Dissertation On

Detection of Brain Tumor of MRI based images using Machine Learning- An Approach

Thesis submitted towards partial fulfillment of the requirement for the degree of

Master in Multimedia Development

Submitted by

MAMATA KHAMARU

EXAMINATION ROLL NO.: M4MMD22008

UNIVERSITY REGISTRATION NO.: 154516 of 2020-21

Under the guidance of

Mr. JOYDEEP MUKHERJEE

School of Education Technology

Jadavpur University

Course affiliated to

Faculty of Engineering and Technology

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled **“Detection of Brain Tumor of MRI based images using Machine Learning- An Approach”** is a bonafide work carried out by **Mamata Khamaru** under supervision and guidance of **Mr**. **Joydeep Mukherjee** for partial fulfillment of the requirements for the degree of **Master in Multimedia Development** in **School of Education Technology**, during the academic session 2020-2022.

_ _ _ _ _ _ _ _ _ _ _ _

Mr. Joydeep Mukherjee

Supervisor

Jadavpur University

Kolkata:700032

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Dr. Matangini Chattopadhyay

Director

School of Education Technology

Jadavpur University

Kolkata:700032

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Dean-FISLM

Jadavpur University

Kolkata:700032

CERTIFICATE OF APPROVAL

This foregoing thesis is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of final examination

for evaluation of thesis

DECLARATION OF ORIGINALITY AND COMPLLIANCE OF ACADEMIC ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of her **Master in Multimedia Development** studies during academic session 2020- 2022.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referenced all materials and results that are not original to this work.

Name: Mamata Khamaru

Examination Roll Number: M4MMD22008

Thesis Title: **Detection of Brain Tumor of MRI based images using Machine Learning- An Approach**

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _   _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Signature   **Date**

ACKNOWLEDGEMENT

I feel fortunate while presenting this dissertation at **School of Education Technology, Jadavpur University, Kolkata,** in the partial fulfillment of the requirement for the degree of **Master in Multimedia Development.**

I hereby take this opportunity to show my gratitude towards my mentor, **Mr. Joydeep Mukherjee,** who has guided and helped me with all possible suggestions, support, aspiring advice and constructive criticism along with illuminating views on different issues of this dissertation which helped me throughout my work.

I would like to express my warm thanks to **Dr. Matangini Chattopadhyay, Director of School of Education Technology** for his timely encouragement, support and advice. I would also like to thank **Dr. Ranjan Parekh** and **Dr. Saswati Mukherjee** for their constant support during my entire course of work. My thanks and appreciation goes to my classmates from M.Tech in Information Technology(Courseware Engineering) and Master in Multimedia Development. I do wish to thank all the departmental support staffs and everyone else who has different contributions to this dissertation.

Finally, my special gratitude to my parents who have invariably sacrificed and supported me and made me achieve this height.

Date:

Place: Jadavpur, Kolkata

Mamata Khamaru

Master in Multimedia Development

School of Education Technology

Jadavpur University

Kolkata:700032

Contents

Topic	Page no.

Executive Summary

The dissertation work is implemented with Logistic Regression Algorithm , Support Vector Machine Algorithm , Decision Tree Algorithm, KNN Algorithm , Random Forest Algorithm , Linear Discriminant Analysis Algorithm , Gradient Booster Algorithm . All algorithms are supervised machine learning algorithm.

The proposed approach takes into account data from different patients. The main goal is to come up with Logistic Regression Algorithm, Support Vector Machine Algorithm and KNN Algorithm . After implementing the mother dataset I have implemented four more algorithms i.e. Random Forest Algorithm, Decision Tree Algorithm, Linear Discriminant Analysis Algorithm and Gradient Booster Algorithm.

1. Introduction

1.1. Brain Tumor

The brain is a vital organ in the human body and responsible for control and decision making. As the managing center of nervous systems, this part is very essential to be protected from any harm and illness. Tumors are the predominant infections caused by abnormal growth of cells that damages the Brain. Meningioma, Glioma, and Pituitary are brain tumors as opposed to the other types. Meningiomas are mostly a non-cancerous class of tumors that often develop in the narrow walls that usually surround the brain . Brain tumors are one of the life-threatening diseases that can directly affect human lives. The correct understanding of brain tumor stages is an important task for the prevention and cure of illness. To do so, Magnetic Resonance Imaging (MRI) is widely used by radiologists to analyze brain tumors. The result of this analysis reveals whether the brain is normal or abnormal. On the other hand, it identifies the type of tumor in the case of abnormality . With the advent of machine learning, the processing of MR images to have a for fast and accurate detection of brain tumors matter.

1.2. Machine Learning

In the beginning, approaches consisted of three steps: pre-processing of MR images, , feature generation, and extraction and classification. The median filter was used for the improvement of image quality and preserving the edges in the pre-processing stage . Segmentation of images with clustering methods such as k-means, fuzzy C-means, etc. generates beneficial features from images. Image segmentation plays an essential role in analyzing and interpreting images. It has vast applications in brain imaging, such as tissue classification, tumor location, tumor volume estimation, blood cell delineation, surgical planning, matching. In a brain tumor segmentation is applied by using a MR images. Automatic detection of the anatomical structure of the brain with a deep neural network. In a voting technique for an ensemble of visual appearances such as intensity and adaptive shape modes using a combination of discrete Gaussian and higher-order patterns such as Markov-Gibbs, random field classification is used. In this work, after denoising with a non-local mean filter, a Bayesian fuzzy clustering approach is utilized for the segmentation of brain tumors. In the 2D MRI mages are partitioned into the left and right hemisphere and statistical features such as mean, homogeneity, absolute value, and inertia are computed for the Support Vector Machine (SVM) classifier. Due to the vast number of features in step two, most studies consist of a further step to extract features with more valuable information using algorithms like principal component analysis (PCA) or SIFT detectors and SURF descriptors. After a hybrid feature extraction with a covariance matrix, a regularized extreme learning is used to classify brain abnormality . Classification method such as K-Nearest Neighbors, Decision Trees, Support Vector Machine (SVM) and the Random Forest is the popular machine learning technique on image analysis .

1.3. Problem Statement

Detection of Brain Tumor of MRI based images using Machine Learning- An Approach.

Brain tumors are a heterogeneous group of central nervous system neoplasms that arise within or adjacent to the brain. Moreover, the location of the tumor within the brain has a profound effect on the patient's symptoms, surgical therapeutic options, and the likelihood of obtaining a definitive diagnosis. The location of the tumor in the brain also markedly alters the risk of neurological toxicities that alter the patient's quality of life.

There are many techniques for brain tumor detection. I have used Supervised Machine Learning Algorithms i.e. SVM , Logistic Regression , KNN, LDA, Gradient Boosting Algorithm, Random Forest , Decision Tree to identify the efficiency among the peers.

1.4. Objective

Detection of Brain Tumor of MRI based images using various supervised machine learning algorithms to get best algorithm to solve the problem.

2.Background Concept

Some key concepts will be discussed to understand the details of the algorithm that have been implemented for the detection of Brain Tumor will be discussed.

2.1. Logistic Regression

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

Logistic regression can also play a role in data preparation activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load (ETL) process in order to stage the information for analysis.

Logistic regression applications in business:

Organizations use insights from logistic regression outputs to enhance their business strategy for achieving business goals such as reducing expenses or losses and increasing ROI in marketing campaigns.

Why is logistic regression important?

Logistic regression is important because it transforms complex calculations around probability into a straightforward arithmetic problem. Admittedly, the calculation itself is a bit complex, but modern statistical applications automate much of this grunt work. This dramatically simplifies analysing the impact of multiple variables and helps to minimize the effect of confounding factors.

As a result, statisticians can quickly model and explore the contribution of various factors to a given outcome.

Logistic regression works very similar to linear regression, but with a binomial response variable. The greatest advantage when compared to Mantel-Haenszel OR is the fact that you can use continuous explanatory variables and it is easier to handle more than two explanatory variables simultaneously. Although apparently trivial, this last characteristic is essential when we are interested in the impact of various explanatory variables on the response variable. If we look at multiple explanatory variables independently, we ignore the covariance among variables and are subjected to confounding effects, as was demonstrated in the example above when the effect of treatment on death probability was partially hidden by the effect of age.

Logistic regression works very similar to linear regression, but with a binomial response variable. The greatest advantage when compared to Mantel-Haenszel OR is the fact that you can use continuous explanatory variables and it is easier to handle more than two explanatory variables simultaneously. Although apparently trivial, this last characteristic is essential when we are interested in the impact of various explanatory variables on the response variable. If we look at multiple explanatory variables independently, we ignore the covariance among variables and are subjected to confounding effects, as was demonstrated in the example above when the effect of treatment on death probability was partially hidden by the effect of age.

Logistic regression works very similar to linear regression, but with a binomial response variable. The greatest advantage when compared to Mantel-Haenszel OR is the fact that you can use continuous explanatory variables and it is easier to handle more than two explanatory variables simultaneously. Although apparently trivial, this last characteristic is essential when we are interested in the impact of various explanatory variables on the response variable. If we look at multiple explanatory variables independently, we ignore the covariance among variables and are subjected to confounding effects, as was demonstrated in the example above when the effect of treatment on death probability was partially hidden by the effect of age.

A logistic regression will model the chance of an outcome based on individual characteristics. Because chance is a ratio, what will be actually modeled is the logarithm of the chance given by:

$$\log(\frac{\pi}{1-\pi})=\beta0+\beta1x1+\beta2x2+\ldots\beta mxm$$

where π indicates the probability of an event , and β_i are the regression coefficients associated with the reference group and the x_i explanatory variables. At this point, an important concept must to be highlighted. The reference group, represented by β_0, is constituted by those individuals presenting the reference level of each and every variable $x_{1\ldots m}$.

2.2. Support Vector Machine (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

Types of SVM

SVM can be of two types:

Linear SVM: Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

Non-linear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

.The data may not be well linearly separable or not linearly separable at all, which is often the case for real world data. In the example given above, it can happen for example that two customers are acting completely similar in an online shop, with only one of them making a purchase. This would result in not separable data, due to the same attribute vector having different labels.

.The second concern is the possibility of overfitting the SVM. To avoid this, data has to be preprocessed to identify noise and accept some misclassifications. Other- wise, the accuracy values of the SVM will be flawed and result in more erroneous classification for future events.

The first problem can be resolved by using the kernel trick, mapping the *n*-dimensional input data to a higher dimensional space, where the data can be separated linearly.

2.3. Decision Tree

A Decision Tree is a classification technique that focuses on an easily understandable representation form and is one of the most common learning methods. Decision Trees use data sets that consist of attribute vectors, which in turn contain a set of classification attributes describing the vector and a class attribute assigning the data entry to a certain class. A Decision Tree is built by iteratively splitting the data set on the attribute that separates the data as well as possible into the different existing classes until a certain stop criterion is reached. The representation form enables users to get a quick overview of the data, since Decision Trees can easily be visualized in a tree structured format, which is easy to understand for humans.

In Decision Trees, nodes can be separated into the root node, inner nodes, and end nodes, also called leaf's. The root node represents the start of the decision support process and has no incoming edges. The inner nodes have exactly one incoming edge and have at least two outgoing edges. For instance, such a test might ask: "Is the customer older than 35 for the attribute age?". Leaf nodes consist of an answer to the decision problem, which is mostly represented by a class prediction. As an example, a decision problem might be the question whether a customer in an online shop will make a purchase or not, with the class predictions being yes and no. Leaf nodes have no outgoing and exactly one incoming edge. Edges represent the decision taken from the previous node. Given a node *n*, all following nodes that are separated by exactly one edge to *n* are calledchildren of *n*, while *n* is called parent of all its child nodes. Figure 2.2 shows an exampleof a Decision Tree. For instance, a data record, having the attributes cold, polarBear would be passed down to the left subtree, since his temperature attribute is cold and then down to the leaf “North Pole” being classified with the corresponding label.

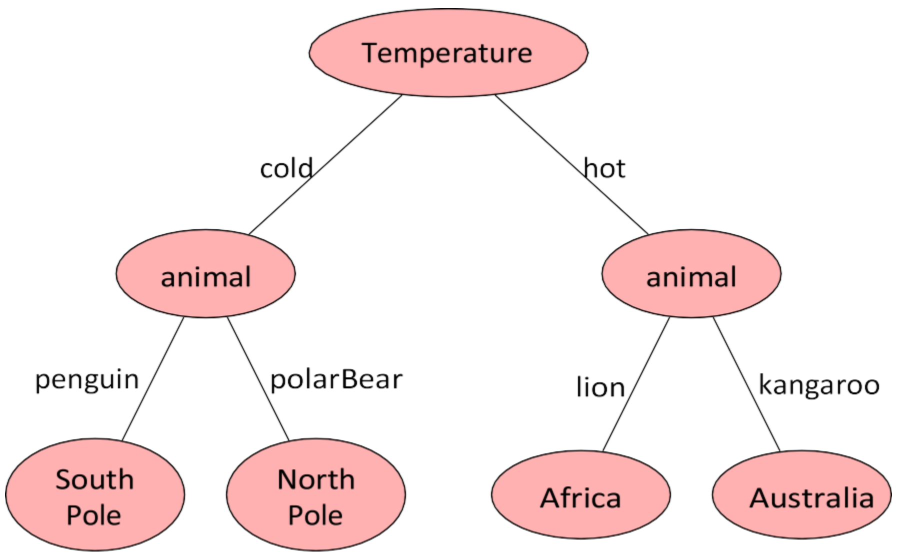


Figure 2.2: Example of a Decision Tree.

Training a Decision Tree with this automated process can result in large Decision Trees with sections of very little power in terms of classification. Additionally, trees tend to be overfitted, which means that they fit the training instances too closely. This results in bad performances when these trees are applied to unseen data. Therefore, a technique called *pruning* has been developed. Its objective is to remove the less or non-productive parts from the Decision Tree, such as parts based on noisy or erroneous data or parts that are overfitted. This often results in further improvements in terms of accuracy and shrinks down the tree size. This process is especially important, due to the fact that every real- world data set contains erroneous or noisy data.

Algorithm *Decision Tree Training Process*

```
1.  training set = S;
2.  attribute set: A;
3.  target Attribute = C;
4.  split criterion = sC;
5.  stop criterion = stop;
6.  $Grow(S, A, C, sC, stop)$
7.  **if** $stop(S)$ = **false**
8.    **then**
9.        **for** all $a_i \in A$
```

```
10.                 do find $a_i$ with the best $sc(S)$;
11.             label current Node with $a$;
12.             for all values $v_i \in a$
13.                 do label outgoing edge with $v_i$
14.                     $Ssub = S$ where $a = v_i$;
15.                     create subNode = $Grow(Ssub, A, C, sC, stop)$;
16.         else currentNode = leaf;
17.             label currentNode with $c_i$ where $c_i$ is most

                common value of $C \in S$;
```

Figure 2.3: Pseudo code of training a Decision Tree

Figure 2.3 shows the process of training a Decision Tree in pseudo code, not taking numeric attributes into account. The algorithm starts by testing whether the stop criterion has been reached or not. If so, the current Node is labeled with the most common value of all existing class labels for the training set. If the stop criterion is not true, the algorithm calculates the split value for all attributes and labels the node with the attribute corresponding to the best split value. Afterwards, it splits the node into multiple nodes, one foreach value of the chosen attribute. The algorithm calls the same process recursively for all training subsets, containing all data records with the corresponding value of the chosenattribute.

2.4.k-nearest neighbor (K-NN)

k-nearest neighbor (K-NN) is a supervised classification learning algorithm used to classify samples. The purpose of this algorithm is to classify a new sample based on its features and labeled training samples. The algorithm is memory-based and does not require a model to be fit. Given a query point x0, k training points closest in distance (Euclidean distance) to x0 are found. Based on the majority of the neighbors found, the new query is classified to its cluster. Any ties in voting are broken at random.

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.
- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the newdata.

2.5. Random Forest

2.5.1 Supervised Random Forests are grown using a collaboration of the bagging and ID3 principles. Each tree in the forest is grown in the following manner. Given a training set, a random subset is sampled (with replacement) and used to construct a tree which resembles the ID3 idea. However, every case in this bootstrap sample is not used to grow the tree. About one third of the bootstrap is left out and considered to be out-ofbag (OOB) data. Also, not every feature is used to construct the tree. A random selection of features is evaluated in each tree. The OOB data is used to get a classification error rate as trees are added to the forest and to measure input variable (feature) importance. After the forest is completed, a sample can be classified by taking a majority vote among all trees in the forest resembling the bootstrap aggregating idea.

2.5.2 Unsupervised Random Forest can be run unsupervised; that is, without any prior information about the input data. Many unsupervised classifiers plot each input case in a multidimensional space based on feature values. If four features are evaluated in classification, each case is plotted in a four dimensional feature space and clustered using Euclidean distance. Random Forest works similarly unsupervised by calculating a variable dissimilarity measure and plotting each case in a space according to dissimilarity measure and clustering using Euclidean distance.

Dissimilarity measures can be found in supervised learning by putting the training data down each tree. If observations i and j fall in the same terminal node, their similarity is increased by one. These values can be stored in matrix form. After the forest has been constructed and similarities have been found, the similarity matrix is divided by the number of trees. The dissimilarity is defined as:

$$D_{ij} = \sqrt{1 - S_{ij}}$$

where S is the similarity matrix. Using multidimensional scaling with dissimilarities as inputs, it is possible to plot points in a Euclidean space where the distance between the points is equal to the dissimilarities. The points can then be clustered.

2.5.3 Synthetic Data When classes are completely unknown, it is necessary to make up or synthesize classes. By smartly creating synthetic data and adding it to the original dataset, we then have two classes: original data, and synthetic data. We can solve this two class problem using Random Forest and evaluate dissimilarity measures. Plotting based off the dissimilarity measures results in data that is able to be clustered appropriately.

2.5.4 Variable Importance

Random Forest can measure variable importance. This is useful for data mining purposes and can assist in unsupervised classification. If we change a single feature’s input value and reclassify the record, we can determine that the feature’s importance is based on the new classification. This is done using OOB data. Each variable m is randomly permuted and the permuted OOB cases are sent through the forest again. Subtracting the number of correctly classified cases using permuted data from the number of correctly classified cases using non-permuted data gives the importance value of variable m. These values are different for each tree, but the average of each value over all trees in the forest gives a raw importance score for each variable .

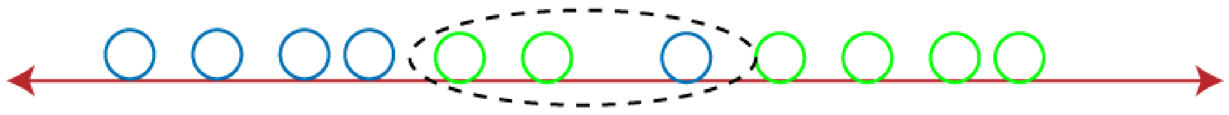
2.6. Linear Discriminant Analysis (LDA)

It is a technique which is frequently used to extract discriminative features that preserve the class separability. LDA involves matrices eigen decomposition which can be computationally expensive in both time and memory, in particular when the number of samples and the number of features are large. This is the case for text and image data sets where the dimension can reach in order of hundreds of thousands or more. By reducing data dimension, we reduce the complexity of data analysis. The accuracy and the computational time of the proposed approach are provided for a wide variety of real image and text data sets. The results show the relevance of the proposed method compared to other methods.

Although the logistic regression algorithm is limited to only two-class, linear Discriminant analysis is applicable for more than two classes of classification problems.

Linear Discriminant analysis is one of the most popular dimensionality reduction techniques used for supervised classification problems in machine learning. It is also considered a pre-processing step for modeling differences in ML and applications of pattern classification.

Whenever there is a requirement to separate two or more classes having multiple features efficiently, the Linear Discriminant Analysis model is considered the most common technique to solve such classification problems. For e.g., if we have two classes with multiple features and need to separate them efficiently. When we classify them using a single feature, then it may show overlapping.



Overlapping

To overcome the overlapping issue in the classification process, we must increase the number of features regularly.

2.7. Gradient Booster

Gradient boosting classifier are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Gradient boosting models are becoming popular because of their effectiveness at classifying complex datasets, and have recently been used to win many data science competitions .In machine learning, there are two types of supervised learning problems:

Classification refers to the task of giving a machine learning algorithm features, and having the algorithm put the instances/data points into one of many discrete classes. Classes are categorical in nature, it isn't possible for an instance to be classified as partially one class and partially another. A classic example of a classification task is classifying emails as either "spam" or "not spam" - there's no "a bit spammy" email.

Regressions are done when the output of the machine learning model is a real value or a continuous value. Such an example of these continuous values would be "weight" or "length". An example of a regression task is predicting the age of a person based off of features like height, weight, income, etc.

Gradient boosting classifiers are specific types of algorithms that are used for classification tasks, as the name suggests.

Features are the inputs that are given to the machine learning algorithm, the inputs that will be used to calculate an output value. In a mathematical sense, the features of the dataset are the variables used to solve the equation. The other part of the equation is the label or target, which are the classes the instances will be categorized into. Because the labels contain the target values for the machine learning classifier, when training a classifier we should split up the data into training and testing sets. The training set will have targets/labels, while the testing set won't contain these values.

Scikit-Learn, or "sklearn", is a machine learning library created for Python, intended to expedite machine learning tasks by making it easier to implement machine learning algorithms. It has easy-to-use functions to assist with splitting data into training and testing sets, as well as training a model, making predictions, and evaluating the model.

3.Literature Survey

Brain tumor is an abnormal mass of tissue with uncoordinated growth inside the skull which may invade and damage nerves and other healthy tissues. Non-homogeneities of the brain tissues result in inaccurate detection of tumor boundaries with the existing methods for contrast enhancement and segmentation of magnetic resonance images (MRI). This paper presents an improved framework for detection of brain tumor.

Automated detection and segmentation of brain abnormalities is therefore a challenging problem of research since decades. Solid brain tumors are malignant masses of tissues formed inside skull as a result of abnormal and uncoordinated growth due to proliferation of atypical cells.

Alfonse and Salem [1] have presented a technique for automatic classification of brain tumor from MR images using an SVM-based classifier. To improve the accuracy of the classifier,features are extracted using fast Fourier transform (FFT)and reduction of features is performed using Minimal-Redundancy-Maximal-Relevance (MRMR) technique. This technique has obtained an accuracy of 98.9%.

Damodharan and Raghavan [2] have presented a neural network based technique for brain tumor detection and classification. In this method, the quality rate is produced separately for segmentation of WM, GM, CSF, and tumor region and claims an accuracy of 83% using neural network based classifier.

Owasis et al [3] proposed how AI assists in the diagnosis and prediction of a disease, it is essential to understand the use and applicability of diverse techniques such as SVM, KNN, Decision Tree, Random Forest and Long short-term memory (LSTM) and many others for various disease detection system (Owasis et al. 2019; Nithya et al. 2020). We conducted an extensive survey based on the machine and deep learning models for disease diagnosis. The study covers the review of various diseases and their diagnostic methods using AI techniques.

Mohamed A. Naser and M. J amal Deen [4] (2020) in their research paper illustrates transfer learning models and use of deep learning for MRI images . It gives exact automatic grading of LGG brain tumors with detection. Full automation is acheived by the use of pipeline of MRI images. It allows simultaneous grading and segmentation of the brain tumors. Author claims that this method shows a promising results as a non- invasive tool for tumors characterization in LGG.

Ijaz et al. [5] proposed (2020) a cervical cancer prediction model for early prediction of cervical cancer using risk factors as inputs. The authors utilize several machine learning approaches and outlier detection for different pre-processing tasks.

Desautels et al. [6] proposed a transfer learning-based machine learning algorithm. They achieved good discrimination. Unplanned readmission prediction can be used to target resources more efficiently.

Flores et al. [7] proposed a method that evaluate heterogeneous and missing clinical data using unsupervised machine learning algorithm to detect important artery disease.

S. Banerjee, S. Mitra, B. U. Shankar, [8] reports the accuracy achieved by seven standard classifiers,viz. i) Decision Tree, ii) Random Forest, iii) Logistic Regression (LR), iv) Linear Discriminant Analysis (LDA), v) Support Vector Machine (SVM), vi) Gradient Booster, and vii) k-nearest neighbours (k-NN).

The accuracy reported in International Journal of Computer Science and communication Vol. 2, No. 2, JulyDecember-2011,pp. 325-331 [9]is on the BRaTS 2015 dataset (a subset of BRaTS 2017 dataset) which consists of 200 HGG and 54 LGG cases. 56 three-dimensional quantitative MRI features extracted manually from each patient MRI and used for the classification

4. Proposed Methodology

The proposed work is to analyse the images MRI based brain tumor detection using supervised machine learning algorithm . It has been used several algorithms i.e. Logistics Regression, Support Vector Machine, Decision Tree, KNN, Random Forest , Linear Discriminant Analysis and Gradient Booster.

First, I have implemented the mother paper which have been implemented with logistic regression, support vector machine and KNN Algorithm . While implementing mother paper first import the required modules. Then I have started to prepare the data. The training dataset is classified as "no" means no tumor in brain and "yes" means tumor in brain. I have marked "no" as 0 and "yes" as 1. Then I have created two class i.e. X and Y and have appended the images in X and mark the classification of images in Y and while appending all the images have been resized into one specific size. Then I have turned X and Y into numpy array. After that splitting the dataset into test and train data. Then features has been scaled and selected with PCA. After selecting the features the model has been trained with support vector machine algorithm, logistic regression algorithm and KNN algorithm. Then each model is evaluated and predicted and tested.

After implementing the mother dataset I have implemented four more algorithms i.e. random forest algorithm, decision tree algorithm, linear discriminant analysis algorithm and gradient booster algorithm.

First import the required modules. Then I have started to prepare the data. The training dataset is classified as "no" means no tumor in brain and "yes" means tumor in brain. I have marked "no" as 0 and "yes" as 1. Then I have created two classes i.e. X and Y and have appended the images in X and mark the classification of images in Y and while appending all the images have been resized into one specific size. Then I have turned X and Y into numpy array. After that splitting the dataset into test and train data. Then features has been scaled and selected with PCA. After selecting the features the model has been trained with support vector machine algorithm, logistic regression algorithm, KNN algorithm, random forest algorithm, decision tree algorithm, gradient booster algorithm and linear discriminant analysis algorithm .Then each model is evaluated and predicted and tested.

5.Experimental Result and Evaluation

The dissertation is done with several supervised machine learning algorithms i.e. Logistic Regression, Support Vector Machine Algorithm, KNN, Random Forest Algorithm, Decision Tree Algorithm, Gradient Booster Algorithm and Linear Discriminant Analysis Algorithm.

Logistic Regression

Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring.

```
logclf.fit(xtrain, ytrain)
logclf.score(xtest, ytest)
0.926530612244898
```

Fig:1

Fig:1 shows the experimental result for Logistic Regression.

Decision Tree

Decision Trees are a type of Supervised Machine Learning (that is you explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves.

```
treeclf.fit(xtrain, ytrain)
treeclf.score(xtest, ytest)
0.963265306122449
```

Fig:2

Fig:2 shows the experimental result for Decision Tree.

Random Forest

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

```
rf_clf.fit(xtrain, ytrain)
rf_clf.score(xtest, ytest)
0.9959183673469387
```

Fig:3

Fig:3 shows the experimental result for Random Forest .

KNN

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

```
print("Training Score:", KNN_model.score(pca_train, ytra
in))
print("Testing Score:", KNN_model.score(pca_test, ytest)
)
```

```
Training Score: 1.0
Testing Score: 0.9510204081632653
```

Fig:4

Fig:4 shows the experimental result for KNN .

LDA

Linear discriminant analysis (LDA) is used here to reduce the number of features to a more manageable number before the process of classification. Each of the new dimensions generated is a linear combination of pixel values, which form a template.

```
LDAclf.fit(xtrain, ytrain)
LDAclf.score(xtest, ytest)
0.8081632653061225
```

Fig:5

Fig:5 shows the experimental result for LDA .

Gradient Boosting

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.

```
gbclf = ensemble.GradientBoostingClassifier()
gbclf.fit(xtrain, ytrain)
gbclf.score(xtest, ytest)
0.9877551020408163
```

Fig:6

Fig:6 shows the experimental result for Gradient Boosting.

SVM

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.

```
print("Training Score:", sv.score(pca_train, ytrain))
print("Testing Score:", sv.score(pca_test, ytest))
```

```
Training Score: 0.9907881269191402
Testing Score: 0.9673469387755103
```

Fig:7

Fig:7 shows the experimental result for SVM.

SL No.	Name of Algorithm	Result
1	*Logistic Regression*	0.9265
2	*Decision Tree*	*0.9632*
3	*Random Forest*	*0.9959*
4	*KNN*	*0.9510*
5	*LDA*	*0.8081*
6	*Gradient Boosting*	*0.9877*
7	*SVM*	*0.9673*

After exploring the confusion matrix and score of all the algorithms it is found that Random Forest Algorithm gives better result i.e. 0.9959183673469387.

6. Conclusion and Future work

From the implementation code it has been seen that among all the algorithms like Logistic Regression , SVM, Decision Tree , Random Forest , Gradient Booster, it can be said that Random Forest Algorithm gives best result .

This work can be further experimented with unsupervised learning algorithms in future. The dataset would be very large for unsupervised learning algorithm.

7. References

1. M. Alfonse , A. B. M. Salem, “An automatic classification of brain tumors through MRI using support vector machine,” Egyptian Computer Science Journal, vol. 40, pp. 11–21, 2016.

2. T. Torheim, E. Malinen, K. Kvaal et al., “Classification of dynamic contrast enhanced MR images of cervical cancers using texture analysis and support vector machines,” IEEE Transactions on Medical Imaging, vol. 33, no. 8, pp. 1648–1656, 2014.

3. A. Hebli, S. Gupta, (2017). Brain tumor prediction and classification using support vector machine. In 2017 international conference on advances in computing,communication and control (ICAC3) (pp. 1–6). IEEE.

4. M. Ghose, R. Pradhan, and S. Ghose, “Decision Tree Classification of Remotely Sensed Satellite Data using Spectral Separability Matrix,” Int. J. Adv. Comput. Sci. Appl., vol. 1, no. 5, pp. 93–101, 2010.

5. Anon, (2017). [online] Available at: http://Wikipedia contributors. "K-nearest neighbors algorithm." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 28 Nov. 2016. Web. 28 Nov. 2016 [Accessed 4 Dec. 2017].

6. Brownlee, Jason A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning. Machine Learning Mastery, 21 Sept. 2016, machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithmmachine-learning/

7. Polamuri, Saimadhu How the Random Forest Algorithm Works in Machine Learning. Dataaspirant, 1 Oct. 2017, dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/

8. Park, Hyeoun-Ae An Introduction to Logistic Regression: From Basic Concepts to Interpretation with Particular Attention to Nursing Domain. Journal of Korean Academy of Nursing, vol. 43, no. 2, 2013, p. 154., doi:10.4040/jkan.2013.43.2.154.

9. R. Kumari, “SVM classification an approach on detecting abnormality in brain MRI images,” International Journal of Engineering Research and Applications, vol. 3, pp. 1686–1690, 2013.

10. M. Alfonse and A.-B. M. Salem, "An automatic classification of brain tumors through MRI using support vector machine," Egyptian Computer Science Journal, vol. 40, pp. 11–21, 2016.

11. L. Guo, L. Zhao, Y. Wu, Y. Li, G. Xu, and Q. Yan, "Tumor detection in MR images using one-class immune feature weighted SVMs," IEEE Transactions on Magnetics, vol. 47, no. 10, pp.3849–3852, 2011.

12. N. Gordillo, E. Montseny, and P. Sobrevilla, "State of the art survey on MRI brain tumor segmentation," Magnetic Resonance Imaging, vol. 31, no. 8, pp. 1426–1438, 2013.

13. Javaria Amin a , Muhammad Sharif a , ∗ , Mudassar Raza a , Tanzila Saba b ,Muhammad Almas Anjum c Brain tumor detection using machine learning method", https://doi.org/10.1016/j.cmpb.2019.05.015 0169-2607/© 2019 Published by Elsevier B.V.,Science Direct Computer Methods and Programs in Biomedicine.

14. Sergio Pereira, Adriano Pinto, Victor Alves, and Carlos A. Silva ―Brain Tumor Segmentation Using Convolutional Neural Networks in MRI Images‖IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 35, NO. 5, MAY 2016

Appendix

Data of patients from different hospital is collected.

Mounting to drive

```
[]
from google.colab import drive
drive.mount('/content/drive')
Mounted at /content/drive
```

Load Modules

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import RandomizedSearchCV,
KFold, train_test_split
from sklearn.metrics import accuracy_score
import cv2
from tensorflow import keras
```

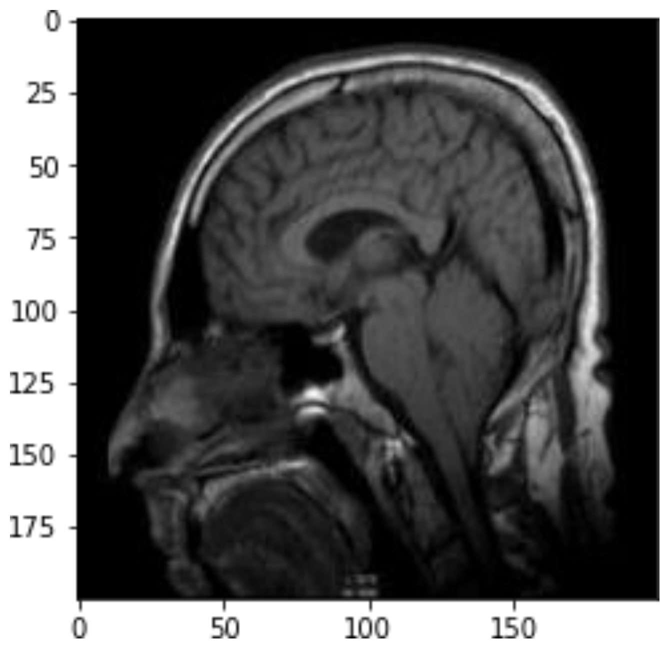
Prepare/collect the data

```
import os

path = os.listdir('/content/drive/MyDrive/Project/main/T
raining/')
classes = {'no':0, 'yes':1}

import cv2
X = []
Y = []
for cls in classes:
    pth = '/content/drive/MyDrive/Project/main/Training/
'+cls
    for j in os.listdir(pth):
        img = cv2.imread(pth+'/'+j, 0)
        img = cv2.resize(img, (200,200))
        X.append(img)
        Y.append(classes[cls])
```

```
X = np.array(X)
Y = np.array(Y
np.unique(Y)

array([0, 1])

pd.Series(Y).value_counts()



      1  827
      0  395
      dtype: int64

X.shape
(1222, 200, 200)
```

Visualize data

```
plt.imshow(X[0], cmap='gray')
<matplotlib.image.AxesImage at 0x7f4428397dd0>
```



Prepare Data

```
X_updated = X.reshape(len(X), -1)
X_updated.shape
```

```
(1222, 40000)
```

Splitting Data

```
xtrain, xtest, ytrain, ytest = train_test_split(X_update
d, Y, random_state=10,test_size=.20)
xtrain.shape, xtest.shape
```

```
((977, 40000), (245, 40000))
```

Feature Scaling

```
print(xtrain.max(), xtrain.min())
print(xtest.max(), xtest.min())
xtrain = xtrain/255
xtest = xtest/255
print(xtrain.max(), xtrain.min())
print(xtest.max(), xtest.min())
```

```
    255 0
      255 0
      1.0 0.0
      1.0 0.0
```

Feature Selection: PCA

```
from sklearn.decomposition import PCA
print(xtrain.shape, xtest.shape)

pca = PCA(.98)
# pca_train = pca.fit_transform(xtrain)
# pca_test = pca.transform(xtest)
pca_train = xtrain
pca_test = xtest
```

```
(977, 40000) (245, 40000)
```

```
print(pca_train.shape, pca_test.shape)
#print(pca.n_components_)
#print(pca.n_features_)
(977, 40000) (245, 40000)
```

Train Model

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

import warnings
warnings.filterwarnings('ignore')

lg = LogisticRegression(C=0.1)
lg.fit(xtrain, ytrain)

LogisticRegression(C=0.1)
sv = SVC()

sv.fit(pca_train, ytrain)
SVC()

from sklearn import neighbors
#KNN_model=neighbors.KNeighborsClassifier(n_neighbors=be
st_k,n_jobs=-1)

#KNN_model.fit(pca_train,pca_train)
```

```
from sklearn.metrics import f1_score,confusion_matrix,ro
c_auc_score
f1_list=[]
k_list=[]

for k in range(1,10):
    clf=neighbors.KNeighborsClassifier(n_neighbors=k,n_j
obs=-1)
    clf.fit(xtrain,ytrain)
    pred=clf.predict(xtest)
    f=f1_score(ytest,pred,average='macro')
    f1_list.append(f)
    k_list.append(k)

best_f1_score=max(f1_list)
best_k=k_list[f1_list.index(best_f1_score)]
print("Optimum K value=",best_k," with F1-
Score=",best_f1_score)

Optimum K value= 4 with F1-Score= 0.9623829264547827

KNN_model=neighbors.KNeighborsClassifier(n_neighbors=1,n
_jobs=-1)
KNN_model.fit(xtrain,ytrain)

KNeighborsClassifier(n_jobs=-1, n_neighbors=1)
```

Evaluation

```
print("Training Score:", lg.score(pca_train, ytrain))
print("Testing Score:",lg.score(pca_test, ytest))

Training Score: 1.0
Testing Score: 0.9714285714285714

print("Training Score:", sv.score(pca_train, ytrain))
print("Testing Score:", sv.score(pca_test, ytest))

Training Score: 0.9907881269191402
Testing Score: 0.9673469387755103
```

```
print("Training Score:", KNN_model.score(pca_train, ytra
in))
print("Testing Score:", KNN_model.score(pca_test, ytest)
)

Training Score: 1.0
Testing Score: 0.9510204081632653
```

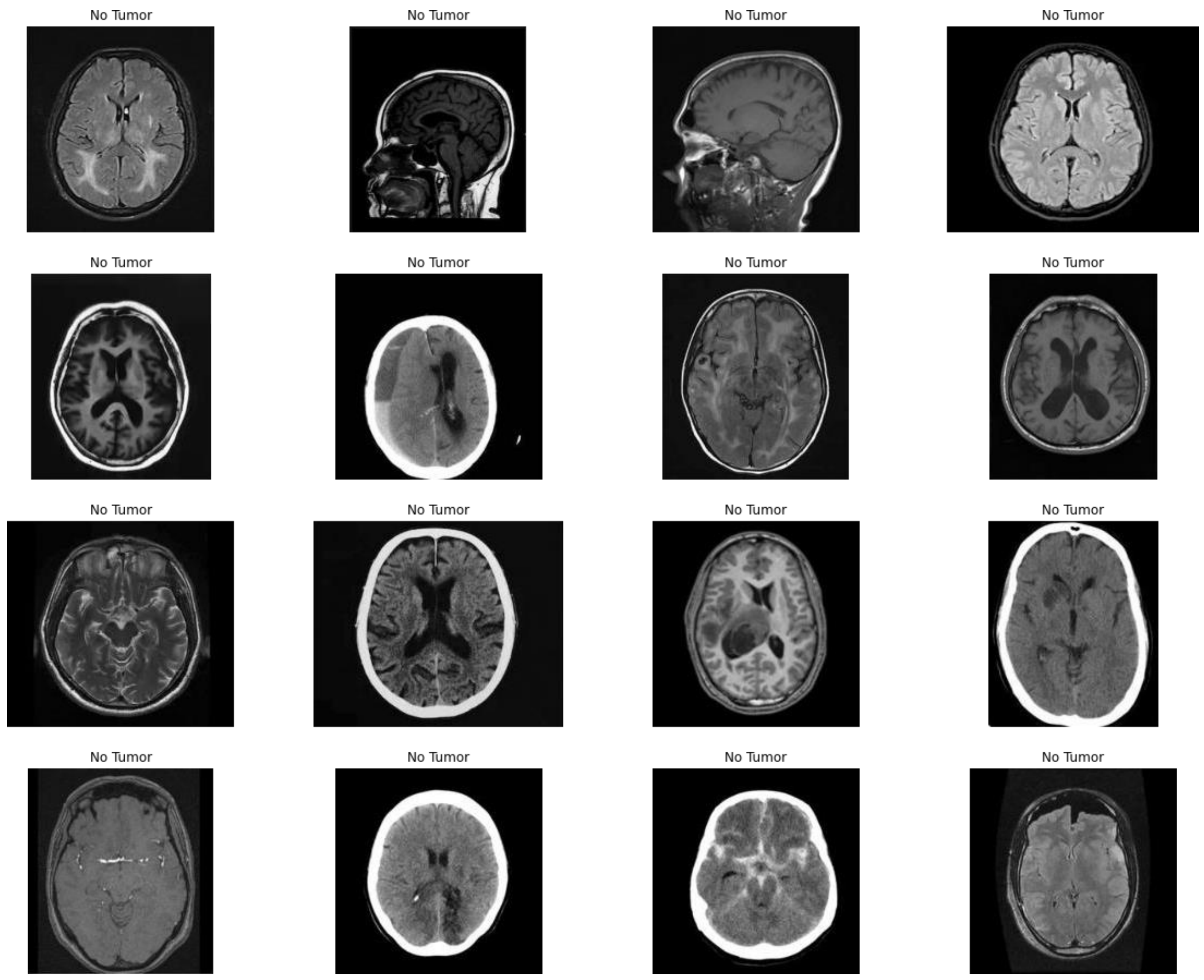
Prediction

```
pred = sv.predict(pca_test)
np.where(ytest!=pred)

(array([ 49, 51, 59, 65, 68, 166, 171, 238]),)

pred[4]

0

ytest[36]

1

pred = lg.predict(pca_test)
np.where(ytest!=pred)

(array([ 3, 49, 50, 59, 65, 166, 238]),)

pred=KNN_model.predict(pca_test)
np.where(ytest!=pred)
#print("Accuracy={}%".format((sum(ytest==pred)/ytest.sha
pe[0])*100))

(array([ 41, 49, 50, 59, 65, 112, 137, 160, 166, 169,
171, 238]),)
```

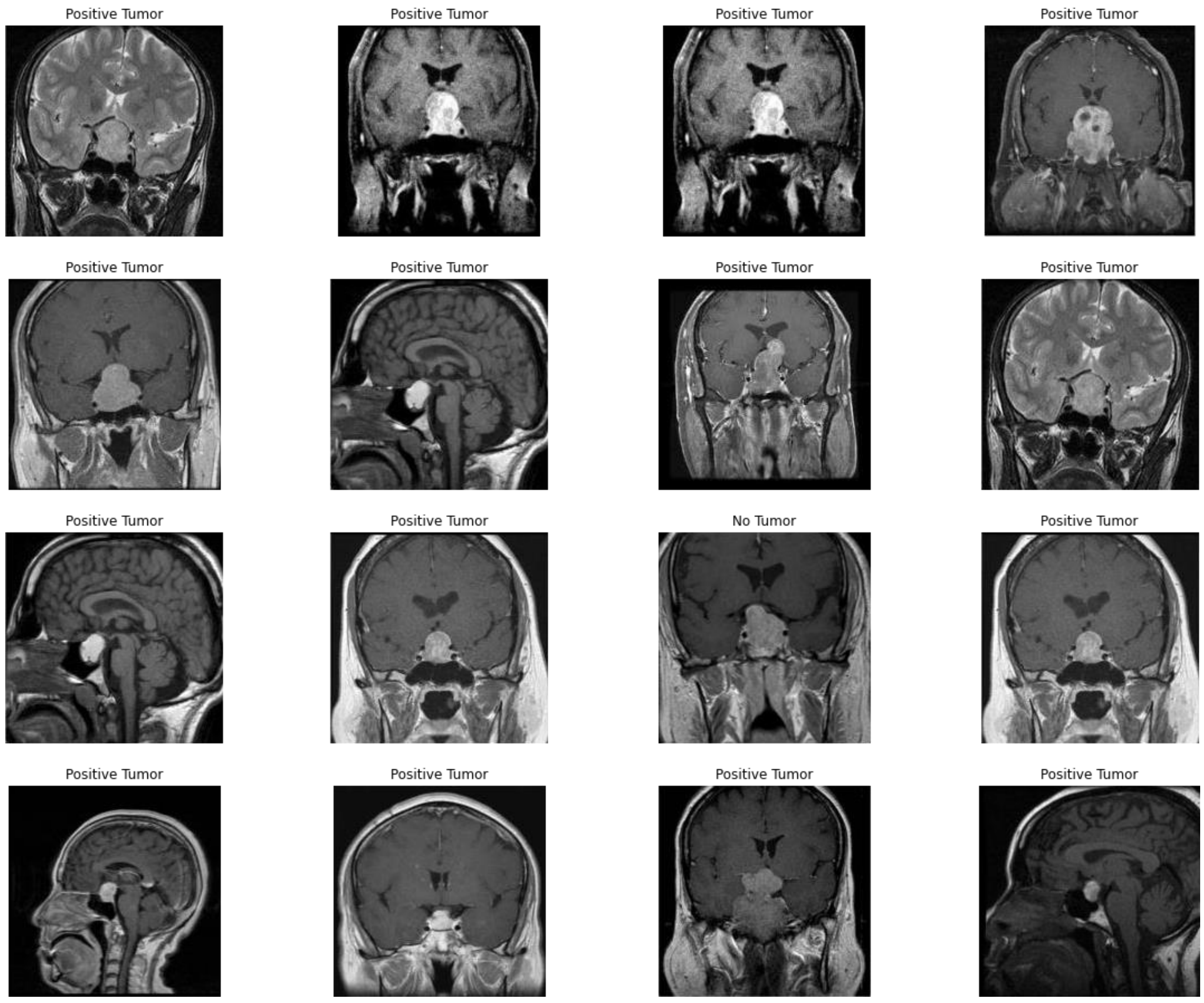
TEST MODEL

```
dec = {0:'No Tumor', 1:'Positive Tumor'}

plt.figure(figsize=(20,16))
```

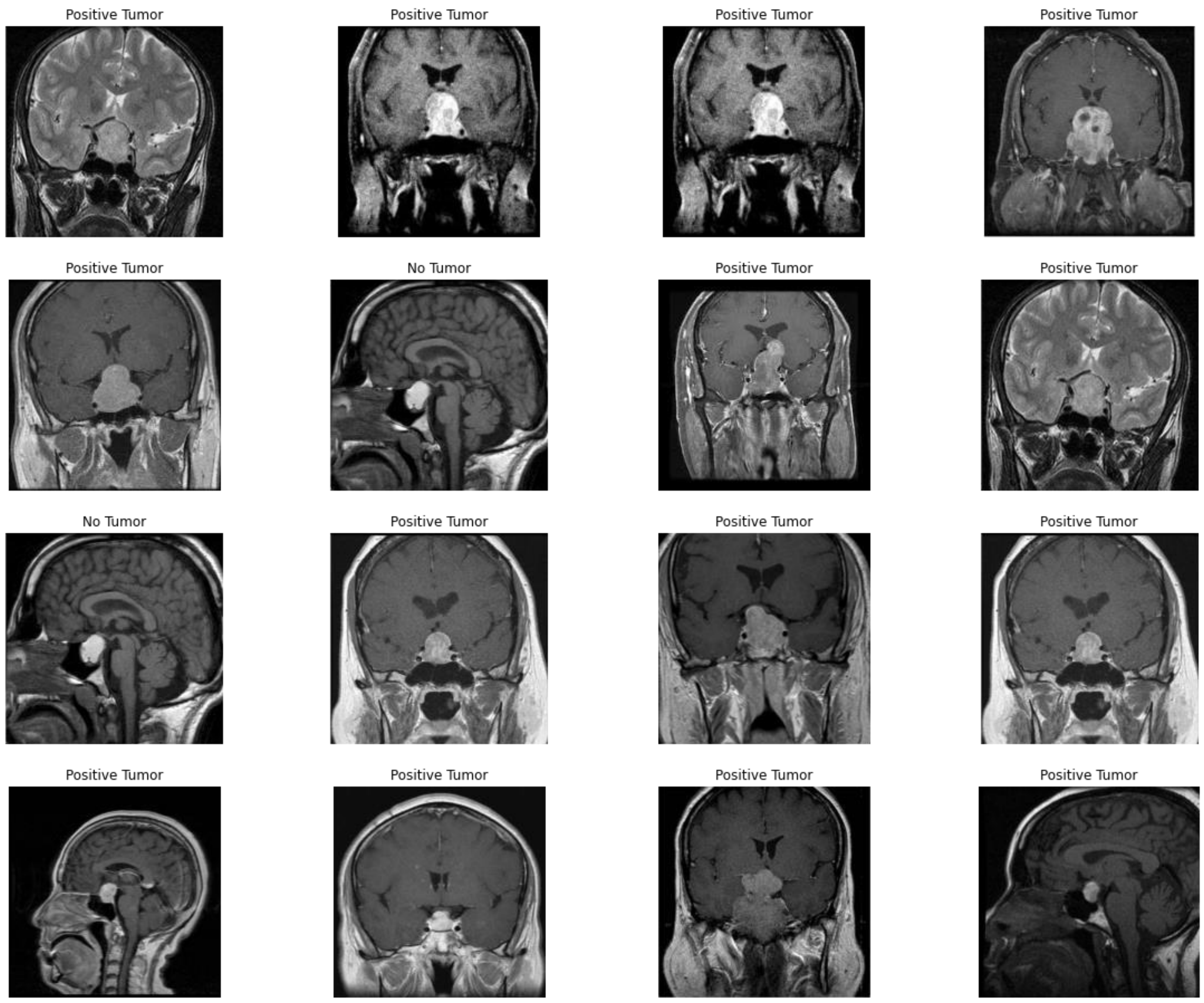
```
p = os.listdir('/content/drive/MyDrive/Project/main/Test
ing/')
c=1
for i in os.listdir('/content/drive/MyDrive/Project/main
/Testing/no/')[:16]:
    plt.subplot(4,4,c)

    img = cv2.imread('/content/drive/MyDrive/Project/mai
n/Testing/no/'+i,0)
    img1 = cv2.resize(img, (200,200))
    img1 = img1.reshape(1,-1)/255
    p = sv.predict(img1)
    plt.title(dec[p[0]])
    plt.imshow(img, cmap='gray')
    plt.axis('off')
    c+=1
```



```
plt.figure(figsize=(20,16))
p = os.listdir('/content/drive/MyDrive/Project/main/Test
ing/')
c=1
for i in os.listdir('/content/drive/MyDrive/Project/main
/Testing/yes/')[:16]:
    plt.subplot(4,4,c)

    img = cv2.imread('/content/drive/MyDrive/Project/mai
n/Testing/yes/'+i,0)
    img1 = cv2.resize(img, (200,200))
    img1 = img1.reshape(1,-1)/255
    p = lg.predict(img1)
    plt.title(dec[p[0]])
    plt.imshow(img, cmap='gray')
    plt.axis('off')
    c+=1
```



```
    plt.figure(figsize=(20,16))
p = os.listdir('/content/drive/MyDrive/Project/main/Testing/')
c=1
for i in os.listdir('/content/drive/MyDrive/Project/main/Testing/yes/')[:16]:
    plt.subplot(4,4,c)

    img = cv2.imread('/content/drive/MyDrive/Project/main/Testing/yes/'+i,0)
    img1 = cv2.resize(img, (200,200))
    img1 = img1.reshape(1,-1)/255
    p = KNN_model.predict(img1)
    plt.title(dec[p[0]])
    plt.imshow(img, cmap='gray')
    plt.axis('off')
    c+=1
```



Classifier

```
import pandas as pd
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
#from pydataset import data
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
#from mlxtend.classifier import EnsembleVoteClassifier
from sklearn.discriminant_analysis import LinearDiscrimi
nantAnalysis as LDA
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

import os
path = os.listdir('/content/drive/MyDrive/Project/main/T
raining/')
classes = {'no':0, 'yes':1}

import cv2
X = []
Y = []
for cls in classes:
    pth = '/content/drive/MyDrive/Project/main/Training/
'+cls
    for j in os.listdir(pth):
        img = cv2.imread(pth+'/'+j, 0)
        img = cv2.resize(img, (200,200))
        X.append(img)
        Y.append(classes[cls])

X = np.array(X)
Y = np.array(Y)

np.unique(Y)
array([0, 1])

pd.Series(Y).value_counts()
      1  827
      0  395
      dtype: int64

X.shape
(1222, 200, 200)

X_updated = X.reshape(len(X), -1)
X_updated.shape
(1222, 40000)

xtrain, xtest, ytrain, ytest = train_test_split(X_update
d, Y, random_state=10,test_size=.20)
xtrain.shape, xtest.shape
((977, 40000), (245, 40000))

print(xtrain.max(), xtrain.min())
print(xtest.max(), xtest.min())
xtrain = xtrain/255
xtest = xtest/255
print(xtrain.max(), xtrain.min())
print(xtest.max(), xtest.min())
255 0
255 0
1.0 0.0
1.0 0.0

logclf=LogisticRegression(penalty='l2',C=0.001, random_s
tate=0)
treeclf=DecisionTreeClassifier(max_depth=3,criterion='en
tropy',random_state=0)
knnclf=KNeighborsClassifier(n_neighbors=1,p=2,metric='mi
nkowski')
LDAclf=LDA()

treeclf.fit(xtrain, ytrain)
treeclf.score(xtest, ytest)
0.963265306122449

y_pred = treeclf.predict(xtest)

#from sklearn.metrics import confusion_matrix
confusion_matrix(ytest, y_pred)
```

```
array([[ 75,   6],
       [  3, 161]])

from sklearn import ensemble
rf_clf = ensemble.RandomForestClassifier(n_estimators=10
0)
rf_clf.fit(xtrain, ytrain)
rf_clf.score(xtest, ytest)
0.9959183673469387

logclf.fit(xtrain, ytrain)
logclf.score(xtest, ytest)
0.926530612244898
y_pred = logclf.predict(xtest)

#from sklearn.metrics import confusion_matrix
confusion_matrix(ytest, y_pred)
array([[ 63,  18],
       [  0, 164]])

knnclf.fit(xtrain, ytrain)
knnclf.score(xtest, ytest)
0.9510204081632653

y_pred = knnclf.predict(xtest)

#from sklearn.metrics import confusion_matrix
confusion_matrix(ytest, y_pred)
array([[ 70,  11],
       [  1, 163]])

LDAclf.fit(xtrain, ytrain)
LDAclf.score(xtest, ytest)
0.8081632653061225

y_pred = LDAclf.predict(xtest)

#from sklearn.metrics import confusion_matrix
confusion_matrix(ytest, y_pred)
array([[ 57,  24],
       [ 23, 141]])

gbclf = ensemble.GradientBoostingClassifier()
gbclf.fit(xtrain, ytrain)
gbclf.score(xtest, ytest)
0.9877551020408163

# Let's  tune this Gradient booster.
gbclf = ensemble.GradientBoostingClassifier(n_estimators
=50)
gbclf.fit(xtrain,ytrain)
gbclf.score(xtest, ytest)
0.9795918367346939

# Importing library

#from scipy.stats import kurtosis

# Creating a dataset
#dataset = xtrain[0]


# Calculate the kurtosis
#print(kurtosis(dataset, axis=0, bias=True))
```

DL

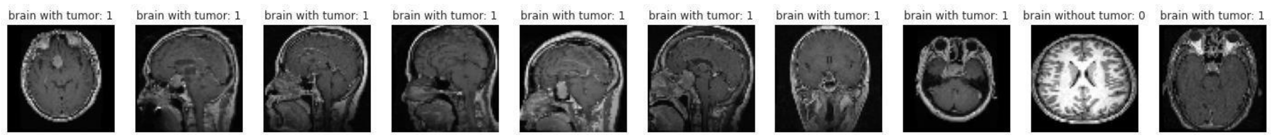
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g
. pd.read_csv)
import os
import matplotlib.pyplot as plt
import seaborn as sb
sb.set_style('whitegrid')
import tensorflow as tf
from tensorflow.keras import layers,models,optimizers,pr
eprocessing
from tensorflow.keras.layers import Conv2D,Dense,MaxPool
ing2D,Flatten,Dropout

main_dir = "/content/drive/MyDrive/Project/main/Training
/"
classification_dirs = [("no",), ("yes",)]
resolution = 64
```

```
def load_images(root_dir_name):
    x = []
    y = []

    for label, sub_dir_names in enumerate(classification
_dirs):
        for sub_dir_name in sub_dir_names:
            print(f"loading {root_dir_name} {sub_dir_nam
e}")
            sub_dir_path = os.path.join(main_dir, root_d
ir_name, sub_dir_name)
            for image_name in os.listdir(sub_dir_path):
                image_path = os.path.join(sub_dir_path,
image_name)
                image = preprocessing.image.load_img(ima
ge_path, color_mode="grayscale", target_size=(resolution
, resolution))
                x.append(preprocessing.image.img_to_arra
y(image))
                y.append(label)


    x = np.array(x) / 255.0
    y = np.array(y)

    return x, y
    return x, y

x_train, y_train = load_images("/content/drive/MyDrive/P
roject/main/Training/")
x_test, y_test = load_images("/content/drive/MyDrive/Pro
ject/main/Testing/")
```

```
loading /content/drive/MyDrive/Project/main/Training/ no
loading /content/drive/MyDrive/Project/main/Training/ yes
loading /content/drive/MyDrive/Project/main/Testing/ no
loading /content/drive/MyDrive/Project/main/Testing/ yes
```

```
x_train.shape
(1222, 64, 64, 1)

x_test.shape
(179, 64, 64, 1)
```

```
c = 10

fig, subplots = plt.subplots(1, c)
fig.set_size_inches(25, 3)
for i in range(c):
    n = np.random.randint(0, len(x_train))
    num = y_train[n]
    word = "out" if num == 0 else ""

    subplots[i].imshow(x_train[n].reshape((resolution, r
esolution)), cmap="gray")
    subplots[i].set_title(f"brain with{word} tumor: {num
}")
    subplots[i].axis("off")
plt.show()
```



```
input_shape = (64,64,1)
model = models.Sequential()
model.add(Conv2D(32,kernel_size = (2,2),strides = (1,1),
activation = 'linear',input_shape = input_shape))
model.add(MaxPooling2D(pool_size = (2,2),strides = (2,2)
))
model.add(Conv2D(64,kernel_size = (2,2),strides = (1,1),
activation = 'linear'))
model.add(MaxPooling2D(pool_size = (2,2),strides = (2,2)
))
model.add(Conv2D(128,kernel_size = (2,2),strides = (1,1)
,activation = 'linear'))
model.add(MaxPooling2D(pool_size = (2,2),strides = (2,2)
))
model.add(Conv2D(256,kernel_size = (2,2),strides = (1,1)
,activation = 'linear'))
model.add(MaxPooling2D(pool_size = (2,2),strides = (2,2)
))
model.add(Conv2D(512,kernel_size = (2,2),strides = (1,1)
,activation = 'linear'))
model.add(MaxPooling2D(pool_size = (2,2),strides = (2,2)
))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(256, activation="linear"))
model.add(Dense(1, activation="sigmoid"))
model.summary()
Model: "sequential"
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
 Layer (type)                         Output Shape                 Param#
==========================================================================
====
 conv2d (Conv2D)                        (None, 63, 63, 32)             160

 max_pooling2d (MaxPooling2D )          (None, 31, 31, 32)               0

 conv2d_1 (Conv2D)                      (None, 30, 30, 64)            8256

 max_pooling2d_1 (MaxPooling2D)            (None, 15, 15, 64)            0


 conv2d_2 (Conv2D)                          (None, 14, 14, 128)        32896

 max_pooling2d_2 (MaxPooling2D)                (None, 7, 7, 128)         0


 conv2d_3 (Conv2D)                            (None, 6, 6, 256)     131328

 max_pooling2d_3 (MaxPooling2D)                   (None, 3, 3, 256)    0


 conv2d_4 (Conv2D)                              (None, 2, 2, 512)    524800

 max_pooling2d_4 (MaxPooling 2D)                     (None, 1, 1, 512)    0


 flatten (Flatten)                                     (None, 512)        0

 dropout (Dropout)                                   (None, 512)          0

 dense (Dense)                                     (None, 256)       131328

 dense_1 (Dense)                                      (None, 1)        257

==========================================================================
Total params: 829,025
Trainable params: 829,025
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer = 'rmsprop',loss = "binary_cross
entropy",metrics = ['accuracy'])

model.fit(x_train,y_train,batch_size = 7,epochs = 25, va
lidation_data=(x_test, y_test))
```

```
Epoch 1/25
175/175 [==============================] - 9s 46ms/step - loss:
0.4879 - accuracy: 0.7962 - val_loss: 0.4224 - val_accuracy: 0.7877
Epoch 2/25
175/175 [==============================] - 9s 49ms/step - loss:
0.1961 - accuracy: 0.9411 - val_loss: 0.1758 - val_accuracy: 0.8939
Epoch 3/25
175/175 [==============================] - 8s 46ms/step - loss:
0.1271 - accuracy: 0.9656 - val_loss: 0.3089 - val_accuracy: 0.9050
Epoch 4/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0836 - accuracy: 0.9755 - val_loss: 0.6070 - val_accuracy: 0.8436
Epoch 5/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0444 - accuracy: 0.9877 - val_loss: 0.0839 - val_accuracy: 0.9665
Epoch 6/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0609 - accuracy: 0.9853 - val_loss: 0.4211 - val_accuracy: 0.9050
Epoch 7/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0092 - accuracy: 0.9975 - val_loss: 0.0882 - val_accuracy: 0.9497
Epoch 8/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0474 - accuracy: 0.9885 - val_loss: 0.3767 - val_accuracy: 0.9050
Epoch 9/25
175/175 [==============================] - 8s 45ms/step - loss:
0.0333 - accuracy: 0.9926 - val_loss: 0.9540 - val_accuracy: 0.8547
Epoch 10/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0104 - accuracy: 0.9984 - val_loss: 0.9468 - val_accuracy: 0.8883
Epoch 11/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0267 - accuracy: 0.9935 - val_loss: 1.0687 - val_accuracy: 0.8771
```

```
Epoch 12/25
175/175 [==============================] - 9s 52ms/step - loss:
0.0404 - accuracy: 0.9959 - val_loss: 1.0717 - val_accuracy: 0.8883
Epoch 13/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0112 - accuracy: 0.9967 - val_loss: 0.5373 - val_accuracy: 0.9497
Epoch 14/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0127 - accuracy: 0.9951 - val_loss: 0.0567 - val_accuracy: 0.9665
Epoch 15/25
175/175 [==============================] - 8s 47ms/step - loss:
0.0168 - accuracy: 0.9967 - val_loss: 0.3284 - val_accuracy: 0.9553
Epoch 16/25
175/175 [==============================] - 8s 47ms/step - loss:
0.0248 - accuracy: 0.9975 - val_loss: 0.0936 - val_accuracy: 0.9721
Epoch 17/25
175/175 [==============================] - 8s 47ms/step - loss:
0.0093 - accuracy: 0.9975 - val_loss: 0.1757 - val_accuracy: 0.9721
Epoch 18/25
175/175 [==============================] - 8s 47ms/step - loss:
0.0136 - accuracy: 0.9975 - val_loss: 1.9985 - val_accuracy: 0.9106
Epoch 19/25
175/175 [==============================] - 8s 47ms/step - loss:
0.0065 - accuracy: 0.9975 - val_loss: 1.4758 - val_accuracy: 0.9106
Epoch 20/25
175/175 [==============================] - 8s 46ms/step - loss:
0.0207 - accuracy: 0.9975 - val_loss: 3.8979 - val_accuracy: 0.8324
Epoch 21/25
175/175 [==============================] - 8s 47ms/step - loss:
0.0041 - accuracy: 0.9992 - val_loss: 0.4383 - val_accuracy: 0.9274
Epoch 22/25
175/175 [==============================] - 8s 47ms/step - loss:
1.4356e-04 - accuracy: 1.0000 - val_loss: 1.3686 - val_accuracy: 0.8994
Epoch 23/25
175/175 [==============================] - 8s 47ms/step - loss:
1.4269e-04 - accuracy: 1.0000 - val_loss: 0.0026 - val_accuracy: 1.0000
Epoch 24/25
175/175 [==============================] - 8s 47ms/step - loss:
1.2337e-05 - accuracy: 1.0000 - val_loss: 1.2157 - val_accuracy: 0.8883
Epoch 25/25
175/175 [==============================] - 8s 47ms/step - loss:
5.4701e-06 - accuracy: 1.0000 - val_loss: 0.8670 - val_accuracy: 0.9162
<keras.callbacks.History at 0x7fd4e4ce4590>
```

```
y_test_results = model.predict(x_test)

c = 10
fig, subplots = plt.subplots(1, c)
fig.set_size_inches(30, 9)
for i in range(c):
    n = np.random.randint(0, len(x_test))
    guess = str(round(y_test_results[n][0], 2)).ljust(4,
 "0")
    actual = y_test[n]

    subplot = subplots[i]
    subplot.imshow(x_test[n].reshape((resolution, resolu
tion)), cmap="gray")
    subplot.set_title(f"predicted: {guess}, actual: {act
ual}")
    subplot.axis("off")
plt.show()

# Score of test data
#pred = model.predict(X_test)
from sklearn.metrics import accuracy_score,f1_score,conf
usion_matrix,classification_report
from sklearn import metrics

score = accuracy_score(y_test, y_test_results.round())
print(score*100,'%')

91.62011173184358 %

"""clf_labels=['Logistic Regression','Decision Tree','KN
N','LDAclf']
for clf, label in zip ([logclf,treeclf,knnclf,LDAclf],cl
f_labels):
    scores=cross_val_score(estimator=clf,X=xtrain,y=ytra
in,cv=10,scoring='accuracy')
    print("accuracy: %0.2f (+/-
 %0.2f) [%s]" % (scores.mean(),scores.std(),label))


for clf, label in zip ([logclf,treeclf,knnclf,LDAclf],cl
f_labels):
    scores=cross_val_score(estimator=clf,X=xtrain,y=ytra
in,cv=10,scoring='roc_auc')
```

```
    print("roc auc: %0.2f (+/-
 %0.2f) [%s]" % (scores.mean(),scores.std(),label))"""
clf_labels=[\'Logistic Regression\',\'Decision
Tree\',\'KNN\',\'LDAclf\']\nfor clf, label in zip
([logclf,treeclf,knnclf,LDAclf],clf_labels):\n
scores=cross_val_score(estimator=clf,X=xtrain,y=ytrain,c
v=10,scoring=\'accuracy\')\n print("accuracy: %0.2f (+/-
%0.2f) [%s]" %
(scores.mean(),scores.std(),label))\n\nfor clf, label in
zip ([logclf,treeclf,knnclf,LDAclf],clf_labels):\n
scores=cross_val_score(estimator=clf,X=xtrain,y=ytrain,c
v=10,scoring=\'roc_auc\')\n print("roc auc: %0.2f (+/-
%0.2f) [%s]" % (scores.mean(),scores.std(),label))
```
