Project Report
on
Automatic Optimal N-Shot learning for Brain Tumor Detection

Submitted in partial fulfilment of the course of

PG-Diploma

in

Big Data Analytics

From **C-DAC ACTS (Bangalore)**

Guided by:

Mr. Ramesh Naidu

Presented by:

Mr. Sagar Pandurang Dhandare	PRN: 210950125055
Mr. Vivek Deorao Pathre	PRN: 210950125077
Mr. Abhishek Raju Chavan	PRN: 210950125013
Mr. Shriraj Sunil Pathak	PRN: 210950125063

Candidate's Declaration

We hereby certify that the work being presented in the report titled: **Automatic Optimal N-Shot learning for Brain Tumor Detection**, in partial fulfilment of the requirements for the award of PG Diploma Certificate and submitted to the department of PG-DBDA of the C-DAC ACTS Bangalore, is an authentic record of our work carried out during the period, 1st March 2022 to 29th March 2022 under the supervision of Mr. Ramesh Naidu, C-DAC Bangalore. The matter presented in the report has not been submitted by us for the award of any degree of this or any other Institute/University.

Name and Signature of Candidate:

Mr. Sagar Pandurang Dhandare	PRN: 210950125055
Mr. Vivek Deorao Pathre	PRN: 210950125077
Mr. Abhishek Raju Chavan	PRN: 210950125013
Mr. Shriraj Sunil Pathak	PRN: 210950125063

Counter Signed by:

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Mr. Sagar Dhandare

Mr. Vivek Pathre

Mr. Abhishek Chavan

Mr. Shriraj Pathak

Have successfully completed their project on

Automatic Optimal N-Shot learning for Brain Tumor Detection

Under the guidance of

Mr. Ramesh Naidu

Mr. Ramesh Naidu	Ms. Uma Prasad
(Project Guide)	(Course Co-ordinator)

Acknowledgement

This project **“Automatic Optimal N-Shot learning for Brain Tumor Detection”** was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We all are very glad to mention the name of **Mr. Ramesh Naidu** for his valuable guidance to work on this project. His guidance and support helped us to overcome various obstacles and intricacies during the course of project work.

Our most heartfelt thanks goes to **Ms. Uma Prasad** (Course Coordinator, **PG-DBDA**) who gave all the required support and kind coordination to provide all the necessities like extra Lab hours to complete the project and throughout the course up to the last day at C-DAC ACTS, Bangalore.

From,
The whole team.

Contents

Abstract

A Brain tumor is considered as one of the aggressive diseases, among children and adults. Proper treatment, planning, and accurate diagnostics needs to be implemented to improve the life expectancy of the patients. The best technique to detect brain tumors is Magnetic Resonance Imaging (MRI). A huge amount of image data is generated through the scans. These images are examined by the radiologist. A manual examination can be error-prone due to the level of complexities involved in brain tumors and their properties. Despite deep convolutional neural networks achieved impressive progress in medical image computing and analysis, its paradigm of supervised learning demands a large number of training images and complex hyperparameter optimization processes. In clinical practices, collection of huge number of MRIs is difficult to acquire. We have developed a deep learning model with few-shot learning and Automatic Hyperparameter Optimization for brain tumor detection.

Keywords: Machine Learning, Deep Learning, Few-Shot Learning, Automatic Hyperparameter Optimization, Brain Tumors, Algorithms.

1. Introduction and Overview

The brain is a most important organ in the human body which controls the entire functionality of other organs and helps in decision making. It is primarily the control center of the central nervous system and is responsible for performing the daily voluntary and involuntary activities in the human body.

The tumor is a fibrous mesh of unwanted tissue growth inside our brain that proliferates in an unconstrained way. The right way of understanding of brain tumor and its stages is an important task to prevent and to carry out the steps in curing the illness. To do so, Magnetic Resonance Imaging (MRI) is widely used by radiologists to analyze brain tumors. The result of the analysis carried out in this project we'll try to reveals whether the brain is normal one or diseased one by applying the Deep Learning and Machine Learning techniques.

Understanding the Objective and Project Flow

For Deep Learning

1. Acquired the dataset form Kaggle having 250 Images with labels, we tried analysing this dataset using Convolutional Neural Network (CNN) vs Few-Shot Learning (FSL) and compared their prediction outcome and accuracy.

For Machine Learning

1. After which we took the same dataset of 250 images extracted the feature vectors from the images using Mahotas Zernike Moments (extracted 32 moments), and fed it to different Machine Learning Algorithms like Random Forest, Gradient Boosting, XGBoost and Support Vector Machine after which the one which was performing the best was taken further for Automatic Hyperparameter Optimization in which we have used tools like Optuna, Ray-Tune and Scikit-Optimize.

2. Since the even after Automatic Hyperparameter Optimization results were not up to the mark with (250 images dataset) so we fetched another dataset from Kaggle with 2759 images and carried out all the above-mentioned

experiments in Machine Learning on this dataset as well, thus giving us satisfactory outcome.

We have analysed and compared all the results and explained the outcomes below.

2. Dataset

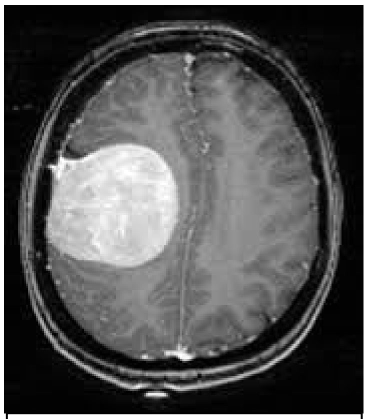
The two datasets are taken from Kaggle website. This dataset contains MRI images of brain tumor.

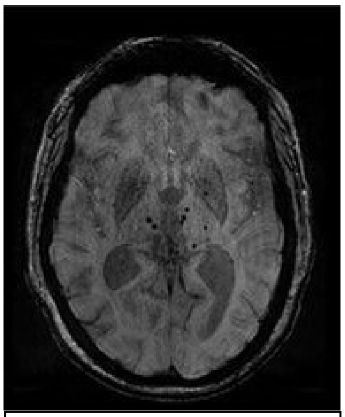
2.1Dataset 1 (with 250 Images)

There are two folders one represents the normal brain image and the other represents the tumor images. Totally there are 250 images in both these folders. Figure 1 shows the sample normal and brain tumor image. Totally 154 tumorous and 96 non-tumorous images are taken. The images are of different shapes (eg.630X630,225X225). 200 images are used for training and 50 images for testing is taken. Out of 200 training image, 123 images are tumor image and 77 images are non tumor image. Among 50 testing images, 31 tumor images and 19 non tumor images.

2.2 Dataset 2 (with 2759 Images)

There are two folders one represents the normal brain image and the other represents the tumor images. Totally there are 2759 images in both these folders. Totally 1654 tumorous and 1105 non-tumorous images are taken. The images are of different shapes (eg.630X630,225X225). 2207 images are used for training and 552 images for testing is taken. Out of 2207 training image, 1309 images are tumor image and 898 images are non tumor image. Among 552 testing images, 345 tumor images and 207 non tumor images.


With Tumor (YES)


Without Tumor (NO)

3. Materials and Methods

Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so

3.1 Algorithms used for Prediction

1. Convolutional Neural Network
2. Few-Shot Learning
3. Random Forest Classifier
4. Gradient Boosting Classifier
5. XGBoost Classifier
6. Support Vector Machine
7. XGBoost Classifier with Optuna for Hyperparameter Optimization.
8. XGBoost Classifier with Ray Tune for Hyperparameter Optimization.
9. XGBoost Classifier with Scikit Optimize for Hyperparameter Optimization.

3.2 Steps followed for applying Convolutional Neural Networks

The technique used is Convolutional Neural Network which is applied on the brain tumor dataset and their performance on classifying the image is analysed. Steps followed in applying CNN on the brain tumor dataset are as follow,

1. Create a new Environment.
2. Download and import the needed packages and libraries.
3. Set the class labels for images (1 for Brain Tumor and 0 for No Brain Tumor)
4. Convert the images into shape (128x128)
5. Normalize the Image
6. Split the images into the train and test set images.
7. Create the sequential model.
8. Compile the model.
9. Fit the train dataset.
10. Evaluate the model using the test images.
11. Plot the graph comparing the train and test accuracy.
12. Draw the confusion matrix for actual output against the predicted output

The CNN sequential model is generated by implementing different layers. The input image is reshaped into 128x128. The Convolutional layer is applied on the input image with the “Relu” as activation function, padding as same which means the output images looks like the input image and the number of filters are 32, 64, 128, 256 for different Convolutional layers. The max pooling applied with the 2x2 window size and dropout function is called with 50% of dropout. Flatten method is applied to convert the features into one dimensional array. The fully connected layer is done by calling the dense method with the number of units as 256 and “Relu” as the activation function. The output layer has 2 unit to represent the two classes and the “SoftMax” as activation function.

3.3 Steps followed for applying Few-Shot Learning

The technique used is Few-Shot Learning which is applied on the brain tumor dataset and their performance on classifying the image is analysed. Steps followed in applying FSL on the brain tumor dataset are as follows,

1. Create a new Environment.
2. Download and import the needed packages and libraries.
3. Creating a Class to load a dataset.
4. Discover the class label names.
5. Split the images and labels into Independent and Dependent features.
6. Converting images and labels into NumPy array.
7. Apply One-hot encoding on the labels.
8. Apply Stratified split on training and testing data.
9. Using Pretrained Image Embeddings (This module implements the R101x1 architecture, trained to perform multi-label classification on ImageNet-21k, a dataset with 14 million images labelled with 21,843 classes. Its outputs are the 2048-dimensional feature vectors, before the multi-label classification head. This model can be used as a feature extractor or for fine-tuning on a new target task.)
10. Create the model.
11. Compile the model.
12. Defined the Hyperparameters.
13. Fit the model on the training data.
14. Evaluate the model on testing data.
15. Cross-Validating the evaluated model (Performs n-fold cross-validated training and evaluation of a model’s hyperparameter configuration)
16. Search for the best Hyperparameter.
17. Train our final model using our best-found Hyperparameters and evaluate it on the test data.
18. Check the Accuracy and Visualize.

The objective of this study is to construct a prediction model for Brain Tumor detection with good Accuracy.

The FSL model is generated by implementing different layers (including pre-trained once). This pre-trained model is used as a feature extractor or for fine-tuning on a new target task. The input image is reshaped into 256x256.

The fully connected layer is done by calling the dense method with the number of hidden units as 64, activation function as “Relu”, Kernel Regularizer as “l2 regularization” and dropout rate as “0.3”.

The output layer has 2 unit to represent the two classes and the “SoftMax” as activation function along with Kernel Regularizer as “l2 regularization”.

3.4 Steps followed for Machine Learning

The two technologies used are Mahotas Zernike Moments and different Machine Learning algorithm along with Hyperparameters Optimization tools which is applied on the brain tumor dataset and their performance on classifying the feature vectors are analysed. Steps followed for making the Machine Learning model on the brain tumor dataset are as follows,

1. Create a new Environment.
2. Download and import the needed packages and libraries.
3. Load the Image Dataset.
4. Extract the Zernike Moments for the image dataset.
5. Exporting the vectors into a .csv file along with the target variable.
6. Load the dataset using Pandas.
7. Perform EDA.
8. Shuffle the whole dataset.
9. Convert the categorical values of Target feature into numerical values.
10. Dividing the features into Independent and Dependent.
11. Perform Train-Test Split.
12. Create an object for the preferred classifier.
13. Fit the Train data into the model.
14. Predict the Test data.
15. Get the Classification report and Confusion Matrix.

The objective of this study is to construct a prediction model for Brain Tumor detection with good comparable Metrics like Accuracy, Precision, Recall and f1 Score.

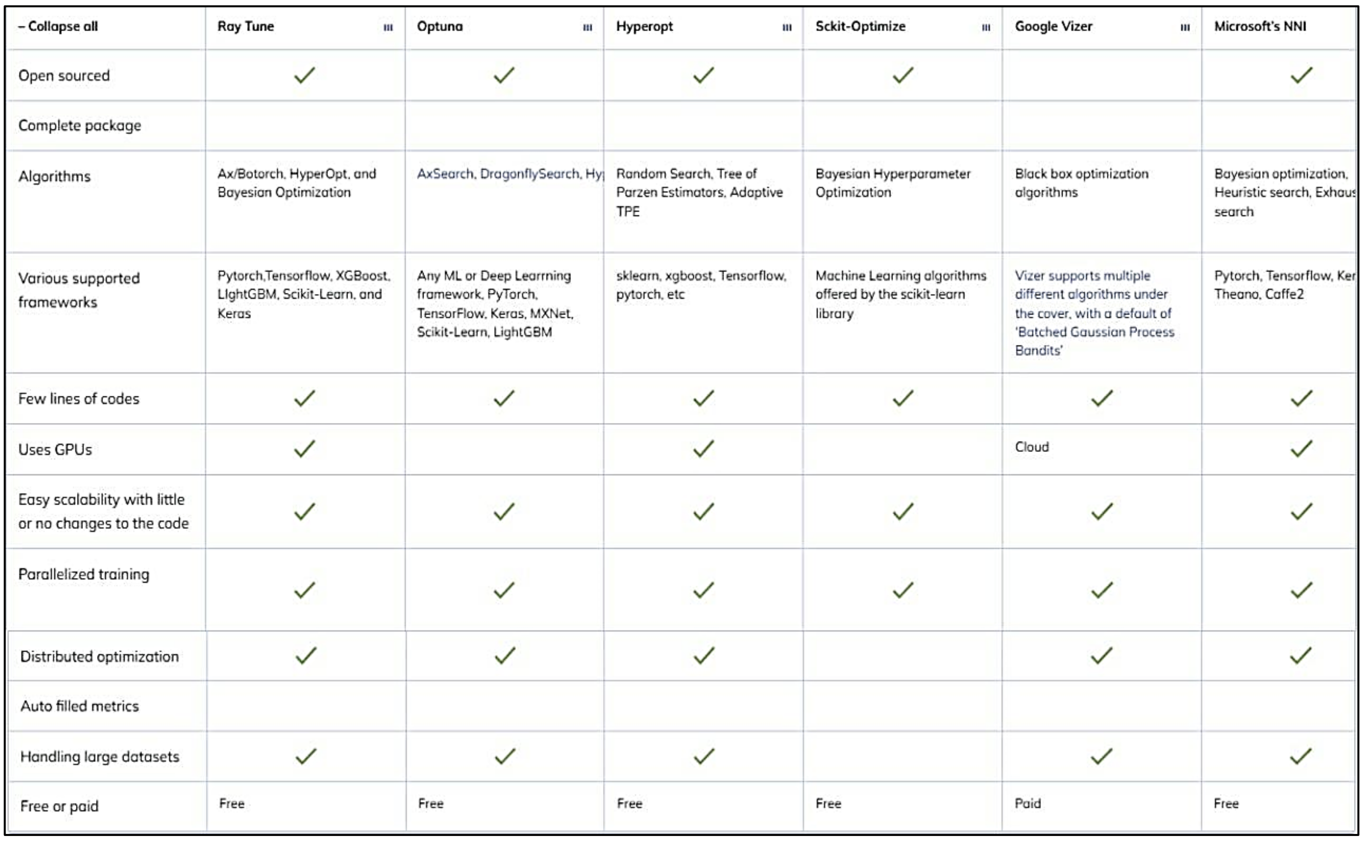
We have explored several different Machine Learning algorithms to see which produces reliable and repeatable results. The algorithms used are: Random Forest, Gradient Boosting, XGBoost and Support Vector Machine.

3.5 Steps to be followed for Hyperparameter Optimization on Machine Learning models

Hyperparameters refer to the parameters that the model cannot learn and need to be provided before training. Hyperparameter tuning basically refers to tweaking the parameters of the model, which is basically a lengthy process.

Before selecting the preferred HPO we had compared various parameters of each tool and selected the one we thought were the best.

The comparison chart is as follows,



– Collapse all	Ray Tune	Optuna	Hyperopt	Sckit-Optimize	Google Vizer	Microsoft's NNI
Open sourced	✓	✓	✓	✓		✓
Complete package						
Algorithms	Ax/Botorch, HyperOpt, and Bayesian Optimization	AxSearch, DragonflySearch, Hy	Random Search, Tree of Parzen Estimators, Adaptive TPE	Bayesian Hyperparameter Optimization	Black box optimization algorithms	Bayesian optimization, Heuristic search, Exhaus search
Various supported frameworks	Pytorch,Tensorflow, XGBoost, LIghtGBM, Scikit-Learn, and Keras	Any ML or Deep Learrning framework, PyTorch, TensorFlow, Keras, MXNet, Scikit-Learn, LightGBM	sklearn, xgboost, Tensorflow, pytorch, etc	Machine Learning algorithms offered by the scikit-learn library	Vizer supports multiple different algorithms under the cover, with a default of 'Batched Gaussian Process Bandits'	Pytorch, Tensorflow, Ker Theano, Caffe2
Few lines of codes	✓	✓	✓	✓	✓	✓
Uses GPUs	✓		✓		Cloud	✓
Easy scalability with little or no changes to the code	✓	✓	✓	✓	✓	✓
Parallelized training	✓	✓	✓	✓	✓	✓
Distributed optimization	✓	✓	✓		✓	✓
Auto filled metrics						
Handling large datasets	✓	✓	✓		✓	✓
Free or paid	Free	Free	Free	Free	Paid	Free

The model which produces the best outcome from the above-mentioned algorithms, will be taken further on which Hyperparameter Optimization will be conducted using Optuna, Ray-Tune, Scikit-Optimize. Steps followed for making the Hyperparameter Optimization on the brain tumor dataset are as follows,

1. Use the existing environment.

2. Download and import the needed packages and libraries.
3. Create an Objective function.
4. Perform all the above-mentioned steps till train-test split.
5. List the range of different hyperparameters.
6. Create an Object of the objective function and pass the values like number of trials, etc.
7. Using Best parameters found from Hyperparameter Optimization tool and run the model.

Hyperparameter optimization is simply a search to get the best set of hyperparameters that gives the best version of a model on a particular dataset, here we have use three such tools which are as follows,

Optuna

Optuna is designed especially for machine learning. It's a black-box optimizer, so it needs an objective function. This objective function decides where to sample in upcoming trials, and returns numerical values (the performance of the hyperparameters). It uses different algorithms, such as Grid Search, Random Search, Bayesian and Evolutionary algorithms to find the optimal hyperparameter values.

Ray Tune

Ray provides a simple, universal API for building distributed applications. Tune is a Python library for experiment execution and hyperparameter tuning at any scale. Tune is one of the many packages of Ray. Ray Tune is a Python library that speeds up hyperparameter tuning by leveraging cutting-edge optimization algorithms at scale.

Scikit-Optimize

Scikit-Optimize is an open-source library for hyperparameter optimization in Python. It was developed by the team behind Scikit-learn. It's relatively easy to use compared to other hyperparameter optimization libraries. It has sequential model-based optimization libraries known as Bayesian Hyperparameter Optimization (BHO). The advantage of BHO is that they find better model settings than random search in fewer iterations.

4. Model Building

4.1 Model Building with Image as a Dataset

From the above-mentioned datasets of 250 Images,

4.1.1 Convolutional Neural Network (CNN)

Firstly, we tried analysing this dataset using **Convolutional Neural Network**,

A convolutional neural network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images. Convolutional neural networks are widely used in computer vision and have become the state of the art for many visual applications such as image classification, and have also found success in natural language processing for text classification.

Convolutional neural networks are very good at picking up on patterns in the input image, such as lines, gradients, circles, or even eyes and faces. It is this property that makes convolutional neural networks so powerful for computer vision. Unlike earlier computer vision algorithms, convolutional neural networks can operate directly on a raw image and do not need any pre-processing.

Creating, Compiling and Evaluating the CNN model and getting the accuracy. Here in CNN even after performing manual hyperparameter tunning and increasing the number of epochs we were able to get a maximum of 68% accuracy.

The. ipynb file from the GitHub repository of the CNN model is attached below for reference.

Jupyter Notebook Link: CNN_250.ipynb

4.1.2 Few-Shot Learning

Few-shot learning can also be called n-Shot learning or Low-shot learning is a topic of machine learning subjects where we learn to train the dataset with lower or limited information.

Traditional machine learning models need to feed data as much as the model can take and because of large data feeding, we enable the model to predict better. Where this learning aims to build accurate machine learning models with less training data. There can be many benefits of it like it can help in reducing the time costs, computational costs, data analysis costs, etc.

Few-shot learning methods basically work on the approach where we need to feed a light amount of data to model for training.

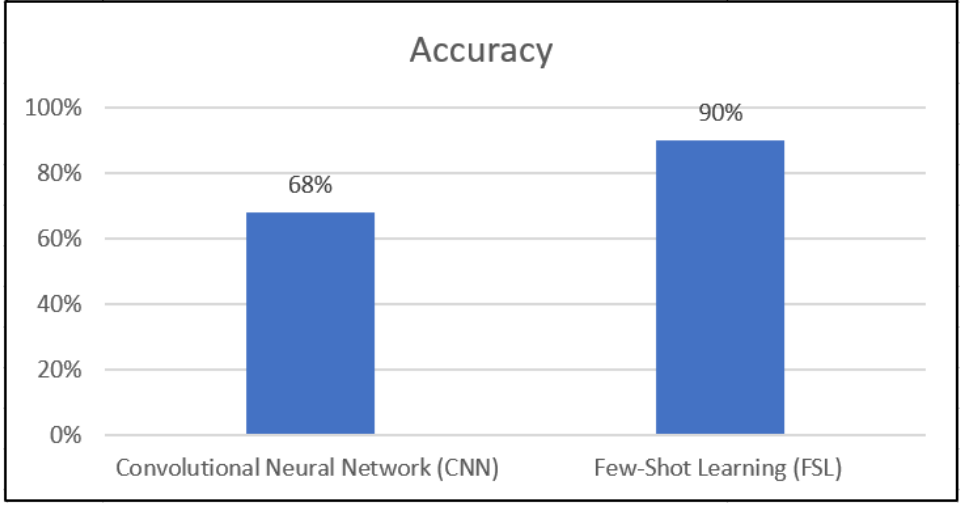
Since the accuracy of CNN model was not satisfactory and the size of the dataset was also small, we tried another algorithm i.e., Few-Shot Learning. After building the FSL model we got the Accuracy of 90%. Which was a satisfactory jump in the performance, but still since the data is of the Healthcare domain, we need to get the accuracy above 90%.

The. ipynb file from the GitHub repository of the FSL model is attached below for reference.

Jupyter Notebook Link: FSL_250.ipynb

As we can clearly conclude from the analysis done above that FSL is performing far better than the CNN, the performance of which can be compared as follows,

Report on the Results Deep Learning Models (with 250 Images)		
Sr.No.	**FrameWork / Libraries**	**Accuracy**
1	Convolutional Neural Network (CNN)	68%
2	Few-Shot Learning (FSL)	90%



4.2 Model Building with .csv

4.2.1 Analysing performance of various ML Algorithms with 250 Image dataset

From the above-mentioned datasets of 250 Images,

First, we've extracted the Zernike moments using Mahotas Python library with 32 degrees and have exported into a .csv file with target label as well.

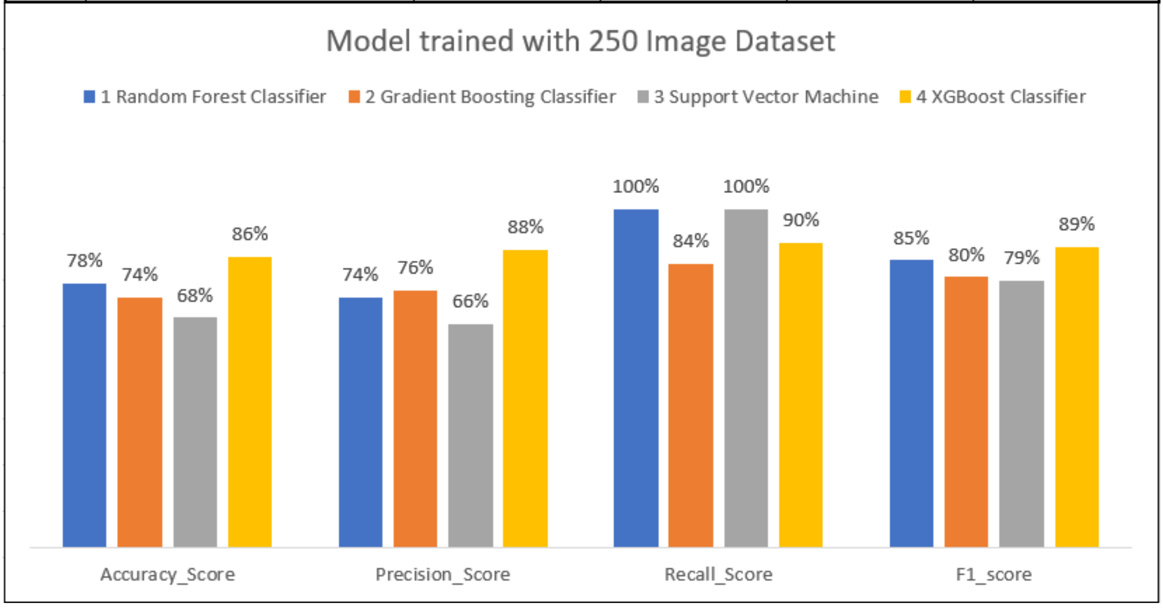
After which we have implemented various Machine Learning Algorithms as mentioned below,

- Random Forest Classifier.
- Gradient Boosting Classifier.
- eXtreme Gradient Boosting Classifier (XGBoost).
- Support Vector Machine.

After building the model and fitting the data from the generated .csv file with default hyperparameter we got the performance metrics as follows,

Results on Different Machine Learning Algorithms (with 250 Image dataset)

Sr.No.	FrameWork / Libraries	Accuracy_Score	Precision_Score	Recall_Score	F1_score
1	**Random Forest Classifier**	78%	74%	100%	85%
2	**Gradient Boosting Classifier**	74%	76%	84%	80%
3	**Support Vector Machine**	68%	66%	100%	79%
4	**XGBoost Classifier**	86%	88%	90%	89%



As can be Clearly visualize from the graph above XGBoost is performing quite well when compared with the other ML Algorithms.

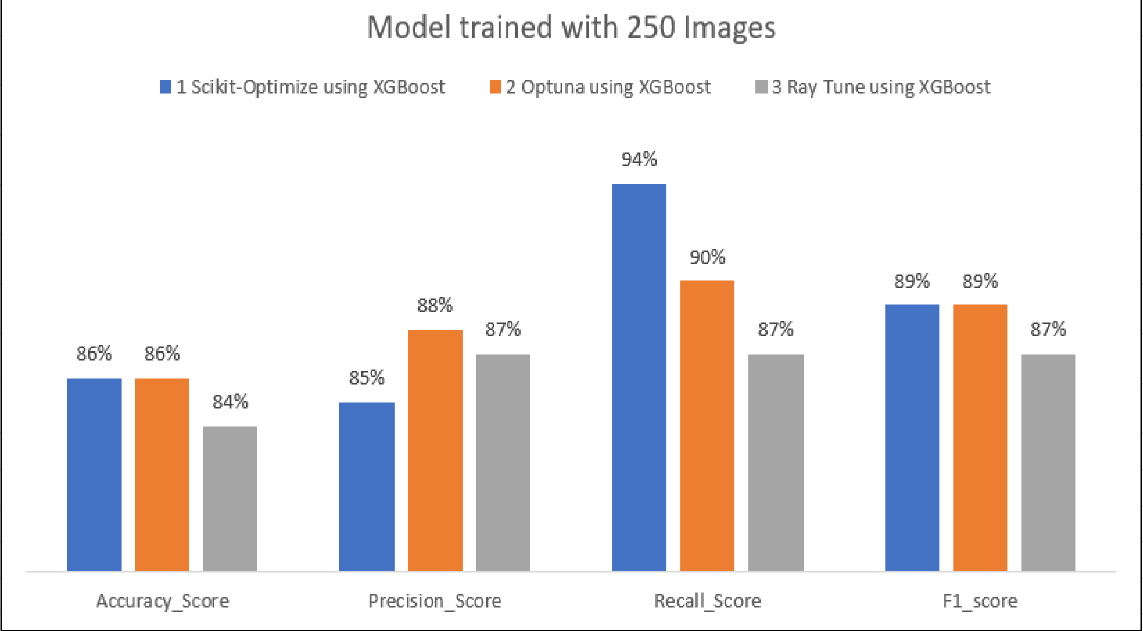
The. ipynb file from the GitHub repository of the ML models is attached below for reference.

Jupyter Notebook Link: ML_All_Algo_250.ipynb

Since the performance metrics of none of the above-mentioned Algorithms were up to the standards of the Healthcare domain, we performed Automatic Hyperparameter Optimization on XGBoost Classifier with HPO tools like Optuna, RayTune and Scikit-Optimize, and the performances of all the tools can be analysed form the table and graph given below,

Report on the Results of Different HPO tools (with 250 Image Dataset)

Sr.No.	FrameWork / Libraries	Accuracy_Score	Precision_Score	Recall_Score	F1_score
1	**Scikit-Optimize using XGBoost**	86%	85%	94%	89%
2	**Optuna using XGBoost**	86%	88%	90%	89%
3	**Ray Tune using XGBoost**	84%	87%	87%	87%



As can be visualize from the graph above almost all the HPO tools are performing quite similar and we cannot really find any distinct difference to get the optimal solution.

Even the Performance metrics are not up to the mark for the Healthcare domain.

The. ipynb files from the GitHub repository of the ML models with HPO are attached below for reference.

Jupyter Notebook Link:

1. Optuna - HPO_Optuna_250.ipynb
2. Ray-Tune - HPO_Raytune_250.ipynb
3. Scikit-Optimize - HPO_ScikitOptimize_250.ipynb

Since even after trying the HPO tools on the 250 Image dataset we are not achieving the expected performance from the given dataset.

Therefore, we have taken a new Image dataset of 2759 Image also downloaded from Kaggle (details of which have already been discussed above)

4.2.2 Analysing performance of various ML Algorithms with 2759 Image dataset

From the above-mentioned datasets of 2759 Images,

First, we've extracted the Zernike moments using Mahotas Python library with 32 degrees and have exported into a .csv file with target label as well.

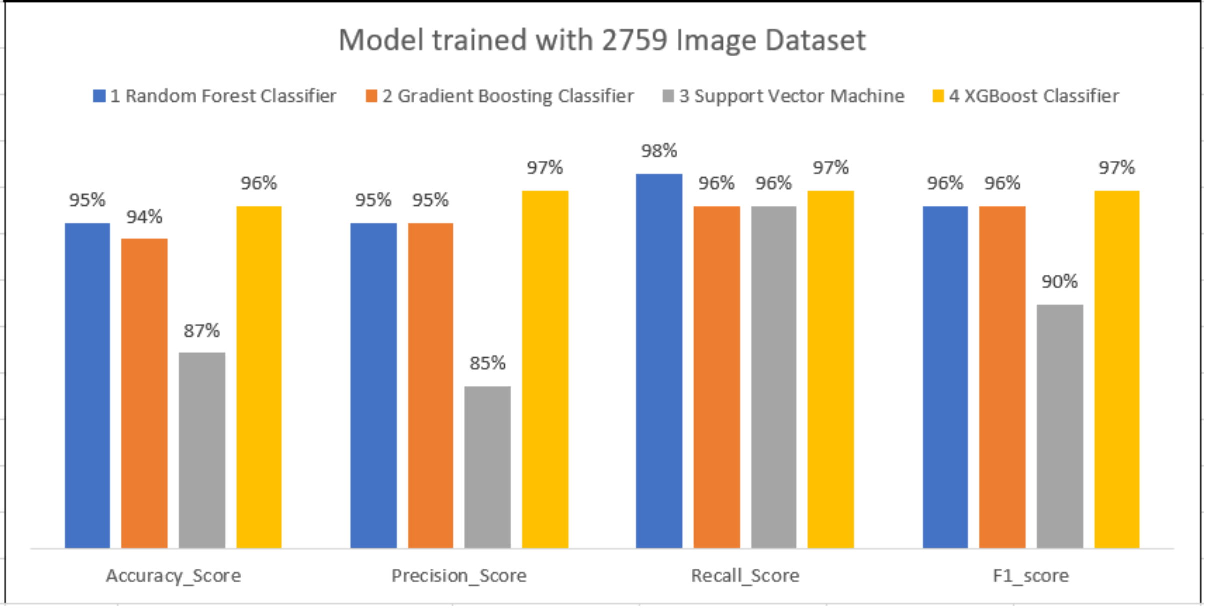
After which we have implemented various Machine Learning Algorithms as mentioned below,

1 Random Forest Classifier.
2 Gradient Boosting Classifier.
3 eXtreme Gradient Boosting Classifier (XGBoost).
4 Support Vector Machine.

After building the model and fitting the data from the generated .csv file with default hyperparameter we got the performance metrics as follows,

Results on Different Machine Learning Algorithms (with 2759 Image dataset)

Sr.No.	FrameWork / Libraries	Accuracy_Score	Precision_Score	Recall_Score	F1_score
1	**Random Forest Classifier**	95%	95%	98%	96%
2	**Gradient Boosting Classifier**	94%	95%	96%	96%
3	**Support Vector Machine**	87%	85%	96%	90%
4	**XGBoost Classifier**	96%	97%	97%	97%



As we can visualize from the graph above XGBoost is performing just well when compared with the other ML Algorithms, but the overall performances of all the models on the test data is very good.

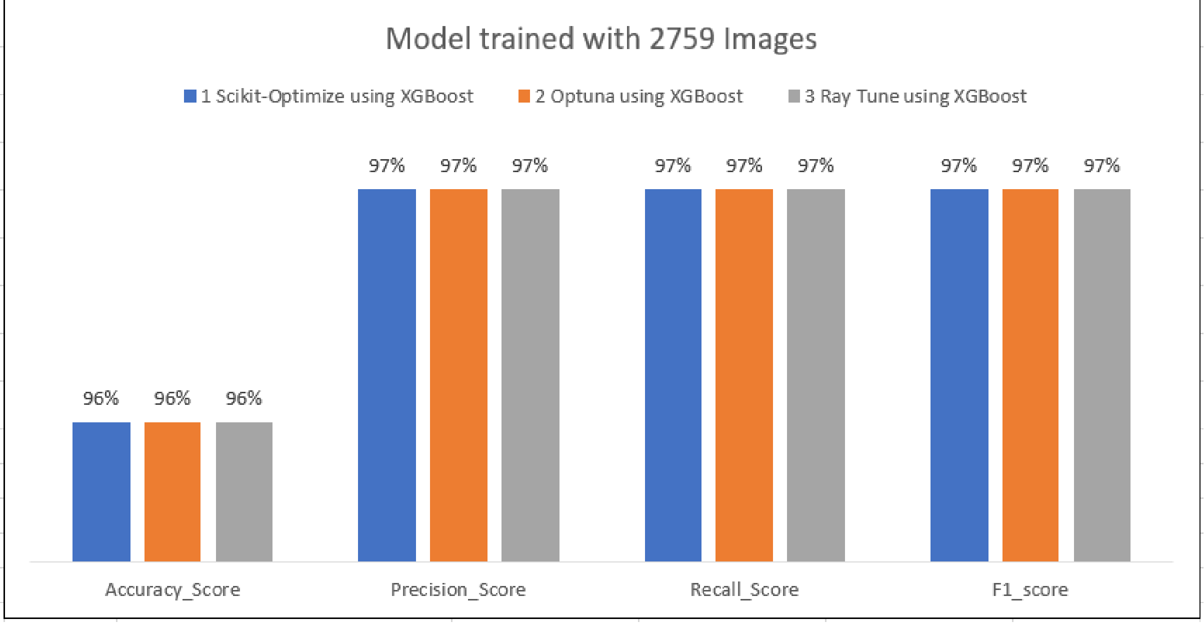
The. ipynb file from the GitHub repository of the ML models is attached below for reference.

Jupyter Notebook Link: ML_All_Algo_3000.ipynb

As the models for Automatic Hyperparameter Optimization was already prepared for 250 image dataset, therefore we decided to conduct HPO on comparatively well performing model i.e., XGBoost Classifier and the performances of all the tools can be analysed form the table and graph given below,

Report on the Results of Different HPO tools (with 2759 Image Dataset)

Sr.No.	FrameWork / Libraries	Accuracy_Score	Precision_Score	Recall_Score	F1_score
1	**Scikit-Optimize using XGBoost**	96%	97%	97%	97%
2	**Optuna using XGBoost**	96%	97%	97%	97%
3	**Ray Tune using XGBoost**	96%	97%	97%	97%



As can be visualize from the graph above almost all the HPO tools are performing equally well on the dataset and thus we cannot really find any distinct difference to get the optimal solution.

But the Performance metrics is very good for the Healthcare domain.

The. ipynb files from the GitHub repository of the ML models with HPO are attached below for reference.

Jupyter Notebook Link:

1. Optuna - HPO_Optuna_3000.ipynb
2. Ray-Tune - HPO_Raytune_3000.ipynb
3. Scikit-Optimize - HPO_ScikitOptimize_3000.ipynb

From the above analysis on the 2759 image dataset we can conclude that there is not much of a difference in the performance metrics for this particular dataset, as the metrics were already very good for all the ML Algorithms HPO was not really needed but in any case when HPO was conducted no difference was found on the performance and all the HPO tools were also giving out the similar outcomes.

5. Conclusion

The dataset and the problem statement that we have taken is of the Healthcare domain which required high accuracy, precision and overall good performing metrics.

For which we have firstly considered CNN which is considered as one of the best techniques in analysing the image dataset. CNN model which we build generated here has produces 68% of testing accuracy with manual Hyperparameter tunning and this can be increased by providing more image data.

Since the performance was not satisfactory, we tried Few-Shot Learning and got 90% of testing accuracy, but in the Healthcare domain the results should not be uncertain which can affect an important decision.

So we considered trying out Machine Learning Algorithms by extracting the Zernike moments using Mahotas Python library from the image dataset with degree(feature vectors) as 32. And then applied various ML algorithm. But we did not receive satisfactory results (around 85% test accuracy).

Therefore, we applied different Automatic Hyperparameter Optimization Tools but even then performance was not as expected.

So, we added more Images to our dataset and then once again went through the same steps as we earlier did like extracting Zernike moments and apply machine learnings algorithm. Thus, after adding more images, we accrued excellent performance metrics, as the metrics were already very good for all the ML Algorithms HPO was not really needed but in any case, when HPO was conducted no difference was found on the performance and all the HPO tools were also giving out the similar outcomes.

So, we can thus conclude that among all the Algorithms used so far Few-Shot Learning Algorithm is the among the best when we are provided with very limited data and resources thus saving on time cost, computational cost and Data Analysis Cost.

6. Application

Few-shot learning has a wide range of applications in the trending fields of data science such as computer vision, robotics, and much more. They can be used for character recognition, image recognition, and classification approaches. They perform well for some applications of NLP such as translation, text classification, sentiment analysis, etc. In robotics, they can be used to train robots with a small number of training sets.

7. System and Software Requirements

Hardware Specifications

Machine: Desktop/Laptop

Operating system: Windows 10 (or above) or Linux 18 LTS (or above)

Processors: Intel core i5 or AMD 5 series (minimum)

Memory: 8 GB RAM or above

Hard Disk (SSD): 250 GB or more

Video Card (optional): Intel Integrated Graphics (suggested – 4 GB graphics card - NVIDIA)

Network: Ethernet / Wi-Fi with 25 Mbps Speed Connection (UL/DL)

Software Specifications

Language: Python 3.7 or above (stable build)

Platform: Anaconda Latest stable build

Notebooks: Jupyter and Google Colab

Libraries: Pandas, Numpy, Matplotlib, Seaborn, Keras, Tensorflow, Scikit Learn, Mahotas.

Optimizations tools: Optuna, Ray-Tune, Scikit-Optimize.

8. References

[1] Few-Shot Learning

https://www.kaggle.com/code/epeterson17/few-shot-learning-of-rice-leaf-diseases

https://www.analyticsvidhya.com/blog/2021/05/an-introduction-to-few-shot-learning/

https://analyticsindiamag.com/an-introductory-guide-to-few-shot-learning-for-beginners/

[2] Convolutional Neural Network

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[3] Datasets

https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection

https://www.kaggle.com/datasets/abhranta/brain-tumor-detection-mri

[4] Machine Learning Algorithms

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://builtin.com/data-science/random-forest-algorithm

https://xgboost.readthedocs.io/en/stable/

https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

https://scikit-learn.org/stable/modules/svm.html

https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[5] Hyperparameter Optimization Tools and Frameworks

https://neptune.ai/blog/best-tools-for-model-tuning-and-hyperparameter-optimization

https://towardsdatascience.com/10-hyperparameter-optimization-frameworks-8bc87bc8b7e3

https://optuna.org/

https://docs.ray.io/en/latest/tune/index.html

https://scikit-optimize.github.io/stable/

[6] Zernike Moments using Mahotas Python library

https://www.geeksforgeeks.org/mahotas-zernike-moments/

http://mahotas.readthedocs.io/en/latest/