feature representation. The complex task of hand-crafting features for image data leads to the preference of DL over ML due to its limitations in scalability and performance.

Some papers propose methodologies that rely solely on ML without any DL approach. One such paper is using algorithms like SVM, Naive Bayes, and Ensemble, where in most cases Ensemble has outperformed the other models with an accuracy of 97%. The SVM, in their work, resulted in an accuracy of 95.6%; they later upgraded the model using the 5-layered Conventional Neural Network (CNN) technique.[4] A powerful approach using Ensemble Method XGBoost was also proposed in [5], where they achieved an accuracy of 94%. They first enhanced the MRI images using an adaptive median filter and reduced the feature dimension using principal component analysis (PCA), decreasing the variance from 85% to 95%. However, the dataset only consists of two classes (tumor and no-tumor), limiting the ability to classify specific tumor types .[5,6]

In [7], a combination of both ML and DL is used, and the dataset used is "BRATS 2018," which contains two labels: malignant and benign. The implementation of this work is done using the Ensemble method, where a median filter and a denoising CNN (DnCNN) are used to get rid of noise in the pre-processing step. They used transfer learning for feature extraction and an ML classifier that includes a combination of KNN, Decision Tree, and SVM, resulting in a final accuracy of a whopping 99.2%. A similar dataset is also used in [8], where the researchers have analyzed the working of ML classifiers on this dataset, and they got an accuracy of 97.4% for the KNN classifier and also 98.8% for the DT+DNN. The process includes sending the Gray Level Co-occurrence Matrix (GLCM) features from the MR images to the ML classifiers. The performance of different ML algorithms is tested on 116 MR images of both benign and malignant tumors .[8]

In [7], a combination of both ML and DL is used, and the dataset used is "BRATS 2018," in which DL is the most dominant in image recognition tasks due to its ability to learn rich feature representations directly from raw data, which is a key advantage. Transfer learning is a DL technique in which a model designed for one task can be repurposed as the starting point for another model. In a paper [9], this method was also suggested. They trained EfficientNet-B2 with a SoftMax activation function and used global average pooling for the model. They have successfully classified brain tumors with an accuracy of 98.78%.

In [10,11], a method was suggested that was based on the BCM-CNN model. This model was later made better as Inception-ResnetV2, GoogLENet, and ResNet-18 with custom CNN (GoogleNet)-SVM. The researchers preprocessed augmented MRI images before feeding them into a fine-tuned CNN model that integrated with SVM. Using the datasets "Braats 2021" and "Figshare," they claimed to outperform basic CNN with an accuracy of 99.8% and 98%, respectively. A similar solution with an upgraded CNN architecture has also been proposed, where they developed a 10-layered CNN with a SoftMax classifier, achieving an accuracy of 93.67%. They also proposed a hybrid CNN-KNN model, which achieved an accuracy of 99.45% and excelled in other performance metrics, outperforming all other pre-trained models .[12]

Many researchers discussed the severity of brain tumor misclassification and proposed that computational intelligence solutions would be more efficient for this problem, but none of them implemented the exact solution .[13,14].To analyze the research gap between the existing ML and DL approaches, a case study that includes various literature works was done in .[15,16] The main findings in this study include that expert radiologists do brain tumor segmentation (the segmentation is the technique of partitioning an image into several regions based on the set of pixels and intensities in the digital image. Each of the regions that have been differentiated from one to another is based on its characteristics (i.e. color, texture or intensity).[2]) and classification. ML and DL may help radiologists make better decisions. An approach that solely discusses the enhancement of MRI images using conventional imaging and contrast enhancement was proposed in .[17] They employed encoders, ResNet-50, and other CNN frameworks, achieving an accuracy of 98.5%. To improve the performance, a CNN-based DWAE-SVM was employed, which outperformed all the models to date with an accuracy of 99.7%.

Unlike the existing ML methods that primarily focus on binary classification, ensemble classifiers include the power of transfer learning. Additionally, unlike deep learning models that primarily utilize high-level pre-trained neural networks, our approach tackles multi-class image recognition using a simple yet powerful lazy learner KNN. We achieve this by using basic hyperparameter tuning with Optuna, without relying on any other DL architectures, and we achieve an accuracy of 90%. This study demonstrates that we shouldn't disregard simpler machine learning algorithms like KNN, as they can serve as a viable and efficient substitute for deep learning, particularly in situations with limited datasets.

3. Proposed Methodology

In the proposed methodology, we provide structured steps to do image classification using pure ML models. The proposed methodology provides a framework that can suit a wide range of datasets and guarantee good results in image classification. By following these steps, we can harness the true power of simpler ML algorithms and achieve great results across different image classification domains.

Algorithm 1. (H) Image Classification using Machine Learning

(1) **Resize the Image:**

$$\text{Image}_{\text{resized}} = \text{Resize}(\text{Image}, (224, 224))$$

(2) **Convert to RGB Format:**
Ensure that the image is in RGB format:

$$\text{Image}_{\text{RGB}} = \text{Convert}(\text{Image}_{\text{resized}}, \text{RGB})$$

(3) **Reshape the Image Data:**
Reshape the image data to a suitable format for processing:

$$\text{Image}_{\text{reshaped}} = \text{Reshape}(\text{Image}_{\text{RGB}}, (224, 224, 3))$$

(4) **Transform to DataFrame:**
Convert the reshaped image data into a pandas DataFrame for easier manipulation:

$$df_{\text{image}} = \text{pd DataFrame}(\text{Image}_{\text{reshaped}})$$

(5) **Identify Minority Class Samples:**
Let C_{minority} be the set of minority class samples.

(6) **Generate Synthetic Samples:**
For each sample $x_i \in C_{\text{minority}}$, generate synthetic samples $x_{\text{synthetic}}$ using interpolation:

$$x_{\text{synthetic}} = x_i + \lambda(x_j - x_i) \quad \text{for } j \in NN(x_i)$$

where λ is a random number in the range $[0, 1]$ and $NN(x_i)$ denotes the nearest neighbors of x_i.

(7) **Augment the Dataset:**
Add the synthetic samples to the original dataset to balance class distribution.

(8) **Iterate Over Components:**
For each i in the specified range $[1, k]$:

$$\text{PCA}_i = \text{PCA}(n_{\text{components}} = i)$$

- **Train and Evaluate the Model:**

$$\text{accuracy}_i = \text{Evaluate}(\text{Model}, \text{PCA}_i(df_{\text{image}}))$$

(9) **Select Optimal Components:**
Choose i^* that maximizes accuracy:

$$i^* = \arg\max_i(\text{accuracy}_i)$$

(10) **Apply PCA:**
Transform the DataFrame using the optimal number of components:

$$df_{\text{PCA}} = \text{PCA}_{i^*}(df_{\text{image}})$$

(11) **Train the Model:**

$$\text{Model}_{\text{trained}} = \text{Train}(\text{Model}, df_{\text{PCA}}, y)$$

(12) **Make Predictions:**

$$\text{predictions} = \text{Model}_{\text{trained}}(X_{\text{test}})$$

(13) **Evaluate Performance:**
Use accuracy as the primary metric:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

(14) **Define Hyperparameter Space:**
Let H be the set of hyperparameters.

(15) **Grid Search or Random Search:**
Use techniques such as Grid Search:

$$\text{best params} = \arg\max_{h \in H}(\text{Evaluate}(\text{Model}(h)))$$

(16) **Model Comparison:**
Evaluate different models and select the one with the highest accuracy:

$$\text{best model} = \arg\max_{m}(\text{accuracy}(m))$$

(17) **Preprocess the New Image:**
Follow the same preprocessing steps as above.

(18) **Make Prediction:**

$$\text{predicted class} = \text{best model}(\text{new image})$$

Algorithm 1 illustrates the steps for image classification using ML. The first step involves preprocessing. Here, we begin by setting a resolution that is dataset-specific and then convert the image data to numerical data. We need to flatten the obtained data structure to a 2-D structure and store it as a data frame, as ML algorithms only work on 2-D data. If the number of images is very low, we can skip the PCA step; otherwise, it becomes an absolute necessity. In PCA, we must first determine the optimal set of parameters by measuring the performance of the obtained data. Then we need to split the dataset into a training set, a test set, and a validation set (to tune).

Figure 2 provides a visual representation of this process. The process then continues by training the model, testing it using the test data, selecting the optimal hyper-parameters through hyper-parameter tuning, and finally making predictions using the best-fit model. We evaluate the model using various metrics, such as accuracy, precision, recall, F1-Score, and AUC-ROC score. We can use these steps to classify images using pure ML algorithms, even if the dataset changes.

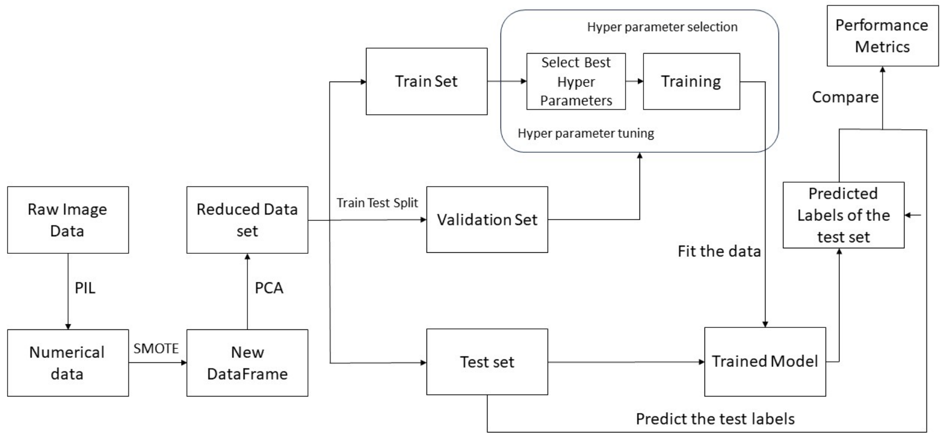


Fig. 2. Work Flow Diagram

3.1. *Data Preprocessing*

Unlike traditional scalar or one-dimensional data, image data is multidimensional. They consist of pixels arranged in a two-dimensional (2D) grid for grayscale images or a three-dimensional (3D) grid for color images, where there are separate 2-D matrices for red, green, and blue. The 2-D or 3-D structure captures spatial information; it provides information about various levels of color intensity within or across the image. Image data occupies more storage space compared to plain text or numerical data. We determine the size of an image during preprocessing by setting its resolution, such as 225 x 225, which represents the number of pixels in each dimension of the image. We can learn about the structure and content of the depicted scene from the spatial relationships between these pixels.

ML algorithms typically only recognize numerical data, necessitating the conversion of all forms of data into numerical data. The conversion of image data to numerical data involves an additional crucial step: determining the image's resolution. The image's resolution is nothing more than the number of pixels arranged in each dimension of the image. Mostly, all the images are 2D, so the resolution will be m x n. RGB values represent each pixel of an image, converting image data into numerical data. A grayscale image represents each pixel with values ranging from 0-255, where 0 represents black and 255 represents white. Data, when represented in this format, will result in a 2D matrix. If the resolution is m x n, the dimension of the matrix would be m x n. If we have k images in the dataset, we need a 3-D matrix with dimensions m x n x k. In the case of a color image, three values represent each pixel. We will use 3 m x n matrices corresponding to red, blue, and green.

The matrix corresponding to red stores the red intensity of each pixel, and so on. Thus, the data will be converted into 3D data with a dimension of 3 x m x n.

There is an additional method of representation. We represent images, whether they are gray-scale or colored, in the same way. A tuple of RGB values represents each pixel of the image. If the images are grayscale, the majority of these values would be 0 and 255, or near 0 and 255. But in colored images, the values range from 0-255. This representation gives a 3D matrix that is m x n x 3. Therefore, we require a 4D array to store all the images in a data set, each represented by a 3D array. We can use this representation when our data set contains a mixture of gray-scale and colored images. Therefore, image data upon conversion will result in 3D data structures, depending on the representation.

Traditional ML algorithms require data in 2D representation (whereas DL algorithms process higher-dimensional data). So, flattening the data into smaller dimensions is an absolute must. We flatten the data into a 2D data structure. The 2D rows represent images, and the columns represent their color values. The values in these columns change based on the image representation we chose earlier. If we choose the third implementation of the standard colored image implementation, the first column specifies the intensity of the red color of the first pixel in each image. If each pixel in the image is numbered sequentially alongside the row, then a formula can be derived to access the data of the n^{th} pixel in the image. Table 1 illustrates the formulas.

Table 1. Index of the R, G, B values of the nth pixel

Red Color of the n^{th} pixel	$(n-1) \times 3$
Green Color of the n^{th} pixel	$(n-1) \times 3 + 1$
Blue Color of the n^{th} pixel	$(n-1) \times 3 + 2$

The number of columns would be m x n x 3, which is quite a substantial number. If the resolution is 225 x 225, then the number of columns would be around 1.5 lakh. Just imagine a dataset with around 1.5 lakh features. If the image's resolution is extremely high, it just makes matters worse. Handling a dataset with 1.5 lakh (or 150,000) features will pose computational challenges and lead to prolonged processing times. Some of the potential issues that may arise are listed below:

- *Dimensionality*: High-dimensional datasets like image data suffer from the curse of dimensionality (a major curse for our model performance), where the distribution of data increases rapidly or exponentially with the increase in the number of features.
- *Memory Consumption*: Storing and manipulating datasets with many features requires significant free memory spaces or resources.
- *Computational Complexity*: Many ML algorithms have an extremely high time complexity, such as $O(n^2)$ or $O(n^3)$, where n is the number of features. As the

number of features increases, the computational complexity will rise.

- *TrainingTime*: Training ML models on this kind of data set will be time-consuming (an experiment on an MRI dataset and models like RF or XGB-Classifier took 24 minutes to complete the training; refer to Table 2 for the architecture of the XGBClassifier used), especially for algorithms that require many iterations and require a lot of computation.

Table 2. Architecture of the XGB classifier Used

Hyperparameter Name	**Value**
enable_categorical	False
objective	`'multi:softmax'`

Two obvious ways to solve the above problems are feature engineering and dimensionality reduction. With image data, feature engineering is impossible because each column of data is of the utmost importance to us. Removing a column of data can even alter the prediction of that image, which is very risky to do in fields like medicine. Instead of using raw pixel values directly, there are some feature extraction techniques like edge detection, texture analysis, scale-invariant feature transform (SIRF), speed-up robust features (SURF), and oriented fast and rotated BRIEF (ORB). We can apply these techniques to extract relevant features or pixels from the images, thereby reducing the dimensionality. This is a workable solution, but feature extraction methods may not always perform well across diverse types of images and various conditions, such as changes or variations in lighting, viewpoint, or scale. For example, edge detection works on images that have clear and distinct edges, but those techniques do not perform well when the images are noisy and cluttered.

Dimensionality reduction is a crucial step that aims to minimize the number of features while maintaining the data's meaning, a challenge often encountered in feature engineering techniques. While most feature engineering techniques rely on certain aspects of the image, such as clear edges, dimensionality reduction does not rely on these aspects. Instead, it utilizes mathematics to identify principal components or independent components, thereby reducing the number of features in the dataset. Upon testing on an MRI scan dataset, principal component analysis has reduced the number of features to 1000 from lakhs while preserving the meaning of the data. The accuracy of the testing data proves this point.

At times, dimensionality reduction techniques help us reduce lakhs of columns into thousands and even hundreds. As a result, we do not require high computations to perform data predictions. Earlier, it would take many minutes to fit data into the algorithms, but now it is done in a matter of seconds or just 1-2 minutes, depending on the algorithm's complexity.We can resolve all computational complexities and memory issues in a single step. Dimensionality reduction techniques have a trade-off between preserving the data's meaning and reducing its dimensionality. In many cases, aggressive dimensionality may result in the loss of important patterns and information, affecting the model's performance.

3.2. *Class Imbalance Handling*

Class imbalance is one cardinal issue to deal with during pre-processing. This problem arises when the count of instances of a class significantly exceeds the count of another class. Class imbalance issues affect the performance of the predictive models. Most of the algorithms work to maximize accuracy and reduce error—the model biases towards the majority class to enhance the overall accuracy. The deficit occurs when the minority class's prediction accuracy is low. These imbalanced datasets can thus lead to inflated accuracy metrics. It can be misleading in applications where accurate prediction of minority classes is important. Ignoring the minority class may result in the loss of valuable information and missed opportunities to learn from the data's underlying patterns or insights.

To ensure that the model learns from the available data successfully and generates trustworthy predictions or classifications, class imbalance handling is essential in ML. In the context of classifying MRI scans, managing class imbalance ensures that the model properly learns to discriminate between various scan types. Accurately diagnosing illnesses or anomalies using MRI scans, for example, is essential to medical diagnostics to plan treatment and provide patient care. In the selected dataset, the number of 'no tumor' case samples was nearly half that of other cases. The need to address class imbalances arose because the model should learn from both the majority and minority classes. Correctly classifying scans as 'no tumor' reduces the count of false positives.

Random over-sampling, a method for addressing this class imbalance, involves randomly replicating instances of the minority class until the distribution of classes balances with that of the dominant class. To lessen prejudice and enhance the model's ability to learn from all groups equally, this strategy aims to provide the model with more examples of the minority class. With more instances of minority classes available, the model can learn minority class patterns and characteristics more efficiently, resulting in improved model performance. This technique preserves all the information by not discarding any instance from the majority classes; please refer to Table 3. Class balancing ensures the preservation of information. This method's potential overfitting due to duplication of minority class instances, where the model might memorize the training data, is a pitfall. This can result in deficient performance on unseen data and reduced model robustness. Increasing the size of the training data necessitates more computations and redundancies in the dataset. If the randomly chosen identical instances of the original minority sample are outliers or noisy data points, it might lead to inaccurate predictions.

Synthetic Minority Over-sampling technique or otherwise SMOTE, a popular technique, synthesizes examples of the minority class to reduce potential bias from

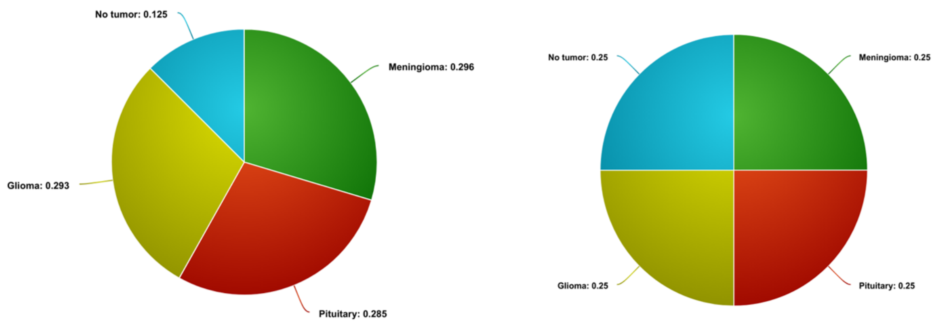


Fig. 3. Distribution of Images Before and After Class Imbalance

unbalanced data, thereby balancing the class distribution. SMOTE works by Initially assigning the no of observations to N as 1:1 as to balance the data.Later A minority class instance, Xi, is randomly selected from the minority class samples.Now KNN algorithm is used to find the K nearest neighbours on which the Synthetic data is generated.

- For each selected instance Xi, N neighbors from its k-nearest neighbors are chosen.
- For each neighbor Xj the difference vector between Xi and Xj is computed

$$\text{diff} = X_j - X_i \tag{1}$$

- A gap Value is generated randomly between 0 and 1.

$$\text{gap} = U(0, 1) \tag{2}$$

- A new Synthetic instance is created using the gap and diff

$$\text{r} = X_i + gap * diff \tag{3}$$

This method helps improve the model's ability to generalize and make accurate predictions across all classes, especially for the minority class. In the context of MRI scan classification, applying SMOTE to the 'no tumor' class helps balance the class distribution and enhances the model's ability to detect abnormalities accurately. This approach allows the model to capture the subtle features present in both normal and abnormal MRI scans effectively, thereby improving diagnostic accuracy and patient care (refer to Fig. 3).

4. Experimental Results and Discussion

The chosen models to perform this classification task are K-Nearest Neighbors, Random Forest, XGBoost, Adaboost, Gradient Boost, Stacking, Tree Models, and Naïve Bayes. The reason for choosing this set of algorithms is to test the data on as many types of ML models as possible. The list includes a probabilistic model in the form of Naïve Bayes. The list encompasses a decision tree model, a KNN-based lazy learning algorithm, three data accuracy-boosting algorithms such as XGBoost, Adaboost, and Gradient Boost, a random forest-style bagging algorithm, and stacking techniques. All types of ML algorithms test image data by making these choices.

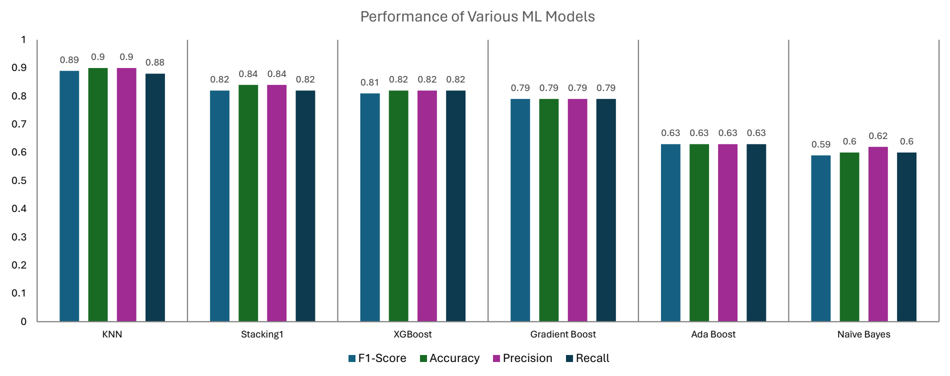


Fig. 4. Accuracy comparison

Accuracy cannot be considered the only measure between the models. Metrics like F1-score and recall can also be considered. We conduct this analysis on an MRI scan dataset, making recall a crucial parameter for comparing two models. Thus, the need to include other performance metrics apart from accuracy arises. The graphs illustrate this; please refer to Figures 4, 5, and 6.

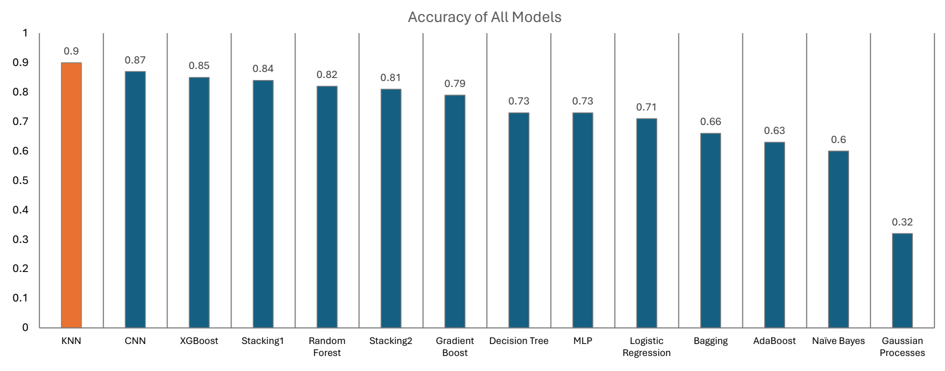


Fig. 5. Sum of F1 score vs Models

KNN outperforms other algorithms in terms of accuracy, precision, recall, and F1-score. The accuracy of KNN is even closer to that of complex DL algorithms

Table 3. Description of Various Machine Learning Algorithms

Algorithm	**Description**
KNN	An algorithm used for classification and regression tasks by finding the majority class label among the K-nearest neighbors of a given input data point.
CNN	A deep learning architecture specializing in image recognition tasks by automatically learning hierarchical patterns from input images through convolutional and pooling layers.
XGBOOST	An optimized gradient boosting algorithm known for its speed and accuracy in supervised ML tasks.
Gradient Boost	An ensemble learning technique that builds multiple weak learners sequentially, each compensating for the errors of its previous models.
Decision Tree	A tree-like model where an input is classified or predicted by traversing through the branches of the tree from the root node to a leaf node based on feature values.
Random Forest	An ensemble learning method that constructs a number of decision trees at training time and outputs the class that has the highest frequency in case of classification and mean of the prediction in case of regression.
Logistic Regression	A linear and a probabilistic model used mainly for binary classification tasks that predicts the probability of occurrence of an event. Based on the probability predicted, a class variable is assigned.
Bagging	A technique that combines multiple models trained on different subsets (bootstrapping) of the training data mainly to optimize variance and improve performance of the model.
Adaboost	A boosting algorithm that combines multiple weak classifiers to form a strong classifier.
Naive Bayes	A probabilistic model based on Bayes' theorem used for classification tasks. This model assumes that all the features are independent.
Stacking 1	A meta-ensemble method where KNN, Random Forest, and Gradient Boosting are utilized as base models, and logistic regression is the meta-model.
Stacking 2	A variant of stacking where Decision Tree and Random Forest are used as base models, and logistic regression is the meta-model.

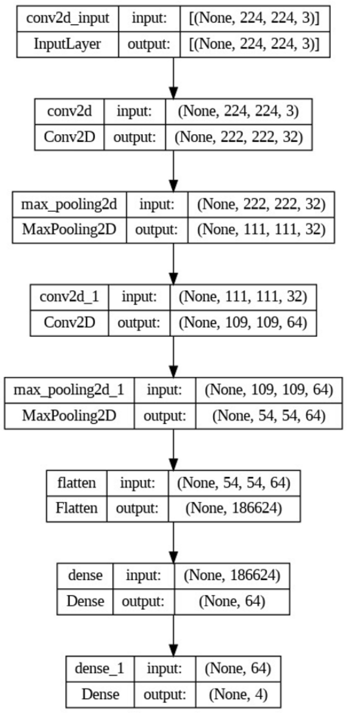
like CNN. Figure 7 depicts the architecture of both algorithms. For image recognition tasks, we can even use ML algorithms, which involve a lot less mathematical computation and a lot less computational complexity. Most of the ensemble techniques, as expected, perform well, giving good accuracy and recall values. Most ensemble techniques specifically aim to identify patterns within the data through an iterative process, which enables them to identify effective spatial relationships in an image. However, this process may result in overfitting of certain models on the data. There were some unexpected trends observed. The KNN demonstrated impressive performance, while probabilistic models demonstrated inefficiency.

Images contain complex patterns that are exceedingly difficult to capture, so probabilistic models do not capture such complex trends due to their simplistic nature. Feature independence is the assumption by most probabilistic models, but each pixel is co-related to image data, which results in inefficiency. Image recognition tasks require models to learn complex nonlinear decision boundaries, which algorithms like logistic regression and the Gaussian process do not follow. Probabilistic models such as Naive Bayes and Gaussian processes have their strengths in certain domains, but they face significant challenges in image recognition or classification due to the complex nature of image data.

KNN is a lazy learning algorithm that operates on the feature space without any explicit training process. Hence, when there is a clear boundary or clear difference between two classes of images, like two types of tumors, KNN, which calculates the similarity between two images through distances, performs effectively. While other machine learning algorithms strive excessively to learn from image data, their tendency to overfit many models leads to poor performance. However, KNN, due to its inherent laziness, does not suffer from overfitting issues. Images often contain complex patterns that are challenging to capture using simple machine learning algorithms. In this situation, KNN, which does not attempt to capture these patterns, has an advantage. However, complex machine learning algorithms, such as Boosting, are able to capture these patterns, leading to improved model performance. Boosting, being an iterative learning algorithm, aims to uncover and exploit various relationships in the data, improving its performance in each iteration and even approving its accuracy.

4.1. *Hyper Parameter Fine-tuning*

Before training the model, we set hyperparameters, which are like configurations that the model cannot learn from the data. Hyperparameter tuning is the process of optimizing the ML model by adjusting these parameters. We perform this process to improve the model's performance and accuracy, enabling it to effectively generalize to previously unseen data. While a brute-force approach is not the preferred method for tuning, we typically employ algorithms such as grid search, random search, and Bayesian optimization, depending on the specific requirements. Image recognition, a high-end topic, necessitates optimal model performance, primarily through the use of deep learning. Using machine learning (ML) for image recognition proves


conv2d_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
conv2d | input: | (None, 224, 224, 3)
Conv2D | output: | (None, 222, 222, 32)
max_pooling2d | input: | (None, 222, 222, 32)
MaxPooling2D | output: | (None, 111, 111, 32)
conv2d_1 | input: | (None, 111, 111, 32)
Conv2D | output: | (None, 109, 109, 64)
max_pooling2d_1 | input: | (None, 109, 109, 64)
MaxPooling2D | output: | (None, 54, 54, 64)
flatten | input: | (None, 54, 54, 64)
Flatten | output: | (None, 186624)
dense | input: | (None, 186624)
Dense | output: | (None, 64)
dense_1 | input: | (None, 64)
Dense | output: | (None, 4)

Parameter	Value
n_estimators	200
min_weight_fraction_leaf	0.0
min_samples_split	5
min_samples_leaf	4
max_features	auto
max_depth	30
criterion	entropy
bootstrap	False

Fig. 6. CNN VS KNN Architecture

inefficient, necessitating parameter tuning in these models to ensure efficient image recognition. Let us consider the medical field, where a model's false prediction such as the false negatives are not tolerable as they pose a greater risk to the patients .[1] Therefore, it is necessary to optimize models, with parameter tuning being the most effective approach. However, there are certain challenges associated with this process.

4.1.1. *Hyperparameter Selection:*

In this process, selecting hyperparameters for tuning is a significant challenge. ML models have anywhere from 0 to 10 hyperparameters, so figuring out which ones require tuning can be a daunting task. Furthermore, it gets tougher because some parameters may significantly affect performance while others may not. So, a deep understanding of the models and the importance of each parameter must be known to avoid a trial-and-error approach to unnecessary computation.

4.1.2. *Computational demands:*

One of the major challenges is computational resource management. As we know, hyper-parameter tuning is a complex process that requires high-end computing devices, so meticulous resource management is required to prevent computational burnout. Some algorithms, such as grid search and random search, have performed well in terms of time complexity, while others have caused the systems to overheat. These algorithms, such as grid search and random search, have taken 8–10 hours (specifically, grid search took 12–14 hours when running Ensemble Method-Stacking), which is considered exceedingly high. If the kernel dies during the tuning process, which commonly happens in compilers due to high memory usage, then the entire process, including preprocessing, needs to roll back and restart, which leads to considerable time loss.

4.1.3. *Inherent Randomness:*

There are numerous algorithms for tuning the model. Each algorithm has its benefits and challenges, such as grid search, which tries all combinations of hyperparameters that are not possible. Random search skips some set of hyperparameters, as the name suggests; it tries the parameters randomly, so we can't judge the model's performance on one iteration. The Bayesian optimization presupposes a distribution of parameters based on the most recently used hyperparameters, a distribution that may not always be accurate, leading to potential underperformance. This list doesn't end here, as methods such as gradient descent and many others have encountered similar problems due to their inherent variability. Tools like Optuna, which uses efficient search algorithms for finding the best hyperparameters that reduce time compared to other methods, are sometimes lacking because they use stochastic methods in optimization and might fall into the local optimum rather than the global optimum. We cannot guarantee that the models that work well on grid search will perform well on other algorithms, and vice versa. So, this is one of the major challenges in hyperparameter tuning.

Observations across various methods of hyperparameter tuning reveal that grid search has consumed more time compared to other methods. Random search has outperformed grid search most of the time. Bayesian optimization displayed limited efficacy across most cases. Notably, Optuna took the least time as it performed pruning, which saved duration, and it worked particularly well on the KNN model. On KNN, both Optuna and Random Search obtained the same accuracy. Tables 5 and 6 depict the optimal parameters of optuna and random search, albeit with varying parameters and values. Some methods yielded negative accuracy after performing hyperparameter tuning, which is depicted in Fig. 8, such as probabilistic models and decision trees, indicating that hyperparameter tuning is not always efficient.

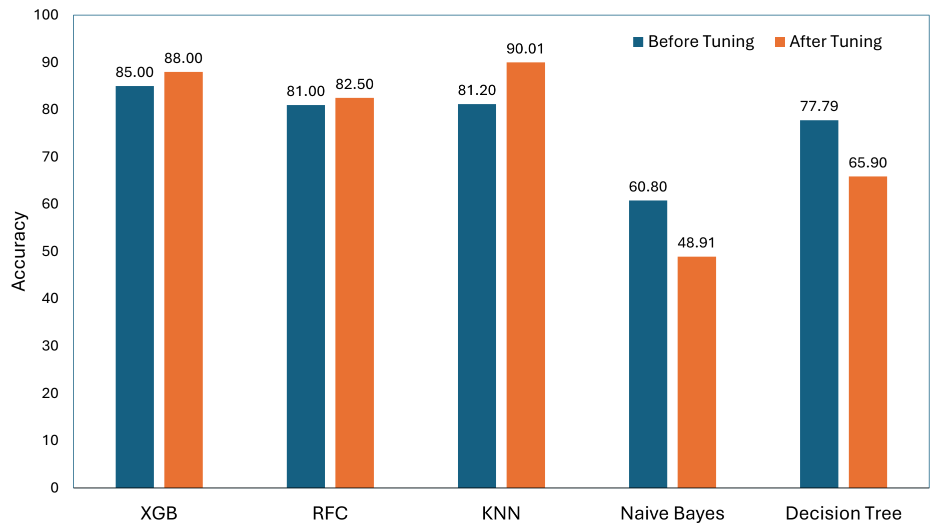


Fig. 7. Overall Performance of models before and after hyperparameter tuning

Table 4. Parameters given by Random Search

Parameter	Value
weights	distance
p	2
N_neighbors	1
N_jobs	-1
Metric params	No
Leaf_size	33
algorithm	kd_tree

Table 5. Parameters given by Optuna

Parameter	Value
weights	distance
p	2
N_neighbors	1

5. Conclusion

Our study explored using simpler ML models instead of complex DL algorithms for image classification tasks, focusing on MRI-scan brain tumor datasets. We found out that simpler ML algorithms like KNN are challenging and outperform DL algorithms like CNN in certain situations. This challenges the assumption that DL is always the best choice and superior to ML algorithms for image classification tasks. In particular, the KNN algorithm proved to be the most effective, with an accuracy close to 90%. This study extends beyond medical image analysis, enhancing performance in various fields that require image classification by exploring a broader spectrum of machine learning techniques. In summary, our research shows that simpler ML models can be effective for image classification tasks, especially for smaller datasets. By analyzing the strengths and weaknesses of different approaches, we can improve decisions about which methods to use, ultimately improving the performance of image classification systems.

Acknowledgements

We, the authors, declare that we have no conflicts of interest regarding the research presented in this paper. We have not received any financial support or other benefits from any organization that may gain or lose financially from the publication of this work. Additionally, we have no personal or professional relationships with individuals or organizations that could influence the research or its outcomes.

References

1. D. Sudharsan, S. Isha Indhu, K. S. Kumar, L. Karthikeyan, L. Srividhya, V. Sowmya, E. Gopalakrishnan, and K. P. Soman, *Analysis of machine learning and deep learning algorithms for detection of brain disorders using MRI data*, in *Artificial Intelligence on Medical Data: Proceedings of International Symposium, ISCMM 2021*, pp. 39–46, 2022. Springer.
2. K. Vikram, H. P. Menon, and D. M. Dhanalakshmy, *Segmentation of brain parts from MRI image slices using genetic algorithm*, in *Computational Vision and Bio-Inspired Computing*, pp. 457–465, 2018. Springer.
3. D. Saranyaraj, *Image de-noising and edge segmentation using bilateral filtering and gabor-cut for edge representation of a breast tumor*, in *2022 International Conference on Engineering and Emerging Technologies (ICEET)*, pp. 1–6, 2022. IEEE.
4. S. Solanki, U. P. Singh, and S. S. Chouhan, *Brain tumor classification using ML and DL approaches*, in *2023 IEEE 5th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA)*, pp. 204–208, 2023. IEEE.
5. P. Kuppusamy, V. S. Kodavaluru, and S. M. Bogar, *Brain tumor classification using optimal features and ensemble learning algorithms*, in *2023 First International Conference on Advances in Electrical, Electronics and Computational Intelligence (ICAEECI)*, pp. 1–8, 2023. IEEE.
6. R. Jayanthi, A. H. Christinal, R. Hephzibah, T. Shekinah, C. Bajaj, and D. A. Chandy, *A novel perspective on brain tumor classification using hybrid algorithm*, in *2023 International Conference on Computer, Electronics & Electrical Engineering & Their Applications (IC2E3)*, pp. 1–4, 2023. IEEE.
7. B. Anilkumar, N. P. Kumar, and K. Sowmya, *MR brain tumour classification using a deep ensemble learning technique*, in *2023 5th Biennial International Conference on Nascent Technologies in Engineering (ICNTE)*, pp. 1–6, 2023. IEEE.
8. G. Anitha, A. Usman, S. Ahmed, and H. Nasser, *Detection of brain tumor using machine learning classifiers–A comparative study*, in *2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON)*, pp. 1–6, 2023. IEEE.
9. S. K. Mishra, D. Kumar, G. Kumar, and S. Kumar, *Multi-classification of brain MRI using EfficientNet*, in *2022 International Conference for Advancement in Technology (ICONAT)*, pp. 1–6, 2022. IEEE.
10. S. Islam, R. Udhayakumar, S. Kar, N. Pavitha, U. S. Aswal, and D. K. J. B. Saini, *Enhancing brain tumor detection classification by computational intelligence*, in *2023*

Second International Conference on Electronics and Renewable Systems (ICEARS), pp. 1044–1049, 2023. IEEE.

11. H. Kibriya, M. Masood, M. Nawaz, R. Rafique, and S. Rehman, *Multiclass brain tumor classification using convolutional neural network and support vector machine*, in *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*, pp. 1–4, 2021. IEEE.
12. G. J. Ferdous, K. A. Sathi, and M. A. Hossain, *Application of hybrid classifier for multi-class classification of MRI brain tumor images*, in *2021 5th International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, pp. 1–6, 2021. IEEE.
13. A. K. Bhagat and D. Vekariya, *Computational intelligence approach to improve the classification accuracy of brain tumor detection*, in *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*, pp. 659–664, 2022. IEEE.
14. G. Hemanth, M. Janardhan, and L. Sujihelen, *Design and implementing brain tumor detection using machine learning approach*, in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1289–1294, 2019. IEEE.
15. S. Solanki, U. P. Singh, S. S. Chouhan, and S. Jain, *Brain tumor detection and classification using intelligence techniques: An overview*, IEEE Access, 2023.
16. K. Ashwini, P. Mathivanan, F. P. Sharon, and A. Kala, *Compressed classification of brain tumor images using novel chaotic map and improved SqueezeNet architecture*, Chinese Journal of Electronics, vol. 33, pp. 1–11, 2023.
17. R. Goel, N. K. Trivedi, and S. Gaur, *Brain tumor detection and segmentation analysis with machine learning and sub-variant techniques: A perspective study*, in *2023 2nd Edition of IEEE Delhi Section Flagship Conference (DELCON)*, pp. 1–4, 2023. IEEE.