BRAIN TUMOR PREDICTION

A PROJECT REPORT

In partial fulfilment of the award of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Under the Guidance of	**Submitted by**
Prof. (Dr.) Manoj Kumar Sachan	**Abhishek Kumar Rai (GCS - 1740592)**
H.O.D (CSE)	**Satish Kaushik (GCS - 1740603)**
	Vikrant Kumar Mall (GCS - 1740578)

SANT LONGOWAL INSTITUTE OF ENGINEERING AND TECHNOLOGY,

LONGOWAL – 148106

DISTRICT- SANGRUR, PUNJAB

June 2021

Table of Contents

i) Certificate

ii) Acknowledgement

iii) Abstract

iv) List of Figures

v) List of Tables

1) Chapter 1. Introduction

2) Chapter 2. Literature Review

3) Chapter 3. Problem Definition

4) Chapter 4. Objectives

5) Chapter 5. Methodology Used

6) Chapter 6. Results and Screenshots

7) Chapter 7. Conclusion and Future Scope References

CERTIFICATE

Certified that this project report “Brain Tumor Prediction” Submitted by “Abhishek Kumar Rai, Vikrant Kumar Mall, Satish Kaushik” in the partial fulfilment of the requirement for the award of Degree in Computer Science & Engineering.

Signature of the students with name

Signature of the Supervisor

Name of the supervisor......................

Designation..

Acknowledgement

If words are considered as a symbol of approval and token of appreciation then let the words play the heralding role expressing my gratitude. The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement. We are grateful to Mr. Jagdeep Singh (A.P, CSE Department), Our project guide Dr. Manoj Kumar Sachan (HOD, CSE Department.), for the guidance, inspiration and constructive suggestion that helped us in the preparation of this project .

ABSTRACT

Automated defect detection in medical is becoming an emergent field in several medical diagnostic applications. Automated detection of tumor in MRI scan data can be very crucial as it provides information about abnormal tissues which is necessary for planning treatment. The conventional method for defect detection in magnetic resonance brain images is human inspection. This method is impractical due to large amount of data. Hence, trusted and automatic classification schemes are essential to prevent the death rate of human due to brain tumor. So, automated tumor detection methods are developed as it would save radiologist time and can obtain a tested accuracy. The MRI brain tumor detection is complicated task due to complexity and variance of tumors. In this project, we propose the machine learning algorithms to overcome the drawbacks of traditional classifiers where tumor is detected in brain MRI using machine learning algorithms. Machine learning and deep learning can be used to efficiently detect cancer cells in brain through MRI scan data.

List of Figures

S. no.	Name
1.	A diagram showing the sub-classes of Artificial Intelligence
2.	A perceptron model of Neural Network
3.	A diagram showing the Transfer Learning
4.	A diagram showing the Iterations in Gradient Boost
5.	Heatmap of Data
6.	Confusion Matrix Gradient Boosting
7.	Model Summary
8.	Cross validation metrics
9.	Scoring metrics
10.	Maximum metrics
11.	Variable Dependencies
12.	Loss and Accuracy Vs Epoch plots of ANN model
13.	Confusion Matrix for ANN Model
14.	Confusion Matrix for Logistic Regression Model
15.	Confusion Matrix for Decision Tree Model
16.	Confusion Matrix for Random Forest Model
17.	Precision Recall graph for Logistic Regression
18.	Precision Recall graph for Decision Tree
19.	Precision Recall graph for Random Forest

List of Tables

S. no.	Name
1.	Metrics Summary

INTRODUCTION

Brain tumor is one of the most rigorous diseases in the medical science. An effective and efficient analysis is always a key concern for the radiologist in the premature phase of tumor growth. Histological grading, based on a stereotactic biopsy test, is the gold standard and the convention for detecting the grade of a brain tumor. The biopsy procedure requires the neurosurgeon to drill a small hole into the skull from which the tissue is collected. There are many risk factors involving the biopsy test, including bleeding from the tumor and brain causing infection, seizures, severe migraine, stroke, coma and even death. But the main concern with the stereotactic biopsy is that it is not 100% accurate which may result in a serious diagnostic error followed by a wrong clinical management of the disease.

Tumor biopsy being challenging for brain tumor patients, non-invasive imaging techniques like Magnetic Resonance Imaging (MRI) have been extensively employed in diagnosing brain tumors. Therefore, development of systems for the detection and prediction of the grade of tumors based on MRI data has become necessary. But at first sight of the imaging modality like in Magnetic Resonance Imaging (MRI), the proper visualization of the tumor cells and its differentiation with its nearby soft tissues is somewhat difficult task which may be due to the presence of low illumination in imaging modalities or its large presence of data or several complexity and variance of tumors-like unstructured shape, viable size and unpredictable locations of the tumor.

The machine learning based approaches like Deep ConvNets in radiology and other medical science fields plays an important role to diagnose the disease in much simpler way as never done before and hence providing a feasible alternative to surgical biopsy for brain tumors.

In this project, we attempted at detecting and classifying the brain tumor through machine learning and deep learning approaches.

LITERATURE REVIEW

Krizhevsky et al. 2012 achieved state-of-the-art results in image classification based on transfer learning solutions upon training a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, he achieved top-1 and top-5 error rates of 37.5% and 17.0% which was considerably better than the previous state-of-the-art. He also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry. The neural network, which had 60 million parameters and 650,000 neurons, consisted of five convolutional layers, some of which were followed by max-pooling layers, and three fully-connected layers with a final 1000-way Softmax. To make training faster, he used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers he employed a recently-developed regularization method called ―dropout‖ that proved to be very effective.

Simonyan& Zisserman 2014 they investigated the effect of the convolutional network depth on itsaccuracy in the large-scale image recognition setting. These findings were the basis of their ImageNet Challenge 2014 submission, where their team secured the first and the second places in the localisation and classification tracks respectively. Their main contribution was a thorough evaluation of networks of increasing depth using architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers after training smaller versions of VGG with less weight layers.

Pan & Yang 2010'ssurvey focused on categorizing and reviewing the current progress on transfer learning for classification, regression and clustering problems. In this survey, they discussed the relationship between transfer learning and other related machine learning techniques such as domain adaptation, multitask learning and sample selection bias, as well as co-variate shift. They also explored some potential future issues in transfer learning research.In this survey article, theyreviewed several current trends of transfer learning.

Szegedyet al.2015 proposed a deep convolutional neural network architecture codenamed Inception, which was responsible for setting the new state of the art for classificationand detection in the ImageNet Large-Scale Visual Recognition Challenge 2014(ILSVRC14). The main hallmark of this architecture is the improved utilizationof the computing resources inside the network. This was achieved by a carefullycrafted design that allows for increasing the depth and width of the network whilekeeping the computational budget constant. His results seem to yield solid evidence that approximating the expected optimal sparse structureby readily available dense building blocks is a viable method for improving neural networks forcomputer vision.

He et al., 2015b introduced the ResNet, which utilizes ―skip connections‖ and batch normalization.Hepresented a residual learning framework to ease the trainingof networks that are substantially deeper than those used previously. He explicitly reformulated the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. He provided comprehensive empirical evidence showing that these residualnetworks are easier to optimize, and can gain accuracy fromconsiderably increased depth. On the ImageNet dataset heevaluated residual nets with a depth of up to 152 layers—8×deeper than VGG nets but still having lower complexity. An ensemble of these residual nets achieves 3.57% erroron the ImageNet test set. This result won the 1st place on theILSVRC 2015 classification task. He also presented analysison CIFAR-10 with 100 and 1000 layers.

Ref. [22] reports the accuracy achieved by seven standard classifiers, viz. i)Adaptive NeuroFuzzy Classifier (ANFC), ii) Naive Bayes(NB), iii) Logistic Regression (LR), iv) Multilayer Perceptron(MLP), v) Support Vector Machine (SVM), vi) Classification and Regression Tree (CART), and vii) k-nearest neighbours (k-NN). The accuracy reported in Ref. [17] 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.

WORKING THEORY OF OUR PROJECT

Artificial Intelligence:

Artificial intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems enabling it to even mimic human behaviour. Its applications lie in fields of Computer Vision, Natural Language Processing, Robotics, Speech Recognition, etc. Advantages of using AI are improved customer experience, accelerate speed to market, develop sophisticated products, enable cost optimisation, enhance employee productivity and improve operational efficiency. Machine Learning (ML) is a subset of AI which is programmed to think on its own, perform social interaction, learn new information from the provided data and adapt as well as improve with experience. Although training time via Deep Learning (DL) methods is more than Machine Learning methods, it is compensated by higher accuracy in the former case. Also, DL being automatic, large domain knowledge is not required for obtaining desired results unlike in ML.

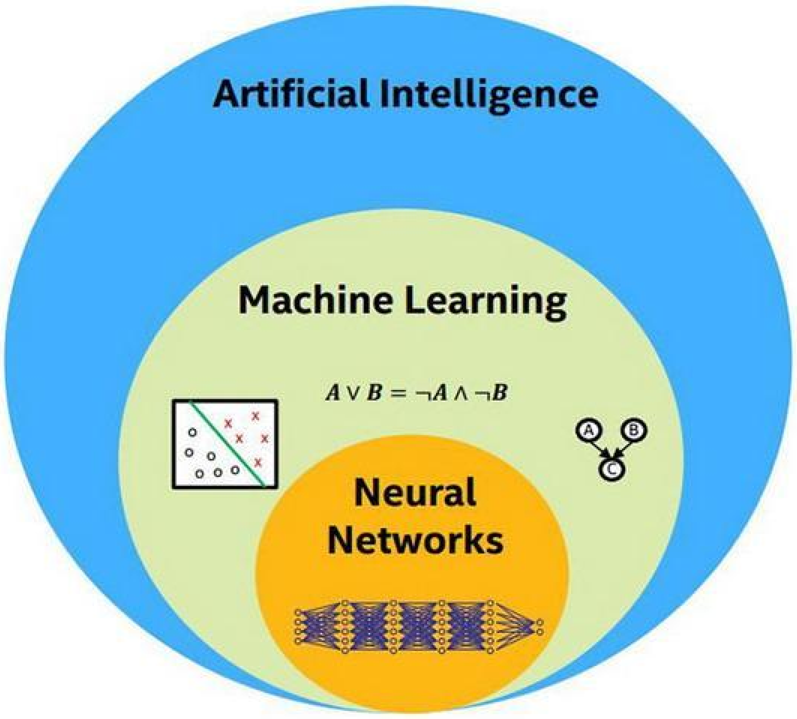


Fig: A diagram showing the sub-classes of Artificial Intelligence

Brain tumor:

In medical science, an anomalous and uncontrollable cell growth inside the brain is recognised as tumor. Human brain is the most receptive part of the body. It controls muscle movements and interpretation of sensory information like sight, sound, touch, taste, pain, etc.

The human brain consists of Grey Matter (GM), White Matter (WM) and Cerebrospinal Fluid (CSF) and on the basis of factors like quantification of tissues, location of abnormalities, malfunctions & pathologies and diagnostic radiology, a presence of tumor is identified. A tumor in the brain can affect such sensory information and muscle movements or even results in more dangerous situation which includes death. Depending upon the place of commencing, tumor can be categorised into primary tumors and secondary tumors. If the tumor is originated inside the skull, then the tumor is known as primary brain tumor otherwise if the tumor‘s initiation place is somewhere else in the body and moved towards the brain, then such tumors are called secondary tumors.

Brain tumor can be of the following types-glioblastoma, sarcoma, metastatic bronchogenic carcinoma on the basis of axial plane. While some tumours such as meningioma can be easily segmented, others like gliomas and glioblastomas are much more difficult to localise. World Health Organisation (WHO) categorised gliomas into - HGG/high grade glioma/glioblastoma/IV stage /malignant & LGG/low grade glioma/II and III stage /benign. Although most of the LGG tumors have slower growth rate compared to HGG and are responsive to treatment, there is a subgroup of LGG tumors which if not diagnosed earlier and left untreated could lead to GBM. In both cases a correct treatment planning (including surgery, radiotherapy, and chemotherapy separately or in combination) becomes necessary, considering that an early and proper detection of the tumor grade can lead to a good prognosis. Survival time for a GBM (Glioblastoma Multiform) or HGG patient is very low i.e. in the range of 12 to 15 months.

Magnetic Resonance Imaging (MRI) has become the standard non-invasive technique for brain tumor diagnosis over the last few decades, due to its improved soft tissue contrast that does not use harmful radiations unlike other methods like CT(Computed Tomography), X-ray, PET (Position Emission Tomography) scans etc. The MRI image is basically a matrix of pixels having characteristic features.

Since glioblastomas are infiltrative tumours, their borders are often fuzzy and hard to distinguish from healthy tissues. As a solution, more than one MRI modality is often employed e.g. T1 (spin-lattice relaxation), T1-contrasted (T1C), T2 (spin-spin relaxation), proton density (PD) contrast imaging, diffusion MRI (dMRI), and fluid attenuation inversion recovery (FLAIR) pulse sequences. T1-weighted images with intravenous contrast highlight the most vascular regions of the tumor (T 1C gives much more accuracy than T1.), called _Enhancing tumor‘ (ET), along with the _tumor core' (TC) that does not involve peritumoral edema. T2-weighted (T2W) and T2W-Fluid Attenuation Inversion Recovery (FLAIR) images are used to evaluate the tumor and peritumoral edema together defined as the _whole tumor‘ (WT). Gliomas and glioblastomas are difficult to distinguish in T1, T1c, T2 and PD. They are better identified in FLAIR modalities.

We have attempted to separate the brain tumor into following types-necrosis (1), edema (2), non- enhancing (malignant) (3) and enhancing (benign) (4) tumor. MRI images can be of three types on the basis of position from which they are taken which are Sagittal (side), Coronal (back) and Axial (top). We have used sagittal images in our project.

Process of brain tumor segmentation can be manual selection of ROI, Semi-automatic and fully-automatic. Popular machine learning algorithms for classification of brain tumor are Artificial Neural Network, Convolutional Neural Network, k-Nearest Neighbour (kNN), Decision Tree, Support Vector Machine (SVM), Naïve Bayes and Random Field (RF). Here, we are using Convolutional Neural Network (CNN) for the detection and classification of the brain tumor.

Basic Operation of Neural Networks:

Neural Networks (NN) form the base of deep learning, a subfield of machine learning where the algorithms are inspired by the structure of the human brain. NN take in data, train themselves to recognize the patterns in this data and then predict the outputs for a new set of similar data. NN are made up of layers of neurons. These neurons are the core processing units of the network. First we have the input layer which receives the input; the output layer predicts our final output. In between, exist the hidden layers which perform most of the computations required by our network.

Our brain tumor images are composed of 128 by 128 pixels which make up for 16,384 pixels. Each pixel is fed as input to each neuron of the first layer. Neurons of one layer are connected to neurons of the next layer through channels .Each of these channels is assigned a numerical value known as _weight‘. The inputs are multiplied to the corresponding weight and their sum is sent as input to the neurons in the hidden layer. Each of these neurons is associated with a numerical value called the _bias‘ which is then added to the input sum. This value is then passed through a threshold function called the _activation function‘. The result of the activation function determines if the particular neuron will get activated or not. An activated neuron transmits data to the neurons of the next layer over the channels. In this manner the data is propagated through the network this is called _forward propagation‘. In the output layer the neuron with the highest value fires and determines the output. The values are basically a probable. The predicted output is compared against the actual output to realize the _error‘ in prediction. The magnitude of the error gives an indication of the direction and magnitude of change to reduce the error. This information is then transferred backward through our network. This is known as _back propagation‘. Now based on this information the weights are adjusted. This cycle of forward propagation and back propagation is iteratively performed with multiple inputs. This process continues until our weights are assigned such that the network can predict the type of tumor correctly in most of the cases. This brings our training process to an end. NN may take hours or even months to train but time is a reasonable trade-off when compared to its scope Several experiments show that after pre-processing MRI images, neural network classification algorithm was the best more specifically CNN(Convolutional Neural Network) as compared to Support Vector Machine(SVM),Random Forest Field.

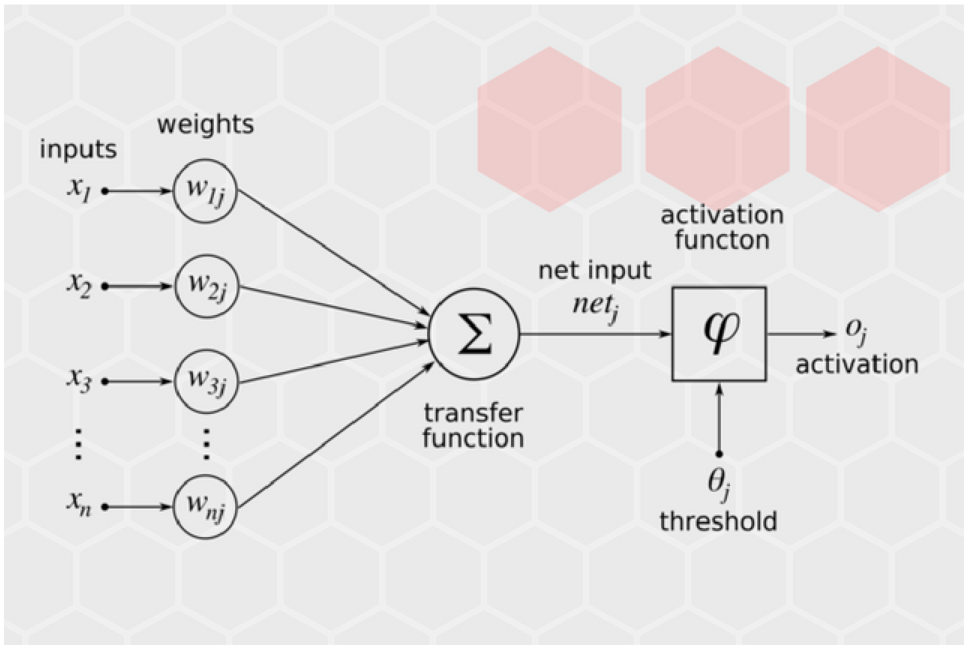


Fig: A perceptron model of neural network

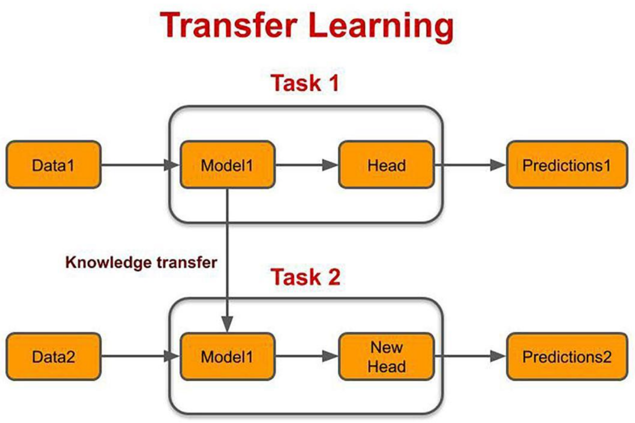
Transfer Learning:

A major assumption in many machine learning and data mining algorithms is that the training and future data must be in the same feature space and have the same distribution. However, in many real-world applications, this assumption may not hold. For example, we sometimes have a classification task in one domain of interest, but we only have sufficient training data in another domain of interest, where the latter data may be in a different feature space or follow a different data distribution. In such cases, knowledge transfer, if done successfully, would greatly improve the performance of learning by avoiding much expensive data labelling efforts. In recent years, transfer learning has emerged as a new learning framework to address this problem.

Transfer learning allows neural networks using significantly less data .With transfer learning, we are in effect transferring the _knowledge‘ that a model has learned from a previous task, to our current one. The idea is that the two tasks are not totally disjoint, as such we can leverage whatever network parameters that model has learned through its extensive training, without having to do that training ourselves. Transfer learning has been consistently proven to boost model accuracy and reduce required training time, less data, less time, more accuracy.

Transfer learning is classified to three different settings: inductive transfer learning, transudative transfer learning and unsupervised transfer learning. Most previous works focused on the settings. Furthermore, each of the approaches to transfer learning can be classified into four contexts based on ―what to transfer‖ in learning. They include the instance-transfer approach, the feature-representation-transfer approach, the parameter transfer approach and the relational knowledge-transfer approach, respectively.

The smaller networks converged & were then used as initializations for the larger, deeper networks- This process is called pre-training. While making logical sense, pre-training is a very time consuming, tedious task, requiring an entire network to be trained before it can serve as an initialization for a deeper network.



Activation Function:

Sigmoid function ranges from 0 to 1 and is used to predict probability as an output in case of binary classification while Softmax function is used for multi-class classification. tanh function ranges from -1 to 1 and is considered better than sigmoid in binary classification using feed forward algorithm. ReLU (Rectified Linear Unit) ranges from 0 to infinity and Leaky ReLU (better version of ReLU) ranges- from -infinity to +infinity. ReLU stands for Rectified Linear Unit for a non-linear operation. The output is $f(x) = \max(0,x)$.ReLU‗s purpose is to introduce non-linearity in our ConvNet. Since, the real world data would want our ConvNet to learn would be non-negative linear values. There are other nonlinear functions such as tanh or sigmoid that can also be used instead of ReLU. Most of the data scientists use ReLU since performance wise ReLU is better than the other two.

Stride is the number of pixels that would move over the input matrix one at a time.

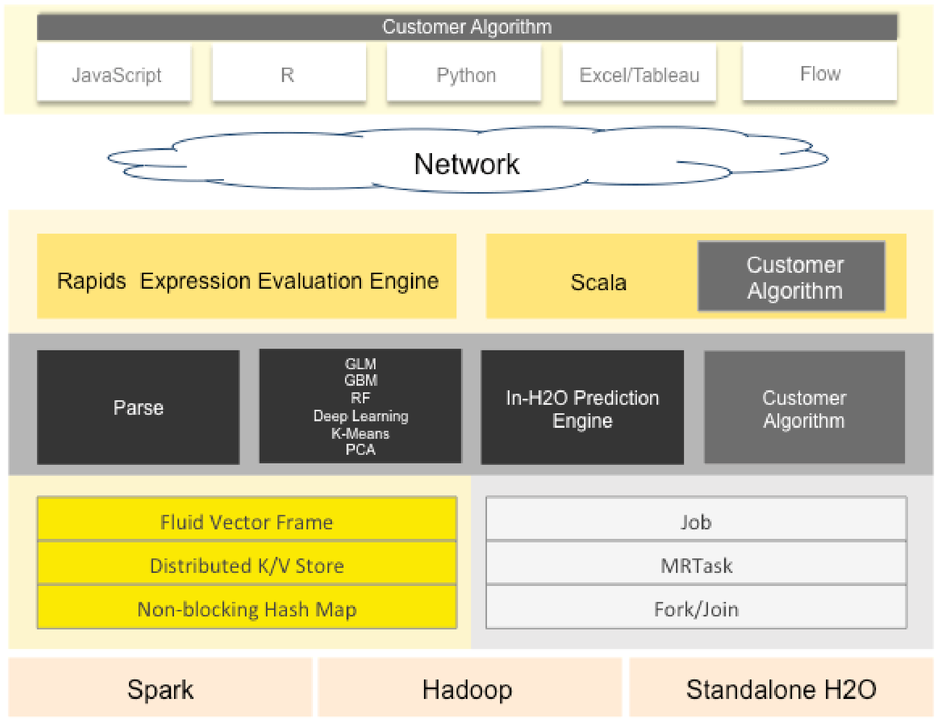
Sometimes filter does not fit perfectly fit the input image. We have two options: either pad the picture with zeros (zero-padding) so that it fits or drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

AutoML: Automatic Machine Learning

In recent years, the demand for machine learning experts has outpaced the supply, despite the surge of people entering the field. To address this gap, there have been big strides in the development of user-friendly machine learning software that can be used by non-experts. The first steps toward simplifying machine learning involved developing simple, unified interfaces to a variety of machine learning algorithms (e.g. H2O).

Although H2O has made it easy for non-experts to experiment with machine learning, there is still a fair bit of knowledge and background in data science that is required to produce high-performing machine learning models. Deep Neural Networks in particular are notoriously difficult for a non-expert to tune properly. In order for machine learning software to truly be accessible to non-experts, we have designed an easy-to-use interface which automates the process of training a large selection of candidate models. H2O's AutoML can also be a helpful tool for the advanced user, by providing a simple wrapper function that performs a large number of modeling-related tasks that would typically require many lines of code, and by freeing up their time to focus on other aspects of the data science pipeline tasks such as data-preprocessing, feature engineering and model deployment.

H2O's AutoML can be used for automating the machine learning workflow, which includes automatic training and tuning of many models within a user-specified time-limit. Stacked Ensembles – one based on all previously trained models, another one on the best model of each family – will be automatically trained on collections of individual models to produce highly predictive ensemble models which, in most cases, will be the top performing models in the AutoML Leaderboard.



H2O offers a number of model explainability methods that apply to AutoML objects (groups of models), as well as individual models (e.g. leader model). Explanations can be generated automatically with a single function call, providing a simple interface to exploring and explaining the AutoML models.

GUI(Graphical user Interface)

A GUI (graphical user interface) is a system of interactive visual components for computer software. A GUI displays objects that convey information, and represent actions that can be taken by the user. The objects change color, size, or visibility when the user interacts with them.

We are using streamlit for this project.

Streamlit

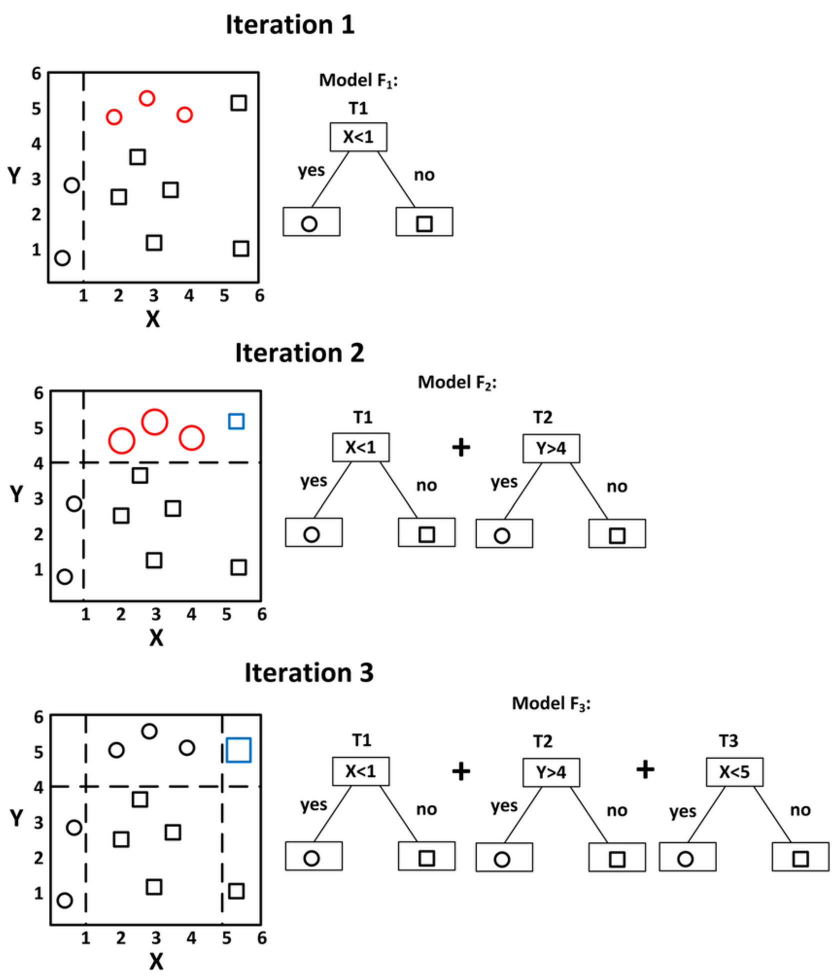
Streamlit is an open-source Python library that can build a UI for various purposes, it is not limited to data apps/machine learning. It is easy to learn, and a few lines of code can create a beautiful web app.

Streamlit is a Python library that helps us develop UIs for our models without HTML/CSS/JS. Most models die inside a Jupyter notebook and are not appealing. But, using Streamlit, you can create a clean UI for your model and showcase it to others. Building a UI lets users use your model in a more user-friendly format.

Gradient Boosting Algorithm

Let's start by understanding Boosting! Boosting is a method of converting weak learners into strong learners. In boosting, each new tree is a fit on a modified version of the original data set. The gradient boosting algorithm (gbm) can be most easily explained by first introducing the AdaBoost Algorithm.The AdaBoost Algorithm begins by training a decision tree in which each observation is assigned an equal weight. After evaluating the first tree, we increase the weights of those observations that are difficult to classify and lower the weights for those that are easy to classify. The second tree is therefore grown on this weighted data. Here, the idea is to improve upon the predictions of the first tree. Our new model is therefore *Tree 1 + Tree 2*. We then compute the classification error from this new 2-tree ensemble model and grow a third tree to predict the revised residuals. We repeat this process for a specified number of iterations. Subsequent trees help us to classify observations that are not well classified by the previous trees. Predictions of the final ensemble model is therefore the weighted sum of the predictions made by the previous tree models.

Gradient Boosting trains many models in a gradual, additive and sequential manner. The major difference between AdaBoost and Gradient Boosting Algorithm is how the two algorithms identify the shortcomings of weak learners (eg. decision trees). While the AdaBoost model identifies the shortcomings by using high weight data points, gradient boosting performs the same by using gradients in the loss function (*y=ax+b+e , e needs a special mention as it is the error term)*. The loss function is a measure indicating how good are model's coefficients are at fitting the underlying data. A logical understanding of loss function would depend on what we are trying to optimise. For example, if we are trying to predict the sales prices by using a regression, then the loss function would be based off the error between true and predicted house prices. Similarly, if our goal is to classify credit defaults, then the loss function would be a measure of how good our predictive model is at classifying bad loans. One of the biggest motivations of using gradient boosting is that it allows one to optimise a user specified cost function, instead of a loss function that usually offers less control and does not essentially correspond with real world applications.



Advantages and Disadvantages of Gradient Boost

Advantages of Gradient Boosting are:

- Often provides predictive accuracy that cannot be trumped.
- Lots of flexibility - can optimize on different loss functions and provides several hyper parameter tuning options that make the function fit very flexible.
- No data pre-processing required - often works great with categorical and numerical values as is.
- Handles missing data - imputation not required.

Disadvantages of Gradient Boosting are:

- Gradient Boosting Models will continue improving to minimize all errors. This can overemphasize outliers and cause overfitting.

- Computationally expensive - often require many trees (>1000) which can be time and memory exhaustive.
- The high flexibility results in many parameters that interact and influence heavily the behavior of the approach (number of iterations, tree depth, regularization parameters, etc.). This requires a large grid search during tuning.
- Less interpretative in nature, although this is easily addressed with various tools.

Solution: Gradient Boosting has repeatedly proven to be one of the most powerful technique to build predictive models in both classification and regression. Because of the fact that Grading Boosting algorithms can easily overfit on a training data set, different constraints or regularization methods can be utilized to enhance the algorithm's performance and combat overfitting. Penalized learning, tree constraints, randomized sampling, and shrinkage can be utilized to combat overfitting.

IMPLEMENTATION METHODOLOGY:

Software Requirements:

Python 3 - We have used Python which is a statistical mathematical programming language like R instead of MATLAB due to the following reasons:

1. Python code is more compact and readable than MATLAB
2. The python data structure is superior to MATLAB
3. It is an open source and also provides more graphic packages and data sets

Keras (with TensorFlow backend 2.3.0 version) - Keras is a neural network API consisting of TensorFlow, CNTk, Theano etc.

Python packages like Numpy, Matplotlib, Pandas for mathematical computation and plotting graphs, SimpleITK for reading the images which were in .mha format and Mahotas for feature extraction of GLCM

Kaggle was used to obtain the online dataset.

Google Colaboratory (open-source Jupyter Notebook interface with high GPU facility) - Google Colab /Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely on cloud. With Colab, one can write and execute code, save and share analyses, access powerful computing resources, all for free from browser.[Jupyter Notebook is a powerful way to iterate and write on your Python code for data analysis. Rather than writing and rewriting an entire code, one can write lines of code and run them at a time. It is built off of iPython which is an interactive way of running Python code. It allows Jupyter notebook to support multiple languages as well as storing the code and writing own markdown.

Hardware Requirements:

Processor: Intel® Core™ i3-2350M CPU @ 2.30GHz

Installed memory (RAM):4.00GB System Type:

64-bit Operating System

Data Acquisition:

Kaggle dataset:

Images can be in the form of .csv (comma separated values), .dat (data) files in grayscale, RGB, or HSV or simply in .zip file as was in the case of our online Kaggle dataset. It contained data of 569 people having 31 features.

Machine Learning Training and Testing:

In supervised network, the network learns by comparing the network output with the correct answer. The network receives feedback about the errors by matching the corresponding labels and weights in different layers and adjusts its weights to minimise the error. It is also known as learning through teacher or _Reinforced Learning‘.

In unsupervised network, there is no teacher i.e. labels are not provided along with the data to the network. Thus, the network does not get any feedback about the errors. The network itself discovers the interesting categories or features in the input data. In many situations, the learning goal is not known in terms of correct answers. The only available information is in the correlation of input data or signals. The unsupervised networks are expected to recognise the input patterns, classify these on the basis of correlations and produce output signals corresponding to input categories. It is a type of dynamic programming that trains algorithm using a system of reward and punishment. Agent learns without human interaction and examples and only by interacting with the environment. For our purpose, we have used supervised network or Reinforced Learning for training our model.

PYTHON PROGRAM FOR THE PROPOSED PROJECT

Import necessary Python packages:

```
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split

import tensorflow as tf

import tensorflow

from tensorflow import keras

from keras.models import Sequential

from keras.layers import Dense

from sklearn.metrics import precision_recall_curve

from sklearn.metrics import f1_score

from sklearn.metrics import auc
```

```
import seaborn as sns

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from jupyterthemes import jtplot

from sklearn import preprocessing

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.metrics import confusion_matrix

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import mean_squared_error, accuracy_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier
```

Data Augmentation:

```
data=pd.read_csv("/content/drive/MyDrive/Data/BrainTumor/BrainTumorData.csv")

data=data.fillna(method='ffill')

X=data[['radius_mean', 'texture_mean', 'perimeter_mean',

    'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',

    'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',

    'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',

    'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',

    'fractal_dimension_se', 'radius_worst', 'texture_worst',

    'perimeter_worst', 'area_worst', 'smoothness_worst',
    'compactness_worst', 'concavity_worst', 'concave points_worst',
    'symmetry_worst', 'fractal_dimension_worst']].values
y=data['diagnosis'].values
```

Data Preprocessing:

```
sou_cha = preprocessing.LabelEncoder()
y = sou_cha.fit_transform(y)
scaler = StandardScaler()
X_scale = scaler.fit_transform(X)
X_train, X_val_and_test, y_train, y_val_and_test = train_test_split(X_scale,y,test_size=0.2)
X_val, X_test, y_val, y_test = train_test_split(X_val_and_test, y_val_and_test, test_size=0.5)
```

Model Construction

Deep learning

```
model = Sequential([
   Dense(250, activation='relu', input_shape=(30,)),
   Dense(100, activation='relu'),
   Dense(10, activation='relu'),
   Dense(1, activation='sigmoid')])
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])
hist = model.fit(X_train, y_train,
          batch_size=32, epochs=2,
          validation_data=(X_val, y_val))
```

Machine Learning

1. Logistic Regression

```
lr_model = LogisticRegression()
          lr_model.fit(X_train,y_train)
```

2. Decision Tree Classifier

```
dt_model = DecisionTreeClassifier()
          dt_model.fit(X_train, y_train)
```

3. Random Forest Classifier

```
rf_model = RandomForestClassifier(n_estimators=400, max_depth=10)
          rf_model.fit(X_train, y_train)
```

H2o AutoML:

```
import h2o
h2o.init()
h2o_df = h2o.H2OFrame(X)
h2o_df.describe()
train, test = h2o_df.split_frame(ratios=[.8])
x = train.columns
y1 = 'C31'
```

```
x.remove(y1)

from h2o.automl import H2OAutoML

aml = H2OAutoML(max_runtime_secs = 600,

            # exclude_algos =['DeepLearning'],

            seed = 1,

            # stopping_metric ='logloss',

            # sort_metric ='logloss',

            balance_classes = False,

            project_name ="Prject_1"

)

# train model and record time % time

aml.train(x = x, y = y1, training_frame = train)

lb =aml.leaderboard

lb.head(rows = lb.nrows)

se

model = h2o.get_model('GBM_grid__1_AutoML_20210501_165920_model_18')

model.model_performance(test)

model.varimp_plot(num_of_features=9)

lb = h2o.automl.get_leaderboard(aml, extra_columns = 'ALL')

lb

model_path = h2o.save_model(model = model, path ='/content/drive/MyDrive/Model', force =
True)

saved_model =
h2o.load_model('/content/drive/MyDrive/Model/GBM_grid__1_AutoML_20210501_165920_m
odel_18')

preds = saved_model.predict(test)

preds
```

Evaluation of model performance:

Deep learning

```
print('\033[91m'"Accuracy on training set:{:.3f}".format(model.evaluate(X_train,y_train)[1]))
print('\033[93m'"Accuracy on Validation set:{:.3f}".format(model.evaluate(X_val,y_val)[1]))
print('\033[92m'"Accuracy on test set:{:.3f}".format(model.evaluate(X_test,y_test)[1]))
```

Machine Learning

1. **Logistic Regression**

```
lr_accuracy = lr_model.score(X_test, y_test)
lr_accuracy
```

2. **Decision Tree Classifier**

```
dt_accuracy = dt_model.score(X_test, y_test)
dt_accuracy
```

3. **Random Forest Classifier**

```
rf_accuracy = rf_model.score(X_test, y_test)
rf_accuracy
```

H2o AutoML

```
saved_model
```

Evaluation of Model Prediction

```
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5)

cm = confusion_matrix(y_test, y_pred)

print(cm)

plt.plot(hist.history['loss'])

plt.title('Model loss Progress During Training')

plt.xlabel('Epoch')

plt.ylabel('Training Loss')

plt.legend(['Training Loss'])

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

from math import sqrt

def errors(y_test,y_predict):

  rsme = float(format(np.sqrt(mean_squared_error(y_test,y_predict)),'.3f'))

  mse  = mean_squared_error(y_test, y_predict)

  mae  = mean_absolute_error(y_test, y_predict)

  r2   = r2_score(y_test, y_predict)

  adj_r2 = 1-((1-r2)*(n-1))/(n-k-1)

  print('RSME =',rsme)

  print('MSE =',mse)

  print('MAE =',mae)

  print('R2 =',r2)

  print('ADJ_R2 =',adj_r2)

from sklearn.metrics import roc_curve

from sklearn.metrics import roc_auc_score, auc



y_score  = model.predict_proba(X_test)[:,0]

y_score1 = lr_model.predict_proba(X_test)[:,1]
```

```
y_score2 = dt_model.predict_proba(X_test)[:,1]
y_score3 = rf_model.predict_proba(X_test)[:,1]
fpr,  tpr,  th = roc_curve(y_test, y_score)
fpr1, tpr1, th1 = roc_curve(y_test, y_score1)
fpr2, tpr2, th2 = roc_curve(y_test, y_score2)
fpr3, tpr3, th3 = roc_curve(y_test, y_score3)


print('roc_auc_score for Artificial Neural Network: ', roc_auc_score(y_test, y_score))
print('roc_auc_score for Logistic Regression: ', roc_auc_score(y_test, y_score1))
print('roc_auc_score for DecisionTree: ', roc_auc_score(y_test, y_score2))
print('roc_auc_score for RandomForest: ', roc_auc_score(y_test, y_score3))
def F1Score(y_test, yhat):
  print(f1_score(y_test, yhat))
y_ann_hat = model.predict(X_test)
y_lr_hat = lr_model.predict(X_test)
y_rf_hat = rf_model.predict(X_test)
y_dt_hat = dt_model.predict(X_test)
F1Score(y_test, y_lr_hat)
F1Score(y_test, y_rf_hat)
F1Score(y_test, y_dt_hat)
lr_probs = dt_model.predict_proba(X_test)
lr_probs = lr_probs[:, 1]
yhat = dt_model.predict(X_test)
lr_precision, lr_recall, _ = precision_recall_curve(y_test, lr_probs)
lr_f1, lr_auc = f1_score(y_test, yhat), auc(lr_recall, lr_precision)
print('Logistic: f1=%.3f auc=%.3f' % (lr_f1, lr_auc))
```

```
no_skill = len(y_test[y_test==1]) / len(y_test)

plt.title('Precision Recall - DECISION TREE')

plt.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill')

plt.plot(lr_recall, lr_precision, marker='.', label='LOGISTIC REGRESSION')

plt.xlabel('Recall')

plt.ylabel('Precision')

plt.show()

y_ann_predict = model.predict(X_test)

y_rf_predict = rf_model.predict(X_test)

y_lr_predict = lr_model.predict(X_test)

y_dt_predict = dt_model.predict(X_test)

errors(y_ann_predict,y_test)

errors(y_rf_predict,y_test)

errors(y_lr_predict,y_test)

errors(y_dt_predict,y_test)
```

Code for GUI :

```
import streamlit as st

import pandas as pd

import sklearn

from sklearn import datasets

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier


st.title("Brain Tumor Prediction")

st.write('Brain Tumor can be defined as a cancerous or non-cancerous mass or growth of
abnormal cells in the brain. Every year more 1 million cases of brain tumor is reported in India.
If it is detected in early or primary stage it can be cured easily and cost effectively.')

st.write("""

# Explore different Classifier for best pick

""")

data=pd.read_csv('/home/abhishek/Desktop/Streamlit_App_Resources/BrainTumorData.csv')

data=data.replace({"M":1,"B":0})

st.image("/home/abhishek/Downloads/St.jpeg")
```

```
if st.checkbox('Show Dataset'):
    st.table(data)
classifier_name=st.sidebar.selectbox("Select Classifier",("KNN","SVM","Logistic
Regression","Random Forest","Decision Tree"))


X=data.iloc[:,:]
st.write("Shape of Dataset including all features", X.shape)
X1=data.iloc[:,2:]
y1=data.iloc[:,1]
st.write("Shape of Dataset including only useful features", X1.shape)


def add_parameters_ui(clf_name):
    params=dict()
  if clf_name=="KNN":
        if st.checkbox('Show Precision Recall graph'):
            st.image('/home/abhishek/Desktop/Streamlit_App_Resources/KNN.jpg')
        n_neighbors=st.sidebar.slider("n_neighbors",1,15)
        p=st.sidebar.slider("p",1,10)
        leaf_size=st.sidebar.slider("leaf_size",10,100)
        params["p"]=p
        params["leaf_size"]=leaf_size
        params["n_neighbors"]=n_neighbors
    elif clf_name=="SVM":
        if st.checkbox('Show Precision Recall graph'):
            st.image('/home/abhishek/Desktop/Streamlit_App_Resources/SVM.jpg')
        C=st.sidebar.slider("C",0.01,10.0)
        params["C"]=C
    elif clf_name=="Logistic Regression":
        if st.checkbox('Show Precision Recall graph'):
            st.image('/home/abhishek/Desktop/Streamlit_App_Resources/Logistic_Regression.jpg')
        C=st.sidebar.slider("C",1.0,10.0)
        max_iter=st.sidebar.slider("max_iter",1,200)
        params["C"]=C
        params["max_iter"]=max_iter
    elif clf_name=="Decision Tree":
        if st.checkbox('Show Precision Recall graph'):
            st.image('/home/abhishek/Desktop/Streamlit_App_Resources/Decision_Tree.png')
        min_samples_split=st.sidebar.slider("min_samples_split",2,40)
        min_samples_leaf=st.sidebar.slider("min_samples_leaf",1,20)
        params["min_samples_split"]=min_samples_split
        params["min_samples_leaf"]=min_samples_leaf
    else:
        if st.checkbox('Show Precision Recall graph'):
            st.image('/home/abhishek/Desktop/Streamlit_App_Resources/Random_Forest.png')
        max_depth=st.sidebar.slider("max_depth",2,15)
        n_estimators=st.sidebar.slider("n_estimators",1,100)
        params["max_depth"]=max_depth
        params["n_estimators"]=n_estimators
    return params

params=add_parameters_ui(classifier_name)

def get_classifier(clf_name, params):
    if clf_name=="KNN":
        clf=KNeighborsClassifier(n_neighbors=params["n_neighbors"])
    elif clf_name=="SVM":
        clf=SVC(C=params["C"])
    elif clf_name=="Logistic Regression":
        clf=LogisticRegression(C=params["C"],max_iter=params["max_iter"])
    elif clf_name=="Decision Tree":

clf=DecisionTreeClassifier(min_samples_split=params["min_samples_split"],min_samples_leaf
=params["min_samples_leaf"])
    else:

clf=RandomForestClassifier(n_estimators=params["n_estimators"],max_depth=params["max_de
pth"], random_state=1234)
    return clf
clf=get_classifier(classifier_name,params)


# Classification
X1_train,X1_test,y1_train,y1_test=train_test_split(X1,y1,test_size=0.2, random_state=1234)


clf.fit(X1_train,y1_train)
y1_pred=clf.predict(X1_test)


acc=accuracy_score(y1_test,y1_pred)
st.write(f"classifier={classifier_name}")
st.write(f"accuracy={acc}")


# Plot
pca = PCA(2)
```

```
X1_projected=pca.fit_transform(X1)



a1=X1_projected[:,0]

a2=X1_projected[:,1]


fig=plt.figure()

lt.scatter(a1,a2,c=y1,alpha=0.8,cmap="viridis")

plt.xlabel("Principle Component 1")

plt.ylabel("Principle Component 2")

plt.colorbar()


st.pyplot(fig)
```

EVALUATION OF THE PREDICTIVE MODEL PERFORMANCE

Data

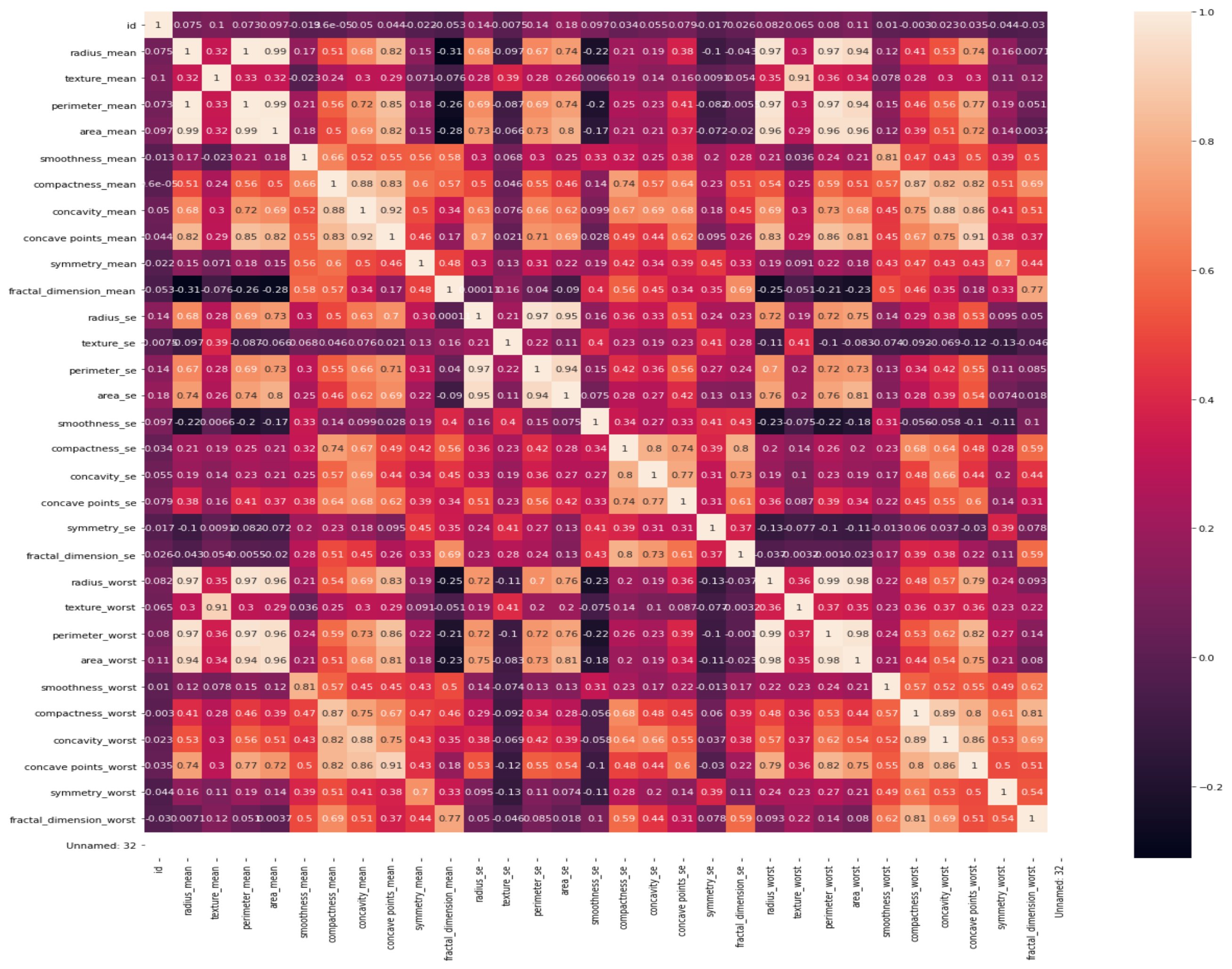
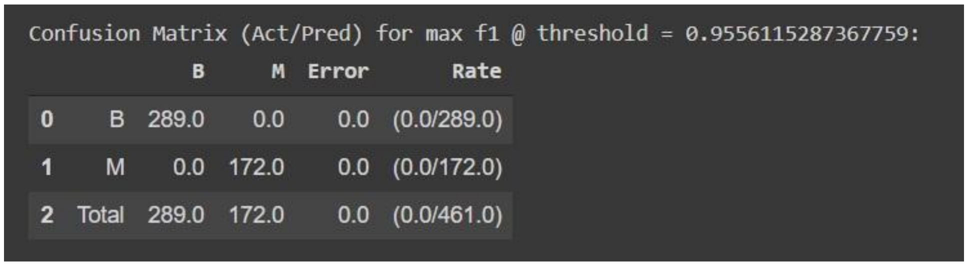


Fig: Heatmap of the data

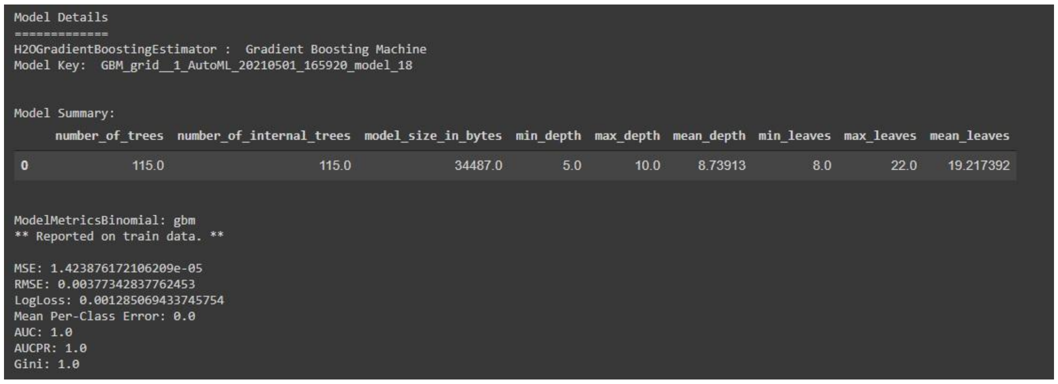
H2o AutoML



Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.9556115287367759:

		B	M	Error	Rate
0	B	289.0	0.0	0.0	(0.0/289.0)
1	M	0.0	172.0	0.0	(0.0/172.0)
2	Total	289.0	172.0	0.0	(0.0/461.0)

Fig: Confusion matrix Gradient boosting

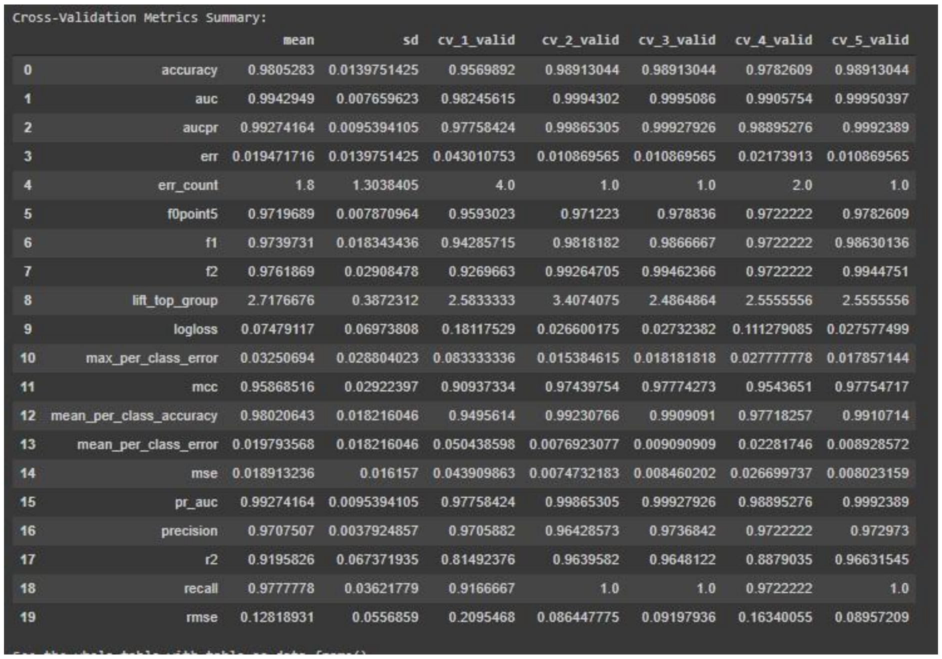


```
Model Details
=============
H2OGradientBoostingEstimator :  Gradient Boosting Machine
Model Key:  GBM_grid__1_AutoML_20210501_165920_model_18


Model Summary:
```

	number_of_trees	number_of_internal_trees	model_size_in_bytes	min_depth	max_depth	mean_depth	min_leaves	max_leaves	mean_leaves
0	115.0	115.0	34487.0	5.0	10.0	8.73913	8.0	22.0	19.217392

```
ModelMetricsBinomial: gbm
** Reported on train data. **

MSE: 1.423876172106209e-05
RMSE: 0.00377342837762453
LogLoss: 0.001285069433745754
Mean Per-Class Error: 0.0
AUC: 1.0
AUCPR: 1.0
Gini: 1.0
```

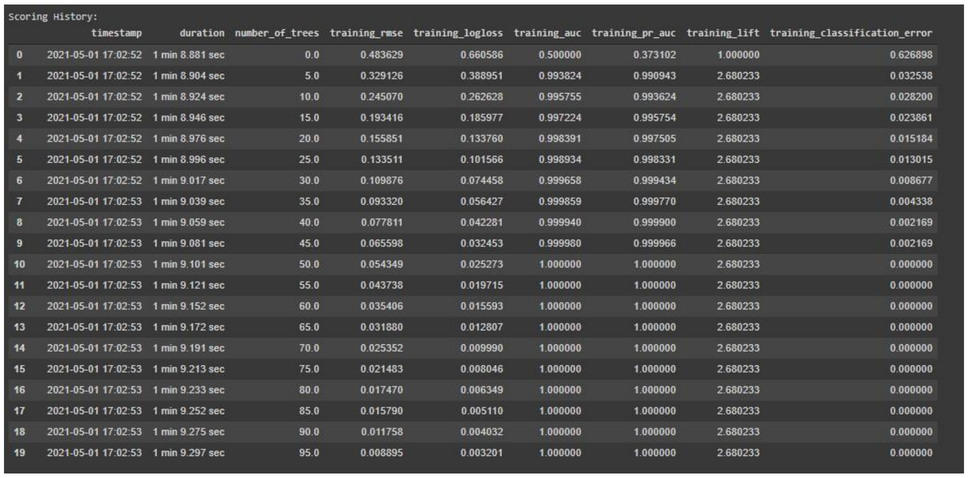
Fig: Model Summary



Cross-Validation Metrics Summary:

		mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid
0	accuracy	0.9805283	0.0139751425	0.9569892	0.98913044	0.98913044	0.9782609	0.98913044
1	auc	0.9942949	0.007659623	0.98245615	0.9994302	0.9995086	0.9905754	0.99950397
2	aucpr	0.99274164	0.0095394105	0.97758424	0.99865305	0.99927926	0.98895276	0.9992389
3	err	0.019471716	0.0139751425	0.043010753	0.010869565	0.010869565	0.02173913	0.010869565
4	err_count	1.8	1.3038405	4.0	1.0	1.0	2.0	1.0
5	f0point5	0.9719689	0.007870964	0.9593023	0.971223	0.978836	0.9722222	0.9782609
6	f1	0.9739731	0.018343436	0.94285715	0.9818182	0.9866667	0.9722222	0.98630136
7	f2	0.9761869	0.02908478	0.9269663	0.99264705	0.99462366	0.9722222	0.9944751
8	lift_top_group	2.7176676	0.3872312	2.5833333	3.4074075	2.4864864	2.5555556	2.5555556
9	logloss	0.07479117	0.06973808	0.18117529	0.026600175	0.02732382	0.111279085	0.027577499
10	max_per_class_error	0.03250694	0.028804023	0.083333336	0.015384615	0.018181818	0.027777778	0.017857144
11	mcc	0.95868516	0.02922397	0.90937334	0.97439754	0.97774273	0.9543651	0.97754717
12	mean_per_class_accuracy	0.98020643	0.018216046	0.9495614	0.99230766	0.9909091	0.97718257	0.9910714
13	mean_per_class_error	0.019793568	0.018216046	0.050438598	0.0076923077	0.009090909	0.02281746	0.008928572
14	mse	0.018913236	0.016157	0.043909863	0.0074732183	0.008460202	0.026699737	0.008023159
15	pr_auc	0.99274164	0.0095394105	0.97758424	0.99865305	0.99927926	0.98895276	0.9992389
16	precision	0.9707507	0.0037924857	0.9705882	0.96428573	0.9736842	0.9722222	0.972973
17	r2	0.9195826	0.067371935	0.81492376	0.9639582	0.9648122	0.8879035	0.96631545
18	recall	0.9777778	0.03621779	0.9166667	1.0	1.0	0.9722222	1.0
19	rmse	0.12818931	0.0556859	0.2095468	0.086447775	0.09197936	0.16340055	0.08957209

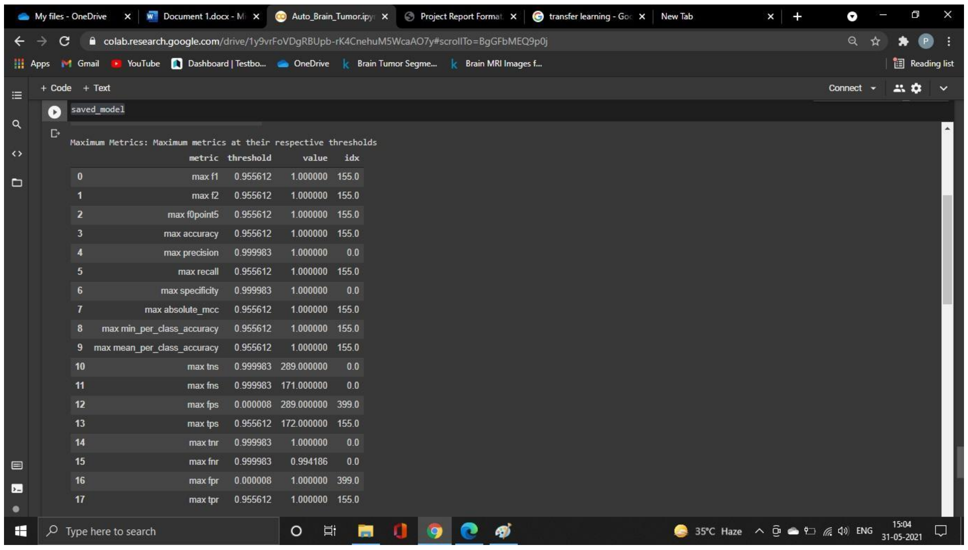
Fig: Cross Validation metrics



Scoring History:

	timestamp	duration	number_of_trees	training_rmse	training_logloss	training_auc	training_pr_auc	training_lift	training_classification_error
0	2021-05-01 17:02:52	1 min 8.881 sec	0.0	0.483629	0.660586	0.500000	0.373102	1.000000	0.626898
1	2021-05-01 17:02:52	1 min 8.904 sec	5.0	0.329126	0.388951	0.993824	0.990943	2.680233	0.032538
2	2021-05-01 17:02:52	1 min 8.924 sec	10.0	0.245070	0.262628	0.995755	0.993624	2.680233	0.028200
3	2021-05-01 17:02:52	1 min 8.946 sec	15.0	0.193416	0.185977	0.997224	0.995754	2.680233	0.023861
4	2021-05-01 17:02:52	1 min 8.976 sec	20.0	0.155851	0.133760	0.998391	0.997505	2.680233	0.015184
5	2021-05-01 17:02:52	1 min 8.996 sec	25.0	0.133511	0.101566	0.998934	0.998331	2.680233	0.013015
6	2021-05-01 17:02:52	1 min 9.017 sec	30.0	0.109876	0.074458	0.999658	0.999434	2.680233	0.008677
7	2021-05-01 17:02:53	1 min 9.039 sec	35.0	0.093320	0.056427	0.999859	0.999770	2.680233	0.004338
8	2021-05-01 17:02:53	1 min 9.059 sec	40.0	0.077811	0.042281	0.999940	0.999900	2.680233	0.002169
9	2021-05-01 17:02:53	1 min 9.081 sec	45.0	0.065598	0.032453	0.999980	0.999966	2.680233	0.002169
10	2021-05-01 17:02:53	1 min 9.101 sec	50.0	0.054349	0.025273	1.000000	1.000000	2.680233	0.000000
11	2021-05-01 17:02:53	1 min 9.121 sec	55.0	0.043738	0.019715	1.000000	1.000000	2.680233	0.000000
12	2021-05-01 17:02:53	1 min 9.152 sec	60.0	0.035406	0.015593	1.000000	1.000000	2.680233	0.000000
13	2021-05-01 17:02:53	1 min 9.172 sec	65.0	0.031880	0.012807	1.000000	1.000000	2.680233	0.000000
14	2021-05-01 17:02:53	1 min 9.191 sec	70.0	0.025352	0.009990	1.000000	1.000000	2.680233	0.000000
15	2021-05-01 17:02:53	1 min 9.213 sec	75.0	0.021483	0.008046	1.000000	1.000000	2.680233	0.000000
16	2021-05-01 17:02:53	1 min 9.233 sec	80.0	0.017470	0.006349	1.000000	1.000000	2.680233	0.000000
17	2021-05-01 17:02:53	1 min 9.252 sec	85.0	0.015790	0.005110	1.000000	1.000000	2.680233	0.000000
18	2021-05-01 17:02:53	1 min 9.275 sec	90.0	0.011758	0.004032	1.000000	1.000000	2.680233	0.000000
19	2021-05-01 17:02:53	1 min 9.297 sec	95.0	0.008895	0.003201	1.000000	1.000000	2.680233	0.000000

Fig: Scoring metrics

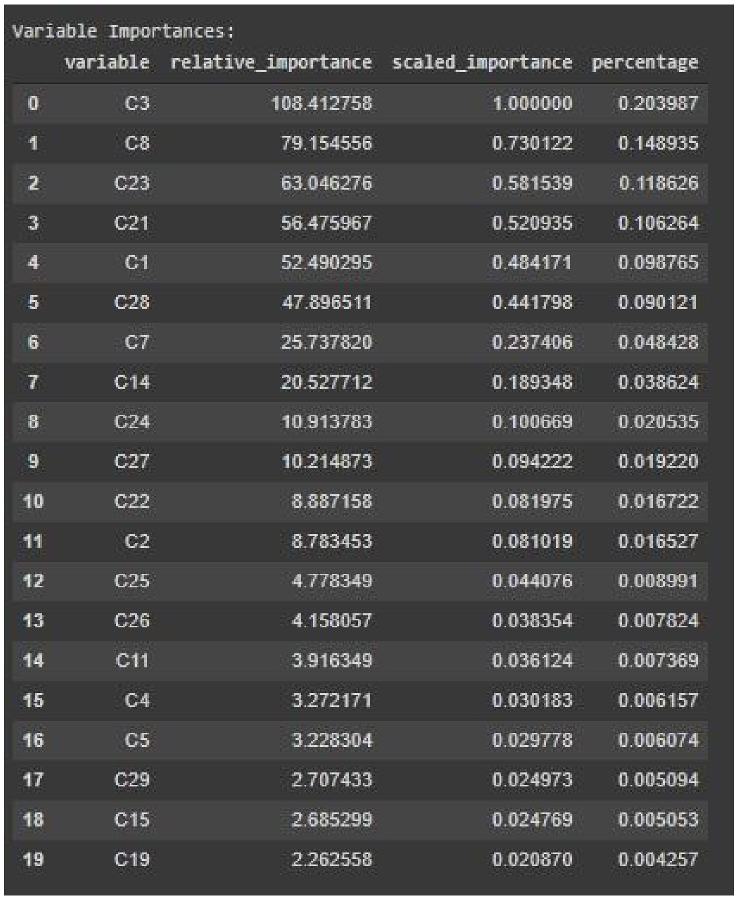


```
saved_model
```

Maximum Metrics: Maximum metrics at their respective thresholds

	metric	threshold	value	idx
0	max f1	0.955612	1.000000	155.0
1	max f2	0.955612	1.000000	155.0
2	max f0point5	0.955612	1.000000	155.0
3	max accuracy	0.955612	1.000000	155.0
4	max precision	0.999983	1.000000	0.0
5	max recall	0.955612	1.000000	155.0
6	max specificity	0.999983	1.000000	0.0
7	max absolute_mcc	0.955612	1.000000	155.0
8	max min_per_class_accuracy	0.955612	1.000000	155.0
9	max mean_per_class_accuracy	0.955612	1.000000	155.0
10	max tns	0.999983	289.000000	0.0
11	max fns	0.999983	171.000000	0.0
12	max fps	0.000008	289.000000	399.0
13	max tps	0.955612	172.000000	155.0
14	max tnr	0.999983	1.000000	0.0
15	max fnr	0.999983	0.994186	0.0
16	max fpr	0.000008	1.000000	399.0
17	max tpr	0.955612	1.000000	155.0

Fig: Maximum metrics

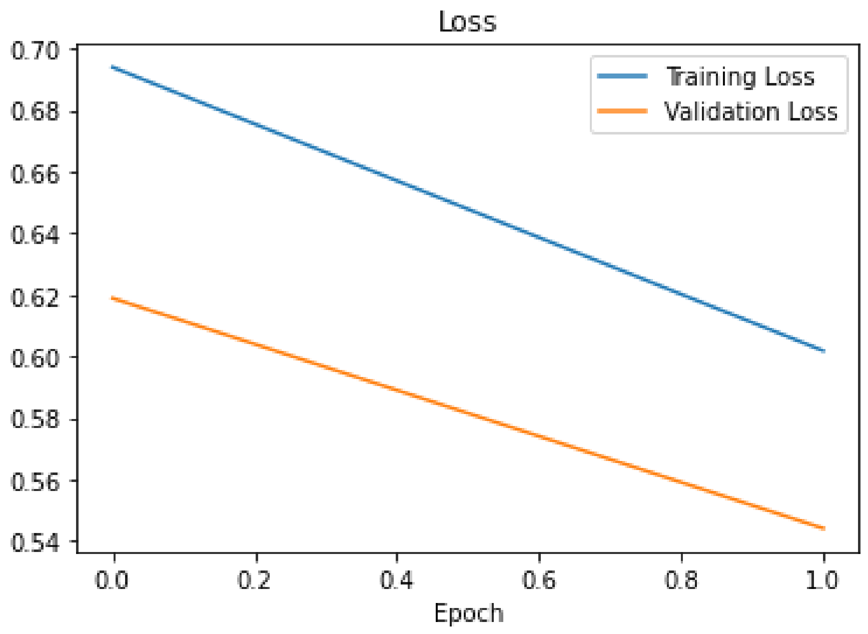


Variable Importances:

	variable	relative_importance	scaled_importance	percentage
0	C3	108.412758	1.000000	0.203987
1	C8	79.154556	0.730122	0.148935
2	C23	63.046276	0.581539	0.118626
3	C21	56.475967	0.520935	0.106264
4	C1	52.490295	0.484171	0.098765
5	C28	47.896511	0.441798	0.090121
6	C7	25.737820	0.237406	0.048428
7	C14	20.527712	0.189348	0.038624
8	C24	10.913783	0.100669	0.020535
9	C27	10.214873	0.094222	0.019220
10	C22	8.887158	0.081975	0.016722
11	C2	8.783453	0.081019	0.016527
12	C25	4.778349	0.044076	0.008991
13	C26	4.158057	0.038354	0.007824
14	C11	3.916349	0.036124	0.007369
15	C4	3.272171	0.030183	0.006157
16	C5	3.228304	0.029778	0.006074
17	C29	2.707433	0.024973	0.005094
18	C15	2.685299	0.024769	0.005053
19	C19	2.262558	0.020870	0.004257

Fig: Variable dependencies

Deep learning model



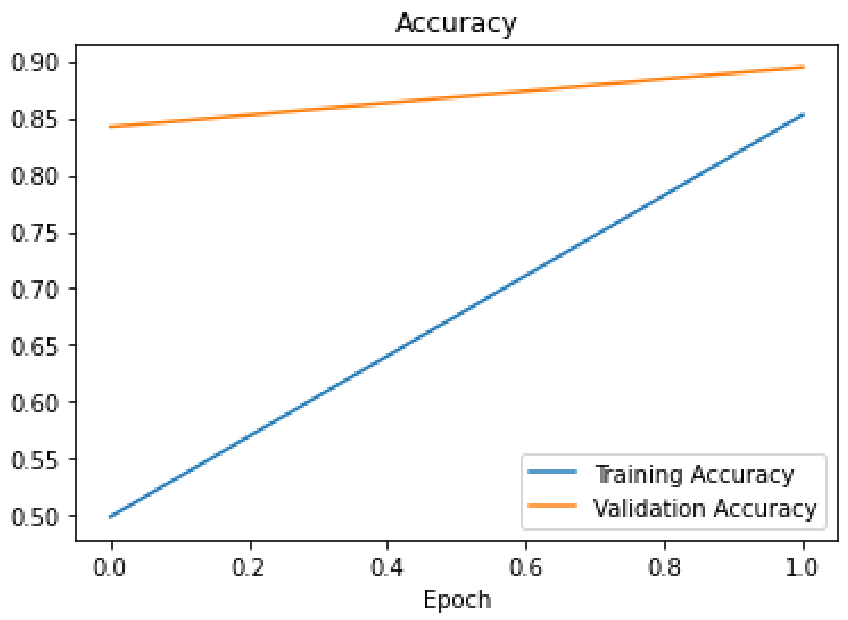
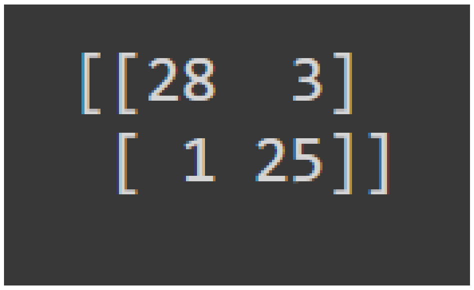


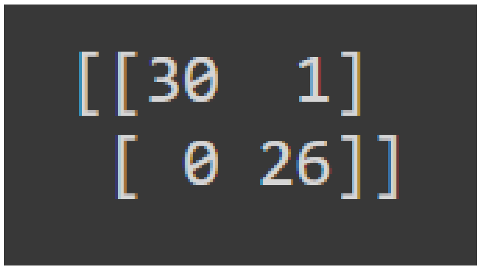
Fig: Loss and Accuracy Vs Epoch plots of a ANN model

CONFUSION MATRIX



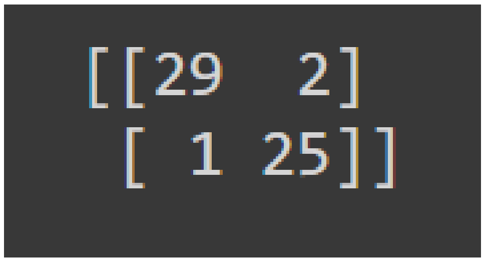
```
[[28  3]
 [ 1 25]]
```

Fig: ANN



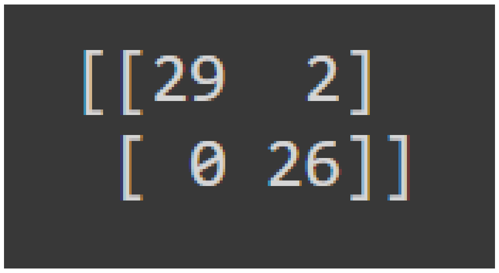
```
[[30  1]
 [ 0 26]]
```

Fig: Logistic Regression



```
[[29  2]
 [ 1 25]]
```

Fig: Decision Tree



```
[[29  2]
 [ 0 26]]
```

Fig: Random Forest

Precision Recall Graph

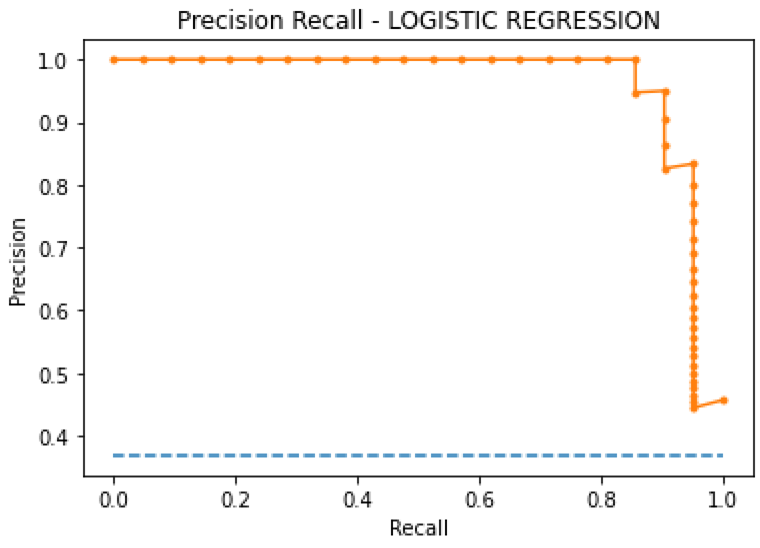


Fig: Logistic Regression

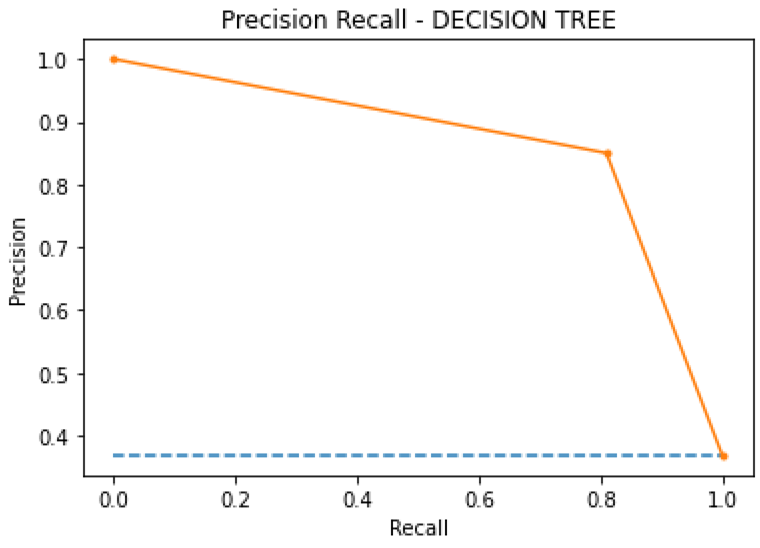


Fig: Decision Tree

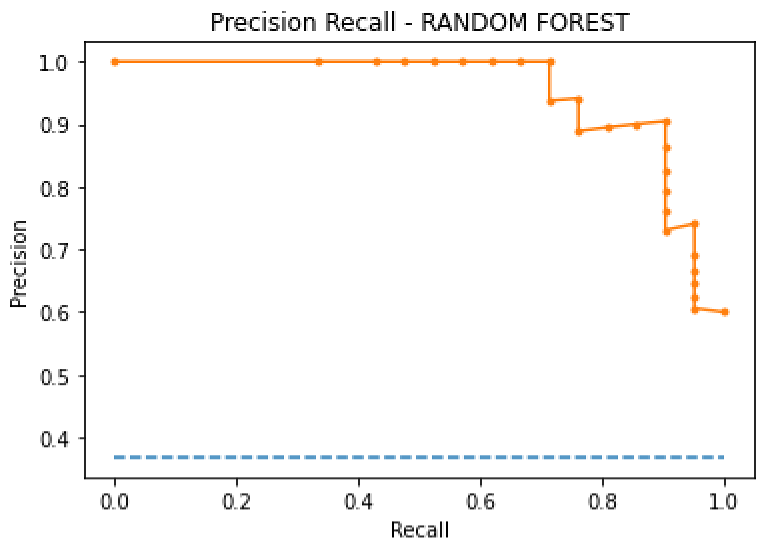


Fig: Random Forest

Metrics Summary

Metrics	ANN	Logistic Regression	Decision Tree	Random Forest	AutoML GBM
Accuracy	93.0%	96.49%	91.22%	94.73%	98.05%
F1 score	0.916	0.95454	0.93023	0.88372	0.9739
RSME	0.441	0.324	0.229	0.35	0.1281
MSE	0.1942	0.1052	0.0526	0.1228	0.0189
MAE	0.4290	0.1052	0.0526	0.1228	0.0194
R2	-46.45	0.5263	0.7564	0.4608	0.9195
Roc_auc_score	0.8955	0.9603	0.8630	0.9636	0.9942

CONCLUSION

Using ANN model, the train accuracy is 91.4% and validation accuracy is 89.5%.The validation result had a best figure of 93.0% as accuracy.It is observed that with using ANN model, although the training accuracy is >90%, the overall accuracy is low unlike where pre-trained model is used.

While using Logistic Regression, Decision Tree and Random Forest our accuracy was 98.24%, 94.7% and 96.4%

Although we get the best accuracy when we use H2o AutoML which uses various machine leaning and deep learning models and get the best accuracy on Gradient Boosting Machine

Model Key: GBM_grid_1_AutoML_20210501_165920_model_18

Our best evaluation metrics scores (i.e F1 score, RSME, MSE, MAE, R2) are from Gradient Boosting Machine.

Hence, we can we use Gradient Boosting for prediction on csv data efficiently.

FUTURE SCOPE

Build an app-based user interface in hospitals which allows doctors to easily determine the impact of tumor and suggest treatment accordingly

Since performance and complexity of ConvNets depend on the input data representation we can try to predict the location as well as stage of the tumor from Volume based 3D images. By creating three dimensional (3D) anatomical models from individual patients, training, planning and computer guidance during surgery is improved.

Using VolumeNet with LOPO (Leave-One-Patient-Out) scheme has proved to give a high training as well as validation accuracy(>95%).In LOPO test scheme, in each iteration, one patient is used for testing and remaining patients are used for training the ConvNets, this iterates for each patient. Although LOPO test scheme is computationally expensive, using this we can have more training data which is required for ConvNets training. LOPO testing is robust and most applicable to our application, where we get test result for each individual patient. So, if classifier misclassifies a patient then we can further investigate it separately.

Improve testing accuracy and computation time by using classifier boosting techniques like using more number images with more data augmentation, fine-tuning hyper parameters, training for a longer time i.e. using more epochs, adding more appropriate layers etc.. Classifier boosting is done by building a model from the training data then creating a second model that attempts to correct the errors from the first model for faster prognosis. Such techniques can be used to raise the accuracy even higher and reach a level that will allow this tool to be a significant asset to any medical facility dealing with brain tumors.

For more complex datasets, we can use U-Net architecture rather than CNN where the max pooling layers are just replaced by upsampling ones.

Ultimately we would like to use very large and deep convolutional nets on video sequences where the temporal structure provides very helpful information that is missing or far less obvious in static images.

Unsupervised transfer learning may attract more and more attention in the future.

REFERENCES

[1] https://keras.io/applications/

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

[3] https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

[4] https://simpleitk.org

[5]https://neurohive.io/en/popular-networks/resnet/

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

[7]http://builtin.com/data-science/transfer-learning

[8]https://openreview.net/forum?id=BJIRs34Fvr

[9]https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6640210

[10] https://arxiv.org/pdf/1409.1556.pdf

[11] https://www.cse.ust.hk/~qyang/Docs/2009/tkde_transfer_learning.pdf

[12] https://arxiv.org/pdf/1409.4842.pdf

[13] https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[14] https://arxiv.org/pdf/1512.03385.pdf

[15] https://coursera.org

[16] https://youtube.com