Image Classification for Brain Tumour Detection using Machine Learning

by
Ong Ge Ye (P21013106),
Tan Soo Hong (P21013177),
Kee Yong Yik (P21013110).

being the project proposal submitted to
INTI International College Penang
as a requirement for
Diploma in Computer Science
Session August 2022

School of Engineering and Information Technology
INTI INTERNATIONAL COLLEGE PENANG

Table of Content

Chapter 1: Introduction

1.1 Problem Background

Brain tumours are caused by the abnormal growth of cells in the brain, affecting how the brain operates by affecting the nervous system, impairing motor and logical thinking functions. These tumours can be benign (non-cancerous) or malignant (cancerous). Benign tumours are less dangerous compared to malignant ones. They can begin in the brain as primary tumours or somewhere else and then migrate into the brain as secondary brain tumours.

In Malaysia, brain cancer is taken seriously despite only making up 1.95% of all malignant cancer diagnoses (Dzali, 2017). The position of the tumour, whether benign or malignant, can cause adverse effects on human behaviour and function, giving it the title of the most fatal type of cancer. According to the Malaysian National Registry Report, brain cancers and cancers of the nervous system are the 11th most common in males and 13th in females around Malaysia.

There are many methods of detecting brain cancer, however the most common methods are through medical imaging analysis, which includes MRI's (Magnetic Resonance Imaging) and CT scans (Computerised Tomography). MRI's uses magnetic fields to form images inside the body, which allows the measurement of the foreign entity if found. CT Scan uses X-Rays to take multiple photos at different angles, which will be combined digitally to form a scanned 3D image of the site.

It is a time consuming and often complicated process to diagnose brain cancer which can involve multiple neurologists and oncologists. A physical exam testing the motor functions will be carried out along with a neurological exam. An MRI or CT can then be done to confirm the diagnosis fully. However, sometimes, these tumours do not affect the functions at the early stages and can be hard to spot on the images, leading to either time-consuming analysis or misdiagnoses.

Machine Learning is a branch of artificial intelligence. It is a technique of allowing computers to learn, hence machine learning. In machine learning, models are trained and used when achieving a task. Models are dynamic algorithms that continue to learn and improve when given (a) dataset/parameters. There are also multiple machine learning types: supervised, unsupervised, transfer and reinforcement. These different techniques create different models which affect accuracy depending on the task at hand.

Commented [GU1]: Chapter 1 : Introduction

1.1 Problem Background

Commented [TSH2R1]: done

Commented [GU3]: All para - set to justify.

Commented [GU4]: reference

Machine Learning is capable of extracting multiple hidden patterns within a dataset, which can be extrapolated based on new data. With sufficient variables and data, machine learning is capable of achieving predictive abilities that humans are unable to accomplish with high accuracy and speed. In brain tumor diagnosis, machine learning has the potential to detect very early stages of brain cancer. It can identify the type and stage of the tumor and compute the location of the tumor in the brain. Machine Learning is able to provide medical professionals with information that can be easily overlooked.

This project aims to develop a web app capable of assisting medical professionals/researchers in diagnosing brain cancers. Throughout the production process, multiple machine learning models will be utilised and analysed to discover the upsides and downsides of each model. The analyses will then be visualised and displayed to show their differences. This project will be done back-end for front-end.

1.2 Problem Statement

Machine Learning has gained popularity in the recent years as personal computers improved. Many members of the public who could use had a computer was able to perform machine learning. Despite this, in our project, there were many academic journals of Brain Tumour Classification using Machine Learning Models which were explained thoroughly. However, there was a lack in comparison between the models and their effectiveness in this task. Different machine learning models excel at different tasks. Certainly, there must be a model or a few of them best fit for brain tumour classification.

Diagnosis is an integral part of the assessment process during treatments as it helps during the treatment process by conceptualising the patient's problem, and it can determine the steps taken during the treatment process. The chance of a diagnosis error is about 5 percent. About 1 out of 20 adult patients get misdiagnosed (DocPanel, 2019). However, in a situation of a diagnosis error, it can cause major effects on the patient in the future, so preventing such issues and detecting them prematurely can increase the patient's chances of survival. The alternative solution for the patients is to go to another doctor for a second opinion on the analysis. This solution does not sound bad until you factor in the cost and the time it will take. The time taken will cause the treatment to start later, which could be detrimental to the patient's health and the cost of having to pay for another doctor is expensive.

This project's goal is to assist doctors during the assessment process. The system will give a second opinion to the doctor for the decision during diagnosis. The primary method of the project is the AI image classifier. The purpose of the plan is to learn and predict the state of the patients through medical imaging.

1.3 Project Objective

The objective of this project is to:

1. To find out which classification machine learning model is the best for brain tumor analysis
2. To train a machine learning model to have >60% accuracy
3. To create a web app to easily utilise the trained machine learning model

1.4 Scope of the Project

This project is an analysis paper on brain tumor classification using image classification. Due to the limitations of resources, we can get and the time constraints, this project will be operating under these conditions to give the best result possible to the best of our abilities in 3 to 4 months. The system in this website cannot be used in actual diagnosis, it is simply a prototype and an analysis on the matter.

This project will be developed using Python, using PyCharm. This project will be focused on machine learning image classification models. Data sets used to train these models will only be collected online from websites such as Kaggle, and FigShare. The system will be able to be accessed from a website that will be made. StreamLit and CSS will be used to develop the system to be used.

The accuracy of said models should be above >60%. The percentage is not as high due to the limited number of data sets, we could acquire and use. The performance is terms of speed will have the models will have to be compromised in exchange for accuracy. The project will also be limited by the hardware we are using. The devices we will be using to train these models are not as powerful. Some data sets collected might not even be used in the training because of hardware limitations. The time it takes to train a single model could take a long time depending on the amount of data and the complexity of the algorithm.

At the end of the project, people can access the website and upload images for brain tumour classification. In addition, the result of the analysis will prove to be beneficial for others who wish to access the analysis.

1.5 Hardware and Software Requirements

Requirements to develop the system:

Hardware	Software
• AMD Ryzen 5 2600/ or Intel Equivalent • 8 GB RAM • 12 GB HDD/SSD Storage Space • Windows 10 and above	- Internet Access - Python Libraries (Scikit-Learn, Open-CV, NumPy) - Python 3.9 and Above - IDE of Choice (Supports Python,HTML,JS)

Requirements to run the system:

Hardware	Software
- Personal Computer - Smart Phone	- Internet Access - Web Browser

1.6 Project and System Limitations

1. Project Limitations
    a. The website will only be hosted on a laptop which does not support high traffic to our website
2. System Limitation
    a. The system only accepts MRI and CT images and it does not factor in other factors of the human body. When the system accepts other images other than MRI or CT, the results may not be accurate.
    b. The system does not accept a file of images. The system will only classify the image based on the individual image. Classifying based on one image may hurt the accuracy of the Classification
    c. The system is unable to detect the location of the tumour, only its presence within the imaging.

1.7 Project Schedule

Commented [GU5]: Change to landscape layout

The project is executed according to the Project Schedule, represented by the following Gantt Chart:

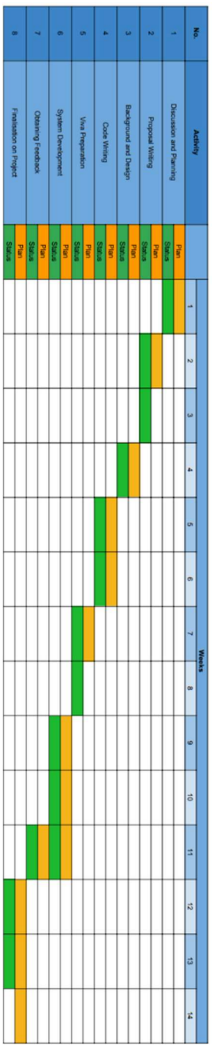


Figure 1.1 Gantt Chart

Chapter 2: Literature Review

2.1 Machine Learning

Commented [GU6]: Include one or two reference

Machine learning is the use and development of computer systems that can learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from the patterns in the data. Machine learning can also be used in our daily life, some examples that are image recognition, speech recognition, medical diagnosis and many more.

Machine learning can be separated into three different types:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Commented [GU7]: Mentioned that supervised and unsupervised learning will be explained in section 2.2.1 and 2.2.2

2.2 Supervised, Unsupervised and Reinforced Learning

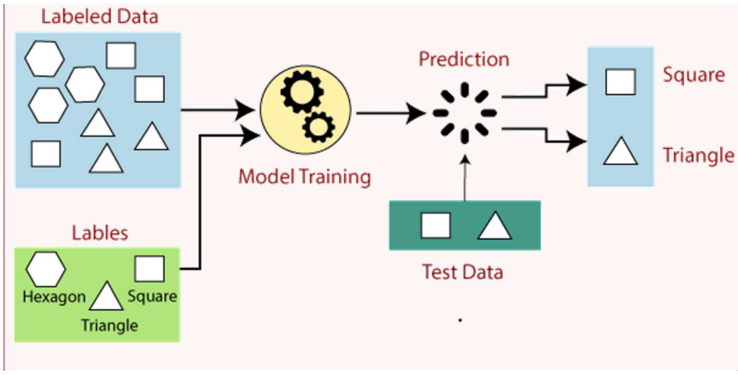
Commented [GU8]: This one put after 2.1

Supervised and unsupervised learning will be explained in the following section, 2.2.1 will be supervised learning, and 2.2.2 will be unsupervised learning, 2.2.3 will be reinforced learning

2.2.1 Supervised Learning

Commented [GU9]: Move to 2.2

Supervised learning is learning with a dataset that has labelled input and output. This means that the dataset dictates how the machine learns. The device will learn to predict based on the outcome given by the training data. In supervised learning, the training data provided to the computers serve as the supervisor, instructing them on correctly predicting the output. It implies that a pupil would learn under a teacher's guidance (Delua, 2021). Figure 2.1 briefly explains the process.



Commented [GU10]: Include figure number

Figure 2.1 – Supervised Learning Process Flow

There are two main data sets, labelled data used to train the model (x_train) and labels (y_train), and test data (x_test, y_test) which is used to test the model after the training. When training the model sufficient data is given to it so it can accurately predict the correct output.

Common supervised learning algorithms are regression and Classification. Regression is an algorithm that finds the relationship between the input and output variable. Classification is used when the output is categorical and there are 2 classes. (www.javatpoint.com, n.d.) In this project SVM will classify as a supervised learning algorithm.

2.2.2 Unsupervised Learning

Commented [GU11]: 2.3

Unsupervised learning is the opposite of supervised learning as it does not use labelled data. Unsupervised learning uses algorithms to analyse and uncover patterns, find similarities and contrast in data in clustered and unlabelled data without human intervention. Unsupervised learning is commonly used to uncover patterns and detect anomalies in large quantities of data. Unsupervised learning is also used in image recognition as it excels at analysing patterns. Figure 2.2 explains the process briefly.

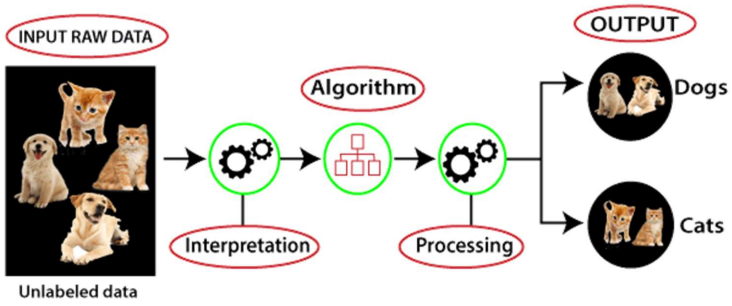


Figure 2.2 – Unsupervised Learning Process Flow

Figure 2.2 shows that the raw unlabelled input data is being used to train the machine. The images will be interpreted by the machine. What this means is that the machine is trying to find patterns in the data. Then by applying a suitable algorithm the machine can categorise the images as the output.

Common unsupervised learning algorithm are clustering and association. Clustering is a type of algorithm used to group data by its attribute (colour, distance, …). Association rule finds relationship between variable in a dataset by using a rule-based method. (IBM Cloud Education, 2020) There are many types of clustering and the algorithm that will be used in this project is K-Means clustering.

2.2.3 Reinforcement Learning

Commented [GU12]: 2.3

Reinforcement Learning is a field of machine learning where the training of the model is done with the environment itself for real experience. The model goes through the process of trial and error and tries to learn as much as possible without a specific goal. Instead of using datasets like supervised and unsupervised learning, in reinforcement learning, the model is trained with

the environment while getting rewarded for making the correct decisions and punished for making the wrong decisions. In reinforcement learning, a model is called an agent.

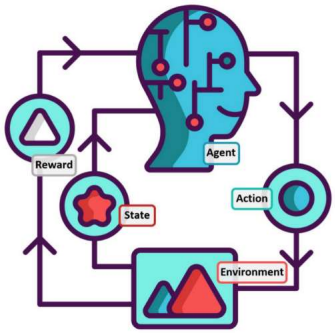


Figure 2.3 – Reinforcement Learning Process Flow

With reinforcement learning, the agent will go through the process of making an action in the environment and when the agent does the correct action or makes the correct decision, we will reward it. If the agent makes the wrong action or the wrong choice, we will punish it. After some trial and error, the agent will develop an efficient policy to make the correct decision.

Policy Gradient is a method to make the agent do the correct action more often in the future. When the agent makes a correct action, the policy gradient will increase the chances of that action happening in the future. The same thing goes for when the agent makes the wrong action except the policy gradient will decrease the chance of the action happening. In the long term this will increase the probability of the agent making the correct decision. (Insight, 2018)

2.4 Types of Classification

Classification is a method of identifying which data belongs to which category. There are two types of classifications, Binary and Multi-Class. In this paragraph, Binary Classification will be discussed. As the name suggests, binary Classification will classify the data into a binary labelled as 1 or 0. An example of a binary classification would be an email spam detector. When an email comes in, the sensor would have to classify the email as spam or primary; the email is labelled into two classes, a binary classification. There are many types of binary classification algorithms; some of them are,

1. Logistic Regression
2. Support Vector Machine (SVM)
3. K-Nearest Neighbour (KNN)

The binary classification algorithm this project would be using is SVM.

Multi-Class Classification is the opposite of Binary Classification. Multi-Class Classification classifies the given data into more than two classes. For example, I want the machine to label the breed of the dog with the given picture. There are many breeds of dogs, and it is not strictly binary like binary Classification. Some of the popular Multi-Class Classification Algorithm would be,

- K-Nearest Neighbour (KNN)
- K-Means Clustering (K-Means)
- Decision Tree

The Multi-Class Classification algorithm this project would be using is K-Means.

2.5 Image Pre-Processing

In order for an image to be recognised and processed in machine learning models, they require undergoing a few processes in order to transform the images into features and characteristics that the algorithm can recognise. The images are simplified and tuned for ease of understandment of each model. An example will be used to ease the explanation of preprocessing. Figure 2.3.1 is photo taken by Jonas Kakaroto of a pink rose downloaded on Pexels.

Figure 2.4.1 shows the sample rose image.

Firstly, the image has to be imported. In this case, A Computer Vision library is used to import these images to be processed, there are many libraries available that can achieve this. OpenCV , a free computer vision library developed by Intel. It has a vast detailed documentation available and most widely used.

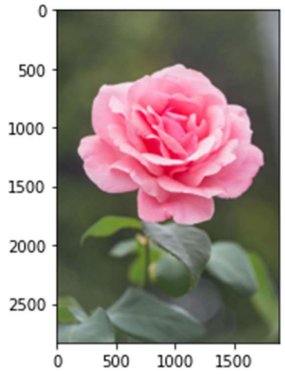


Figure 2.4.2 shows the image displayed by python

The image has to converted into an n-dimensional array first of all. In this way, the features of the images are represented by numbers which is easily interpreted by computers. This is automatically done when importing with OpenCV.

```
array([[[ 51,  59,  52],
        [ 51,  59,  52],
        [ 51,  59,  52],
        ...,
        [144, 144, 138],
        [145, 145, 139],
        [145, 145, 139]],
```

Figure 2.4.3 shows the array of the image

In Figure 2.4.3 Each cell represents the 3 color channel values.

One of the most widely used methods to preprocess images is to grayscale the image. It simplifies the features of the image as each pixel were storing 3 values each for each color channel, red, green and blue values. By grayscaling the image, the pixel values are transformed into 1 value, which is the intensity of the pixel. This reduces the number of numbers taken into account when running it through the model, especially when the colour of the image is irrelevant to its results.

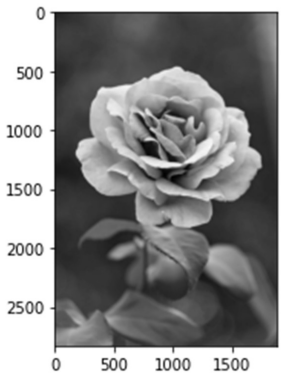


Figure 2.4.4 – Rose in Greyscale

```
array([[ 56,  56,  56, ..., 143, 144, 144],
       [ 56,  56,  56, ..., 143, 144, 144],
       [ 56,  56,  56, ..., 144, 144, 144],
       ...,
       [129, 130, 130, ...,  82,  82,  82],
       [129, 129, 129, ...,  82,  82,  82],
       [128, 128, 129, ...,  82,  82,  82]], dtype=uint8)
```

Figure 2.4.5 – Single Color Channel Array

Secondly is resizing the image, by resizing the image, The image is able to be processed easier. Machine learning models are able to train faster with over 10000 resized images. The resolution of this image is 2829 x 1886. Below is the shape of the array generated when importing the image.

```
(2829, 1886)
```

Figure 2.4.6 – Array Shape

We can see the image is represented as a large matrix of pixel values by the computer vision library.

By using OpenCV, we can resize the image. For this example, Figure 2.4.7 is resized to 256 x 256

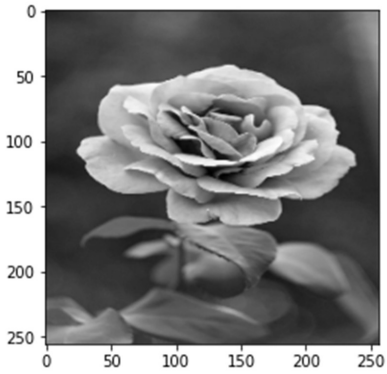


Figure 2.4.7 – Resized Image

Through image processing, we can improve the accuracy along with efficiency of the model training process.

2.6 Machine Learning Models

Commented [GU13]: 2.5 Machine Learning Models
2.5.1 SVM
2.5.2 ...

2.6.1 Support Vector Machine (SVM)

SVM is a powerful yet flexible algorithm as there is a lot of applications for this algorithm and the accuracy of the algorithm is maximised from the data given. It is considered a supervised classification and regression algorithm. Applications for this algorithm include face recognition, image recognition, Bioinformatics and many more

The goal of the algorithm is to separate the output on the graph with a hyperplane. The criteria for a hyperplane are to have a point that is equal in the distance on each side of the plane. With these criteria in mind many hyperplanes can be drawn on the graph

Examples of hyperplane that can be drawn, Figure 2.5.1 represents hyperplane 1 and figure 2.5.2 represents hyperplane 2

Hyperplane 1

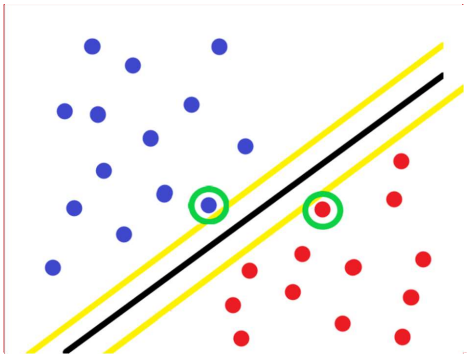


Figure 2.5.1 – Hyperplane 1

Hyperplane 2

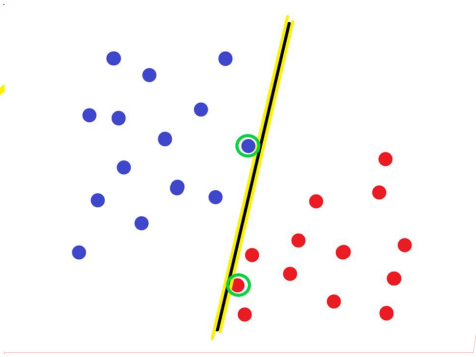


Figure 2.5.2 – Hyperplane 2

Commented [GU14]: Put caption.

Figure 2.2 - ....

Green circle highlights the closest point, blackline is the hyperplane, yellow lines are the margin. Figure 2.5.1 and 2.5.2 are represented in 2 dimensions

The question now becomes how do we determine which hyperplane is better than the other? To determine the best hyperplane all we need to do is to maximise the margin. In the example that I gave, hyperplane 1 is better.

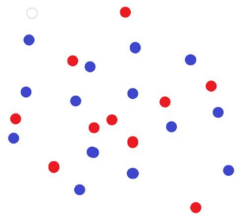


Figure 2.5.3 – Example plot in 2 Dimensions

Figure 2.5.3 is represented in 2 dimensions

In these type of data sets where the points plotted on a 2-dimensional graph are all clustered up, the solution is to use a kernel function. Kernel is a function used to transform the data into

another form. (Tim, 2019) by using the kernel function, the dimension of the data points plotted is changed. There are many kernel functions such as linear, polynomial and many more.

Kernel function

$X1^2 + X2^2$ -> X3, (Tim, 2019)

Figure 2,13 represents in 3 dimensions,

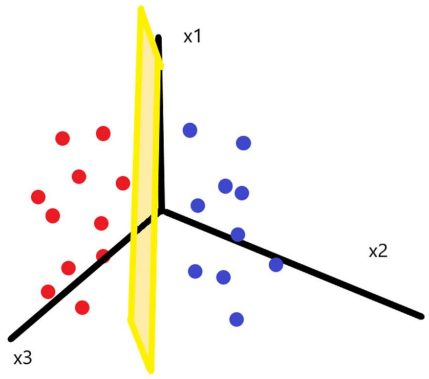


Figure 2.5.4 – 3 Dimensional Plot

yellow box represents a hyperplane in Figure 2.5.4

After applying the kernel function, the machine can determine the best hyperplane. If the data points are still cluttered the same process is done again, to increase the dimension of the hyperplane.

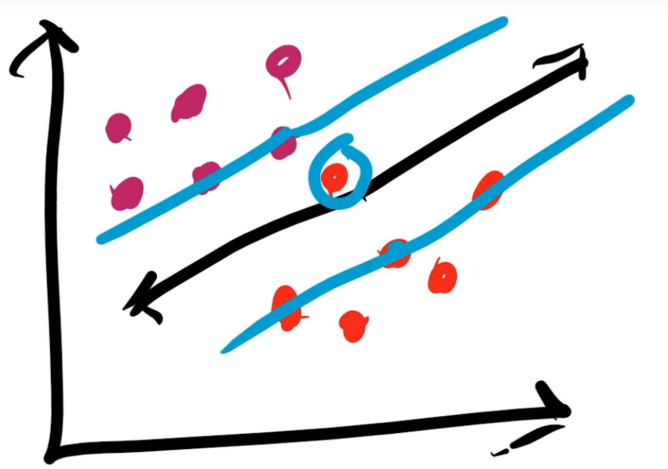


Figure 2.5.5 – Hyperplane Intersection with Datapoint

In a strict circumstance, there should be no points residing in the margin, however, there comes a time when we would allow a point to be in the margin to produce a better hyperplane. By doing that the process is called a soft margin.

In Figure 2.5.5 the red point circled represents a point that resides in the margin in order to produce a better hyperplane.

2.6.2 K-Means

Commented [GU15]: De?

K-Means is an example of an unsupervised clustering algorithm. It will cluster the data into K – number of unique clusters. Unlike other algorithms, this algorithm is not given labels to classify the given dataset into, Instead, Centroids are formed depending on the number of clusters, 1 per, which is assigned according to the central coordinates of each cluster.

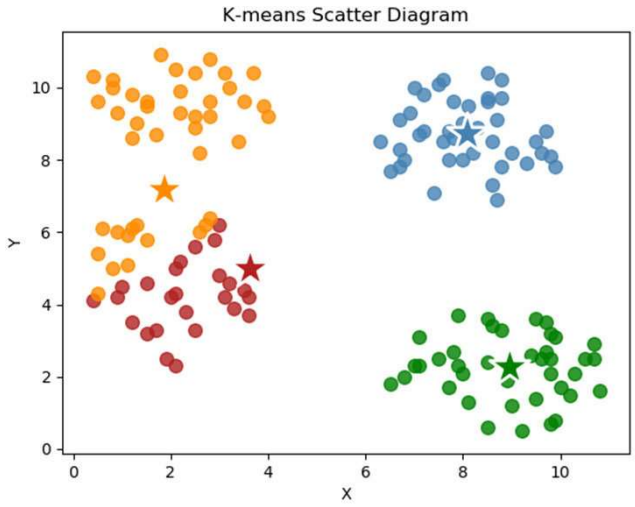


Figure 2.6.1 – K-means Scatter Diagram

The centroids find the 'next nearest centre' by comparing its distance to the data points and then update their new positions as the new centre. This process repeats till the centroids no longer move. Each cluster would then represent one class, and its data points under each cluster is classified to its respective cluster.

Algorithm:

- Choose Original Starting Centroids
- Assign each point to a nearest centroid, re-compute cluster centroids.
- Repeat till centroids are stationary

K-Means aims common objective function is using Sum of Square Error of the euclidean distance of the points.

Each data point is calculated, the error of the data point is the distance to the nearest cluster center, to obtain the SSE, the errors of each datapoint is squared and summed.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

Figure 2.6.1.1 – Formula of SSE

X represents a data point in cluster C_i and m_i is the average of cluster C_i

2.6.3 Teachable Machines

Teachable machine is a web-based tool to quickly create machine learning models without any programming language. The teachable machine is developed by google with the intention of making AI more accessible to everyone by making it easy to use. In this project, the focus will be on image classification. Image classification with teachable machine uses a supervised machine learning algorithm. Teachable machine builds itself on the TensorFlow library.

Unlike all the other methods in this project, teachable machines do not require the user to do any image pre-processing. It has a simple drag-and-drop interface and it even let the user take hundreds of pictures on the website for model training. The user does not need to choose the algorithm for image classification the user simply needs to add the images in each class for training to develop the model.

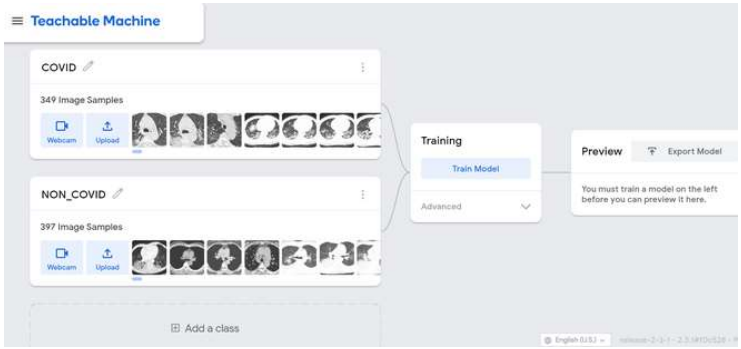


Figure 2.6.2 – Teachable Machine Page

After training the model, the user may preview the accuracy of said model and the user can also upload images to use the model. The model may be exported so we can use it in our system if we need to do so.

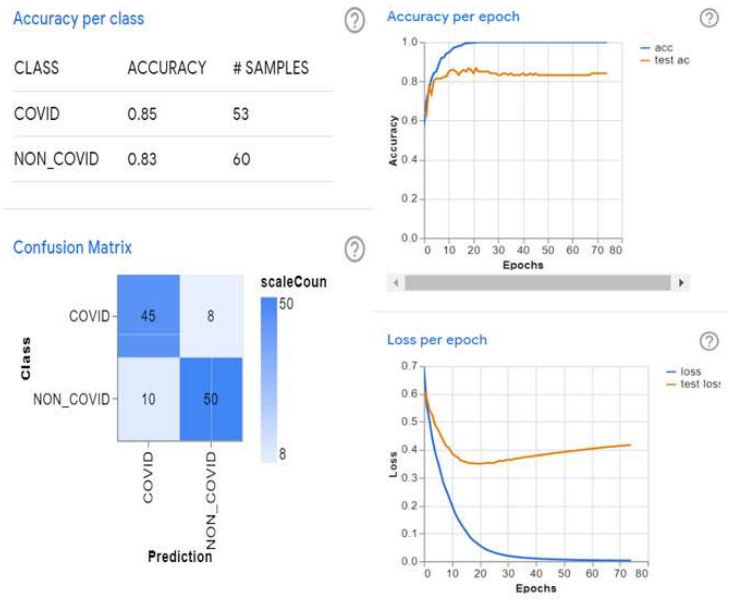


Figure 2.6.3 – Descriptive Classification Metrics/ Training

2.6.4 Convolutional Neural Network (CNN)

CNNs are Convolutional Neural Networks, CNNs are specialised in processing data in grid-like format, e.g matrices. When images are processed and imported by computer vision libraries they are in a n-dimensional array (matrix) making it perfect for image processing as a neural network. CNN's processes information alike our senses, through recognising patterns of significance to determine the object.

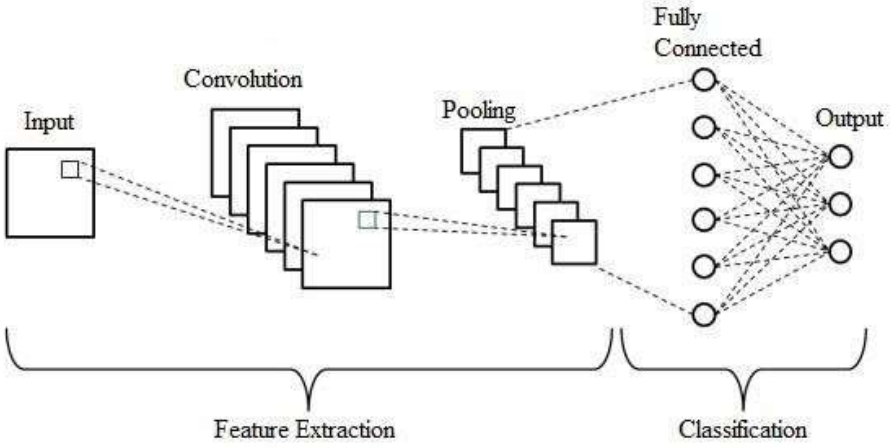


Figure 2.6.4 – CNN Architecture

A CNN's typical architecture consists of Convolution Layer, Pooling and a Fully Connected layer

2.6.5 Convolutional Layer

The convolutional layer is the backbone of the CNN.

The layer performs a product between 2 matrices, one being the image and the kernel. The kernel is mapped onto all areas of the input data. The kernel has also a set of parameters such as size which can alter the effect of the convolutional layer.

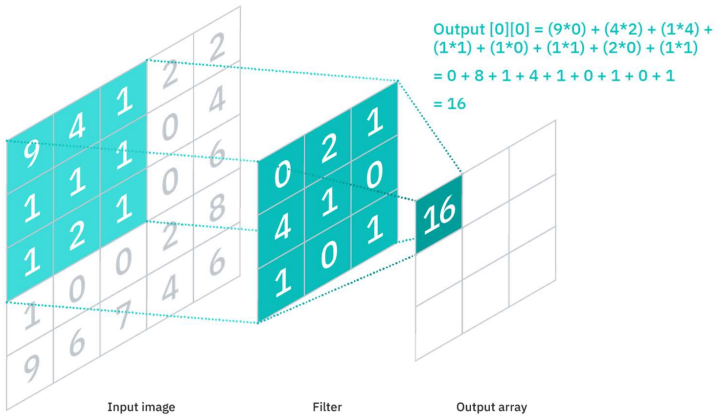


Figure 2.6.5 – Convolution Process

Convolutional Layers helps extract and reduce the parameters of training the model whilst maintaining the features of the image. By reducing the many millions of pixels of an image, the layer transforms the matrices to hundreds of pixels while maintaining the relevant information.

2.6.6 Pooling Layer

The pooling layer retrieves a summarised value based on its surrounding cell values. It helps in reducing the size of the input data, increasing computational efficiency.

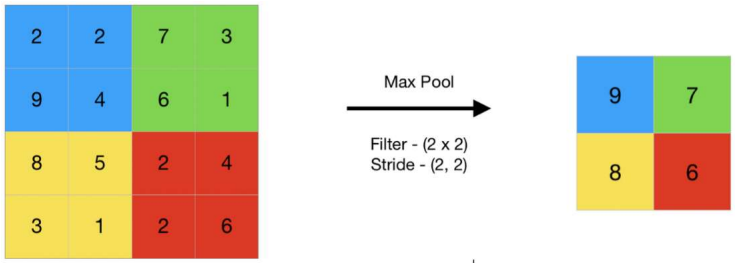


Figure 2.6.6 – Pooling Process

The pooling layer also summarises the product of the convolutional layer.

2.6.7 Fully Connected Layer

The Fully Connected Layer, FC layer contains the biases and weight to classify the image. It helps reduce the amount of human input for the CNN. The biases and weight along with the neurons perform the classification process along with other classifying hidden layers to output a class of the input data.

2.7 Related Works for Other Brain Tumour Prediction

Commented [GU16]: include reference/citation

Commented [TH17R16]: done

Based on other research on brain tumour prediction, we have read through some other papers regarding the same system, but their algorithms are different compared to ours. There is one paper where they are doing the same thing. Still, they are using Artificial Neural Network (ANN) and Convolution Neural Network (CNN) to detect the presence of brain tumours, and after that, the performance is analyzed. (P. Gokila Brindha, M. Kavinraj, P. Manivasakam & P. Prasanth, 2021)

Then, in another paper, we found out that another company is using another algorithm for brain tumour detection: Support Vector Machines (SVM). It detects by getting an image stored in the system, designating it some weightage based on the different objects of the image, and then distinguishing them from each other. It is more accurate compared with other algorithms used. (Gaur, L., Bhandari, M., Razdan, T., Mallik, S., & Zhao, Z., 2022)

Another research uses another kind of algorithm: a Quantum Variational Classifier (QVR). They use QVR to discriminate between glioma, meningioma, no tumour, and pituitary tumour. The images classified as tumours will be passed to the Seg network, where the actual infected region is segmented to analyze the tumour seriousness level. (Javeria Amin, Muhammad Almas Anjum, Muhammad Sharif, & Saima Jabeen, 2022).

Then from another paper, we found how they identify the brain tumour by using Magnetic Resonance Imaging images that have already been recognized as more detailed and more consistent images when compared to Computed Tomography images. (Julio A. Chalela, Chelsea S. Kidwell, Lauran M. Nentwich, Marie Luby, John A. Butman, Andrew M. Demchuk, Michael D. Hill, Nicholas Patronas, Lawrence Latour & Steven Warach, 2007)

The last paper we read is where people use Deep Learning to find the areas of the tumours. Other than that, Positron Emission Tomography, Cerebral Arteriogram, Lumbar Puncture, and Molecular Testing are also used for brain tumour detection. (Avigyan Sinha, Aneesh R.P., Malavika Suresh, Nitha Mohan R., Abinaya D., Ashwin G. Singerji, 2021)

Chapter 3: System Analysis And Methodology

3.1 Research Methodology

1. Data Mining Process.

   The data mining process encapsulates the steps to how information is retrieved from mining large amounts of data. To extract information out of large amounts of data requires undergoing certain processes otherwise the results might be deemed inaccurate. This is sometimes referred to as the KDD process as well, Knowledge Discovery in Databases.

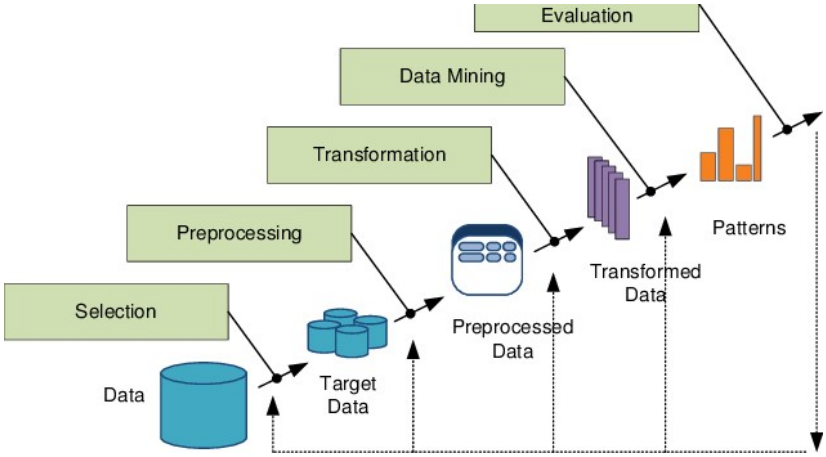


Figure 3.1 – KDD , Data Mining Methodology

   a. Data Selection
      i. This step is sometimes known as data collection. In this phase data is being surveyed and collected from websites, databases, academic experiments/ journals and more.
      ii. In this project, A combined datasets will be utilised, the data sets obtained consist of high contrast medical images of the human brain. They are both acquired off Kaggle, a popular dataset sharing site. Both datasets has 4 categories, glioma, meningioma, pituitary and no tumor. The first 3 classes represent the type of brain cancer while no tumor represents a clean non-cancerous image. The images are all in the form of jpg.

iii. First Dataset is provided by Masoud Nickparvar under a CC0: Public Domain Licence, CC0 License allows us to use freely as there is no copyrights on it. According to the metadata, it is sourced from other kaggle and figshare datasets. This dataset has 1621 glioma images, 1645 meningioma images, 1757 pituitary images, and finally 2000 non cancerous images. This data set has a total of 7022 images of brain imaging scans

iv. Second Dataset is provided by Sartaj under a CC0: Public License as well. The metadata states that it is sourced from other datasets. However the authors of this dataset did get it verified by doctor. This dataset has 928 glioma images, 937 meningioma images, 901 pituitary images, and finally 105 non cancerous images. This data set has a total of 3264 images of brain imaging scans

v. When combined the uncleaned data has a total of 10286 images. The images will be split and cleaned for duplicates in the following processes.

vi. A Short Exploratory Data Analysis, EDA Report:

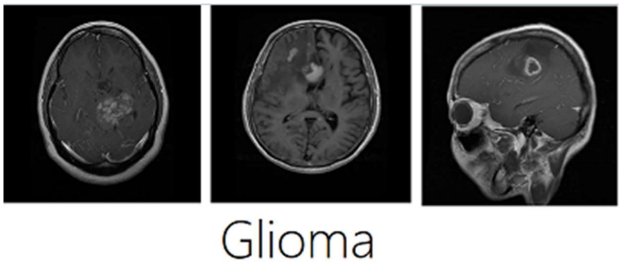


Figure 3.2.1 – Glioma MRI Images

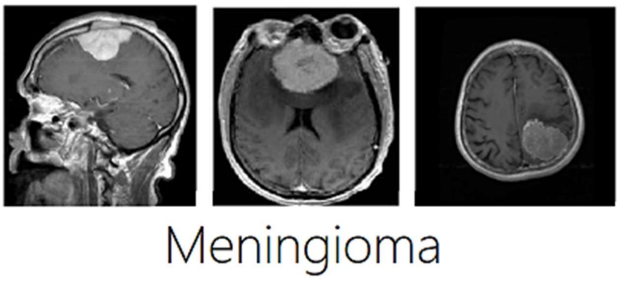


Figure 3.2.2 – Meningioma MRI Images

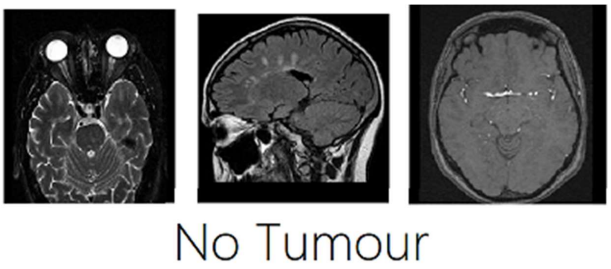


Figure 3.2.3 – No Tumour MRI Images

Figure 3.2.4 – Pituitary Tumour MRI Images

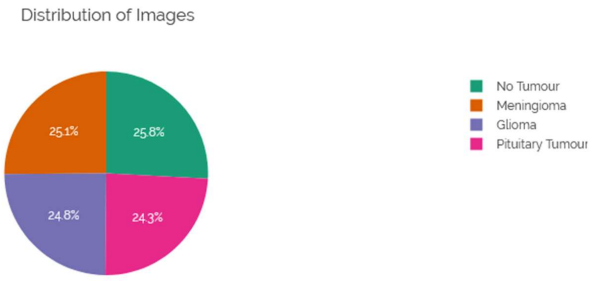


Figure 3.2.5 – Distribution of Dataset Images

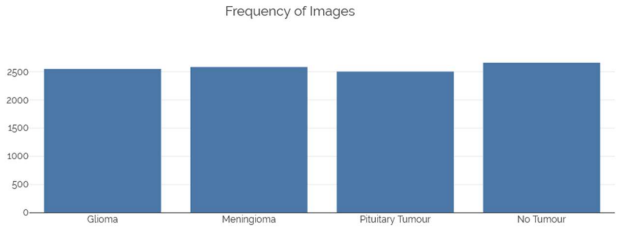


Figure 3.2.6 – Counts of Dataset Images

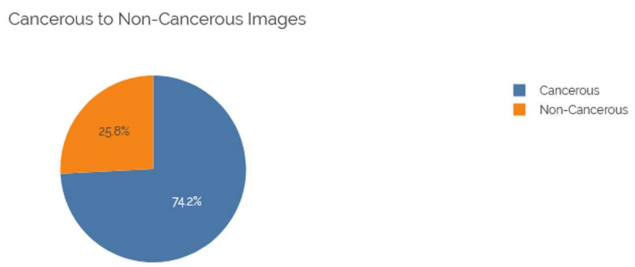


Figure 3.2.7 – Ratio of Cancerous to Non-Cancerous Images

b. Data Pre-processing

Data Pre-processing plays an important role to an accurate model as the data used to train the model has to be consistent and accurate. Before transforming the data, the data have to processed to remove inconsistency in the dataset.

In this project, we are using 2 datasets and for the purpose of training the models, the datasets have to be combined so as to have more training data for the models.

To avoid duplicate images during the merge of the datasets, a python program is made with a function to find duplicate images. Duplicate images could cause overfitting which could lower the accuracy of the models. The main function of the duplicate image detector is a mean squared error function. The purpose of the function is to find the difference between the two matrices of the images.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Figure 3.3 Mean Squared Error Formula

This function is usually used to measure the number of errors in a statistical model. However, in this case, the error can be the difference between the matrix. The operation

is similar despite being modified. The operation goes as follows matrix is subtracted from one another then the subtracted matrix is raised to the power of 2. Then all the number in the matrix is summed before dividing by the matrix's size (128*128). If the image is exactly identical the returned value will be 0. The smaller the returned value the more similar the images are.

The function works because the number in the matrix represents a pixel in the image. If hypothetically the image is identical, the subtraction of the matrix will be 0. The higher the value of the subtracted matrix the more differences the images have.

a. Data Transformation

Data transformation is a process of turning data from one format to another. In this project, we will be converting the pre-processed images into arrays. The goal of this process is to convert the data into a form that is easier for the machine to interpret. By doing this, it will increase the efficiency of the learning process, and increase the accuracy of the model.

The images will be resized to 128px, 128px to have a consistent image size throughout the training process and the images in the dataset will be converted to grayscale to simplify the dataset and to reduce colour complexity so the image can be converted into a simpler array. By doing this the training process will also be sped up and simplified.

[255, 0, 255]

Figure 3.4.1 RGB image array

[70]

Figure 3.4.2 Grayscale image array

RGB images array requires more memory and it is less efficient as it needs 3 values to represent each pixel. Lastly, the grayscale images are converted into arrays so the machines can interpret the data more efficiently as the models can understand the features of an image. This can be done using the OpenCV library

c. Data Mining

Data mining is the process of extracting patterns from huge datasets. The process of extracting the patterns in the datasets differs when using different machine learning algorithms. In this project, each algorithm that is chosen represents a different branch of machine learning. In addition, all the algorithms chosen can do multi-class classification which is more suitable for the datasets chosen for this project.

- SVM is a supervised algorithm
- K-Means is an unsupervised algorithm
- Teachable Machine uses transfer learning
- CNN uses deep learning

The goal of this step is to provide results for comparison between each algorithm as the goal of this project is to find the best algorithm for brain tumor analysis.

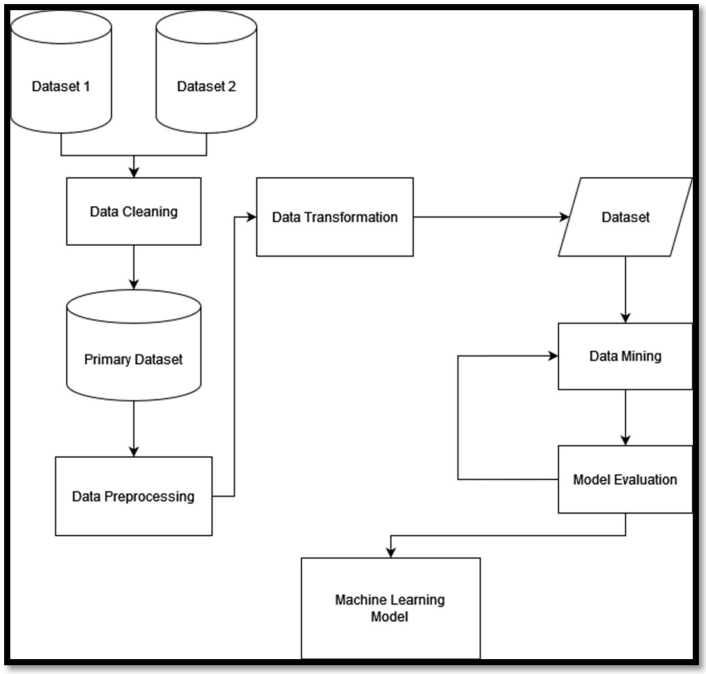


Figure 3.4.3 – Backend Process Flow

d. Evaluation

i. The models have to be evaluated to test its ability to predict with accuracy. There are dozens of metrics that can help to show the model's ability however it does depend on the goal of your model. All of our models will be classifying different classes therefore, the types of methodology that will be covered will be primarily aimed at multi-class classification.

ii. Confusion Matrix

Confusion Matrices are 2x2 tables which list the results of the model versus the actual results. In this case, The confusion matrix will be NxN as this study does multiclass classification. N will represent the number of classes. They display the TP (True Positive), FP (False Positive) , TN (True Negative) , FN (False Negative) for each class. By comparing the numbers a number of results can be extracted.

		Actual		
		Not Sick	**Sick**	**Total**
Prediction	**Not Sick**	**43**	**9**	52
	Sick	**4**	**44**	48
	Total	47	53	100

Figure 3.5.1 Example N * N Confusion Matrix

iii. Precision

Precision is the percentage of true positive's out of all of the predicted positives. Displays the ability for it to identify the true predictions the class

$$\text{Precision} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FP}}$$

Figure 3.5.2 – Precision Formula

iv. Recall

Recall is percentage of true positive's out of the true positives summed with the false negatives. Displays the ability for it to identify the true positives correctly classified

$$\text{Recall} = \frac{\sum \text{TP}}{\sum \text{TP} + \text{FN}}$$

Figure 3.5.3 – Recall Formula

v. F1-Score

F1 Score is the combination of the accuracy and recall of a classifier into one metric by taking the harmonic mean. It is mainly used to compare the performance of two classifiers.

P = Precision

R = Recall

$$\frac{2(P * R)}{P + R}$$

Figure 3.5.4 – F1 Score Formula

3.2 Web Based System

Low Fidelity Design of the system

Commented [GU18]: include 2-3 sentences about this

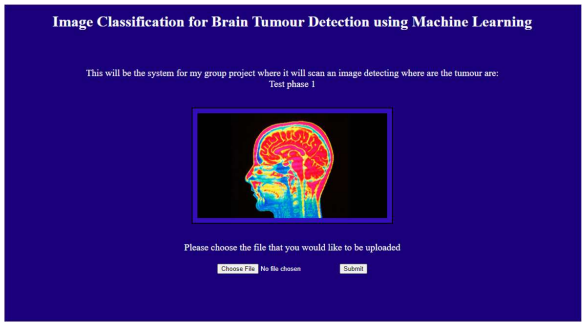


Figure 3.6.1 Home Page

In the low fidelity design, the user can upload any image to the system and once uploaded they will have a preview of the file. After uploading they are able to view the image then once predict, they are able to view whether is there cancer presence and the type of cancer.

High Fidelity

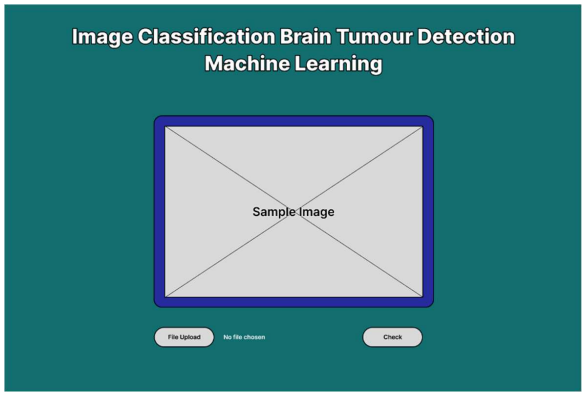


Figure 3.6.2 Home Page

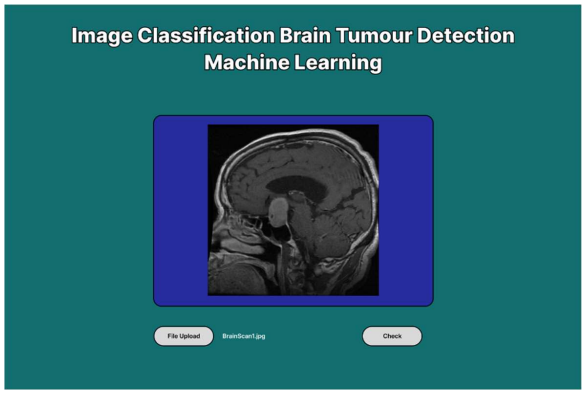


Figure 3.6.3 Home Page with Image Uploaded

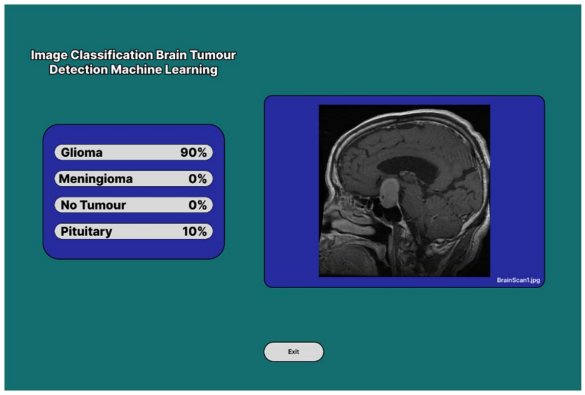


Figure 3.6.4 Prediction Page

The system will prompt the user to upload an image of a brain scan; once the image has been uploaded, the user can proceed with the submission, and the system will scan through the image for any possible brain tumour. Once scanned, if there is a presence of brain tumour it will prompt the percentage of the tumour.

3.3 Programming Languages

The backend of this project is written in Python, Python is High-Level Programming language. Python can use many different machine learning libraries such as SKLearn, Tensorflow. Both libraries are free-to-use and popular among the coding community. Tensorflow provides Machine Learning kits to construct our own models, while sklearn provides many tools to preprocess, clean and train models as well. This project also uses, Image Processing libraries, such as PIL and OpenCV. Python also supports Data Visualisation libraries such as Plotly and Seaborn. Finally popular data science libraries such as Pandas and Numpy for data management.

For the frontend, the languages used for this group project are consisting of Cascading Style Sheets (CSS), and Python. We are also using StreamLit which is a open-source python-based framework that allows developing and deploying interactive machine learning models.

We are using StreamLit for the development of the website, CSS for the design and the style for it and finally Python where is the whole brain of the system.

3.4 Process Flow

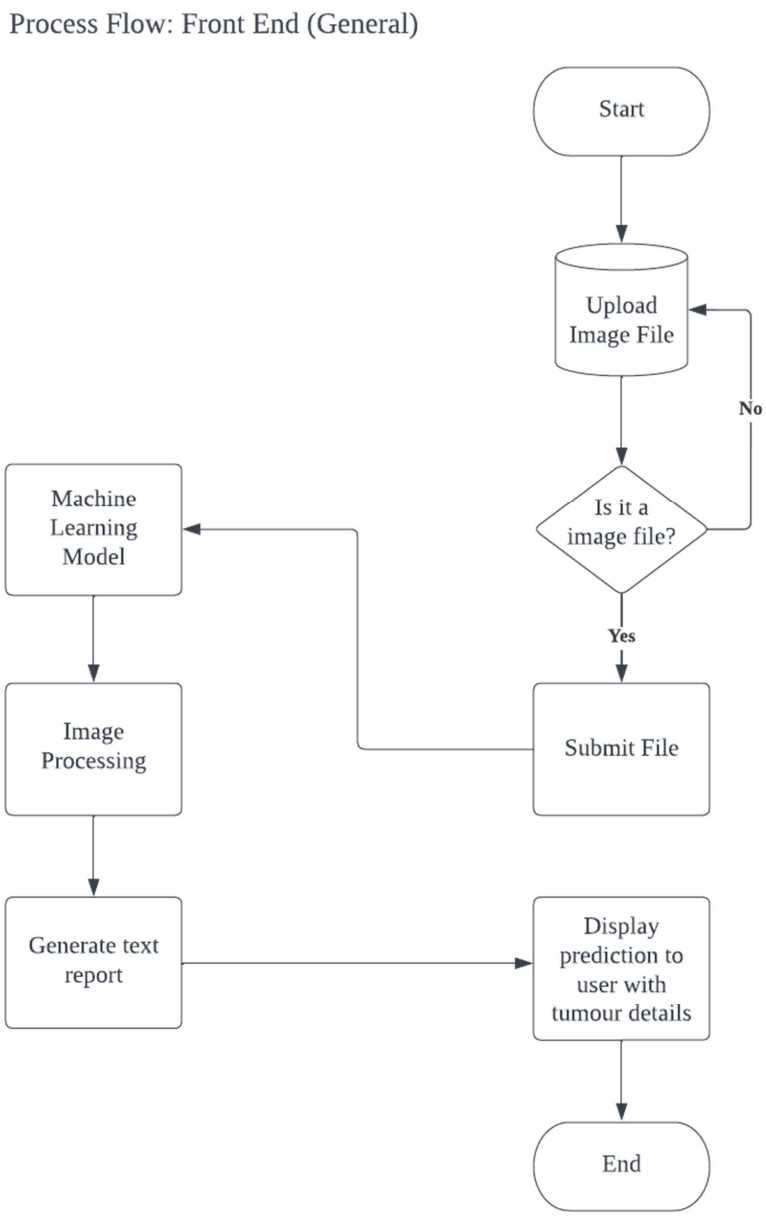


Figure 3.7 Process Flow of the Front End

Chapter 4: System Design & Implementation

Commented [GU19]: Include Chap Name

4.1 Image Processing

This code is used for image processing.

```
import os
import cv2
```

- OS is imported to get the file path and to walk through the directory
- OpenCV is imported for image resizing and to convert images into array

```
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
image_dir = os.path.join(BASE_DIR, "CombinedDataset")

for root,dirs,files in os.walk(image_dir):
    for file in files:
        if file.endswith("png") or file.endswith("jpg"):
            path = os.path.join(root,file)

            img = cv2.imread(path)
            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            img = cv2.resize(img, (128, 128))

            cv2.imwrite(path, img)
```

Processing and saving the file at its location. The images are processed into 128x128 and greyscale using OpenCV functions `cv2.resize(img, (128, 128))`, `cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`.

4.2 Finding Duplicate Images

This is the code used to find and delete duplicate images within the combined dataset

```
import os
import numpy as np
import cv2
```

The necessary libraries will be imported here,

- NumPy is imported for array
- OpenCV is imported for image resizing and to convert images into array
- Scikit-learn is imported for the SVM algorithm to create the model and for data splitting
- OS is imported to get the file path and to walk through the directory
- Pickle is used to save the model

```
def mse(imageA, imageB):
    err = np.sum((imageA.astype("float") - imageB.astype("float")) ** 2)
    err = err / float(imageA.shape[0] * imageA.shape[1])
    return err

duplicate_image = []
duplicate_imagepath = []

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
image_dir = os.path.join(BASE_DIR, "testing")

for root,dirs,files in os.walk(image_dir):
    for file in files:
        if file.endswith("png") or file.endswith("jpg"):
            path1 = os.path.join(image_dir,file)
            img1 = cv2.imread(path1)
            img1 = cv2.resize(img1, (256, 256))

            for roots,dir,filess in os.walk(image_dir):
                for file_check in filess:
                    if file != file_check:
                        path2 = os.path.join(image_dir, file_check)
                        img2 = cv2.imread(path2)
                        img2 = cv2.resize(img2, (256, 256))
                        if mse(img1,img2) <400:
                            print("FILE DELETED")
                            if os.path.isfile(path1):
                                os.remove(path1)
                                duplicate_image.append(file)
                                duplicate_imagepath.append(path1)

print("these images has been removed")
print(duplicate_image)
```

This code can use the Mean Squared Error of the image array to determine the similarity of the images. If the MSE is below a certain threshold, the file will be deleted. This process is time and computing power intensive.

4.3 Explaining Model Implementation

In this section, code used to implement the model will be explained , A virtual environment is created with the required libraries to contain our coding.

4.3.1 Supervised Learning

SVM

```
import numpy as np
import cv2
from sklearn.model_selection import train_test_split
from sklearn import svm
import os
import pickle
```

The necessary libraries will be imported here,

- NumPy is imported for array
- OpenCV is imported for image resizing and to convert images into array
- Scikit-learn is imported for the SVM algorithm to create the model and for data splitting
- OS is imported to get the file path and to walk through the directory
- Pickle is used to save the model

```
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
image_dir = os.path.join(BASE_DIR, "CombinedDataset")

classes = {"notumor": 0, "glioma": 1, "meningioma": 2,"pituitary": 3}
y=[]
x=[]
```

BASE_DIR is a global variable to keep the main file path to the folder of the file which is C:\Users\kelvi\PycharmProjects\MachineLearningTutorial\BrainTumorClassification

Image_dir will contain the path to the folder of the data.

Classes are dictionaries used to turning the labels into numbers additionally it will be used to join the file path to a complete path.

X and Y list are declared before they are used to hold the data.

```
for cls in classes:
    path = os.path.join(image_dir,cls)
    for t in os.listdir(path):
        pth = os.path.join(path, t)
        img = cv2.imread(pth)
```

```
        x.append(img)
        y.append(classes[cls])
```

This is a nested loop to assign the images with their labels and to convert the images into arrays. The first loop on the dictionary is to get the file path. The second loop is to complete the file path to each image. Then with the OpenCV function, the images will be converted to arrays. After that the array will be appended to the x list. And the label of the array will be appended to y list.

```
x = np.array(x)
y = np.array(y)

x_updated = x.reshape(len(x),-1)

x_train, x_test, y_train, y_test
=train_test_split(x_updated,y,random_state=10,test_size=.20)

x_train = x_train/255
x_test = x_test/255
```

The lists will be converted into arrays for easier processing and it takes less memory. X array will be reshaped into a 2-dimensional array, basically flattening each of the image. `train_test_split` is a function used to randomly split up the data, the test data is partitioned for testing later. The testing data will be 20% while the training data will be 80%. Then the array will be divided by 255 as it is the maximum number. The reason 255 is the maximum number is because when the images is converted to arrays the images, the images which has been converted to greyscale can only range to 255. which will make the number $0 \geq x \leq 1$. The reason this is done is because it is easier to process smaller numbers.

```
classifier = svm.SVC()
classifier.fit(x_train, y_train)

print("training score:",classifier.score(x_train,y_train))
print("test score:",classifier.score(x_test,y_test))


with open("SVMmodelforbraintumor.pickle", "wb") as f:
    pickle.dump(classifier, f)
```

The classifier name of the model. The algorithm that this model will using svm.SVC(). Classifier.fit will train the model with the data. Score will return the accuracy value of the model. Finally, using the pickle library, the model will be saved as a pickle file using pickle.dump().

4.3.2 Unsupervised Learning

K-Means

```
import os
import cv2
import numpy as np
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import pickle
```

- OS is imported to get the file path and to walk through the directory
- OpenCV is imported for image resizing and to convert images into array
- NumPy is imported for array
- Scikit-learn is imported for the SVM algorithm to create the model and for data splitting, and to test accuracy
- Pickle is used to save the model

```
y=[]
x=[]

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
image_dir = os.path.join(BASE_DIR, "CombinedDataset")

for root,dirs,files in os.walk(image_dir):
    for file in files:
        if file.endswith("png") or file.endswith("jpg"):
            path = os.path.join(root, file)
            label = os.path.basename(os.path.dirname(path))
            img = cv2.imread(path).flatten()
            x.append(img)
            if (label == 'pituitary'):
                y.append(3)
            elif (label == 'glioma'):
                y.append(1)
            elif (label == 'meningioma'):
                y.append(2)
            else:
                y.append(0)

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

x_train, x_test, y_train, y_test = train_test_split(x,y,random_state=10,test_size=0.2)

x_train = x_train/255
x_test = x_test/255
```

This process is the same as the SVM, except when assigning the label, it uses if-else instead of a dictionary. Ultimately, the code will achieve the same outcome of assigning the dataset with the correct label and splitting it into test and train lists randomly.

```
classifier = KMeans(n_clusters=4)
classifier.fit(x_train)

y_pred = classifier.predict(x_test)
print('Accuracy: {}'.format(accuracy_score(y_test, y_pred)))


with open("KMeansforbraintumor.pickle", "wb") as f:
    pickle.dump(classifier, f)
```

In this part of the code, the model is made with the K-Means algorithm imported from scikit learn. `n_clusters=4` is specified as there are 4 types of labels which identifies each type of clusters. `fit(x_train)` is used to train the model. Since K-Means is an unsupervised learning algorithm, the only data needed is the training data. Then, using the trained model, it can predict the test data and the returned value will be the prediction. With that the accuracy of the model can be acquired with `accuracy_score(y_test, y_pred)` the model is then saved using the pickle module with the file name `KMeansforbraintumor.pickle`

4.3.3 Transfer Deep Learning

Teachable Machines

Teachables Machines is an online Machine Learning webapp made by google. which allows anyone to train a Machine Learning Model. Due to its extremely user friendly UI, It is fairly easy to train an ML Model. Teachable Machines uses a Deep Learning AI to train your model.

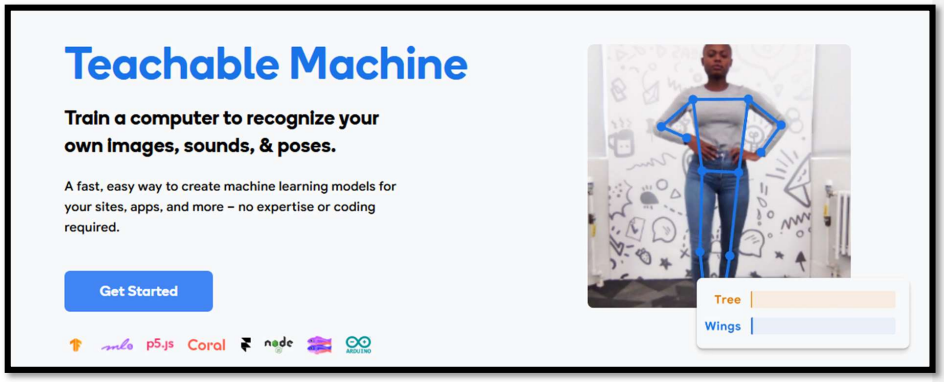


Figure 4.3.1 – Teachable Machines Homepage

Teachables Machine then displays a type of project to work on. In our study, we will be using the Image Project Setting

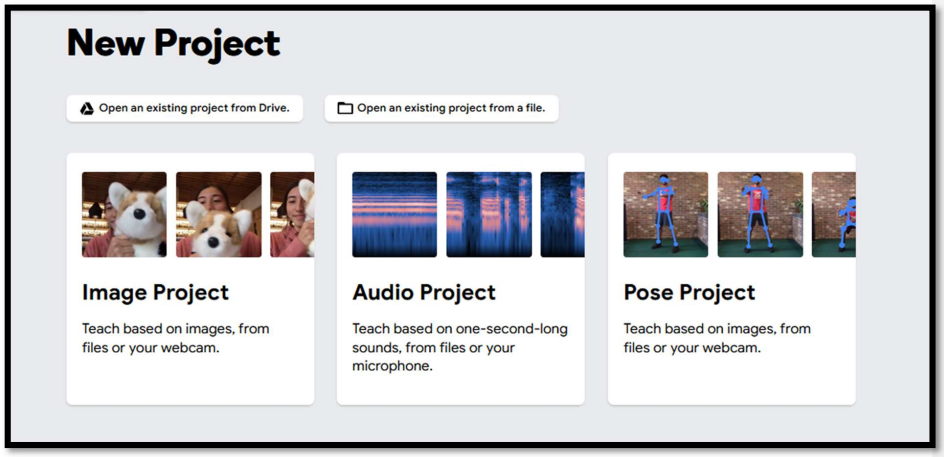


Figure 4.3.2 – Teachable Machines Project Creation

The standard image model setting is picked as it fits our project needs the most

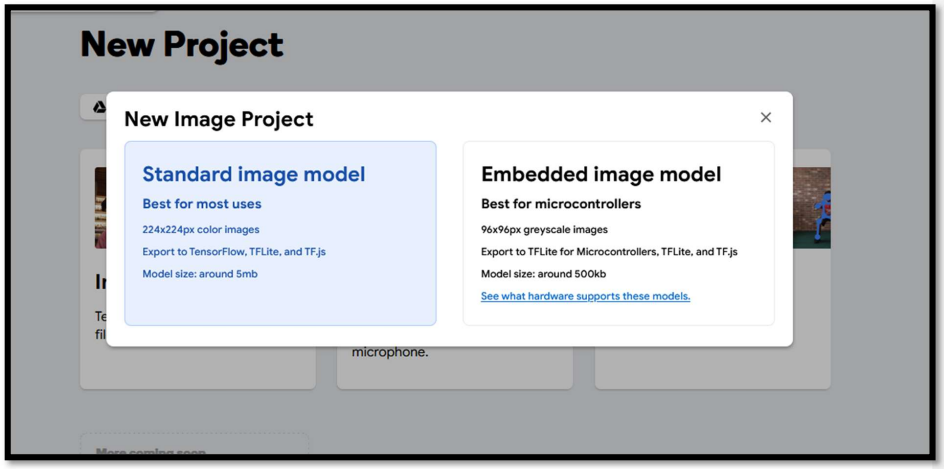


Figure 4.3.3 – Project Setting

The figure below shows the training page of the webapp. For our project, we have certain tasks to complete before training:

1. Loading in Images
2. Adding more classes
3. Renaming Classes
4. Set Training Parameters

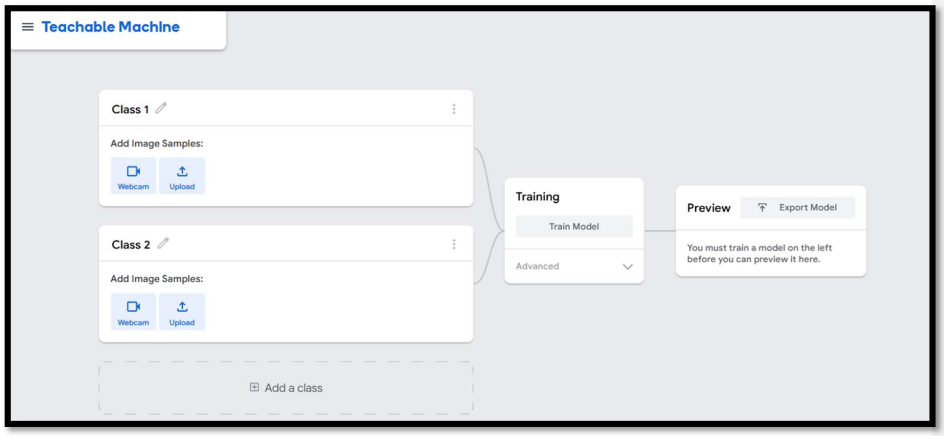


Figure 4.3.4 – Teachable Machines Training Page

The page now has the files loaded in along with the named classes. The model's traning parameters is then altered, 10 epochs is set for the constant in this study.

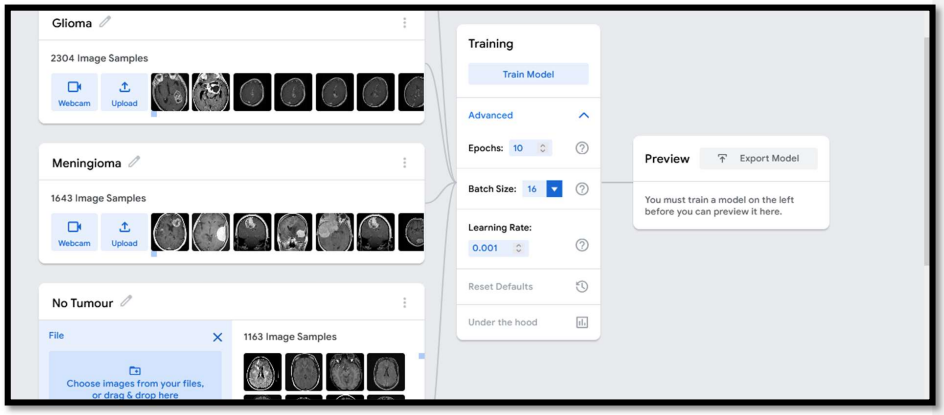


Figure 4.3.5 – Data Insertion and Parameter Setting

After training is completed, It displays a model testing screen.

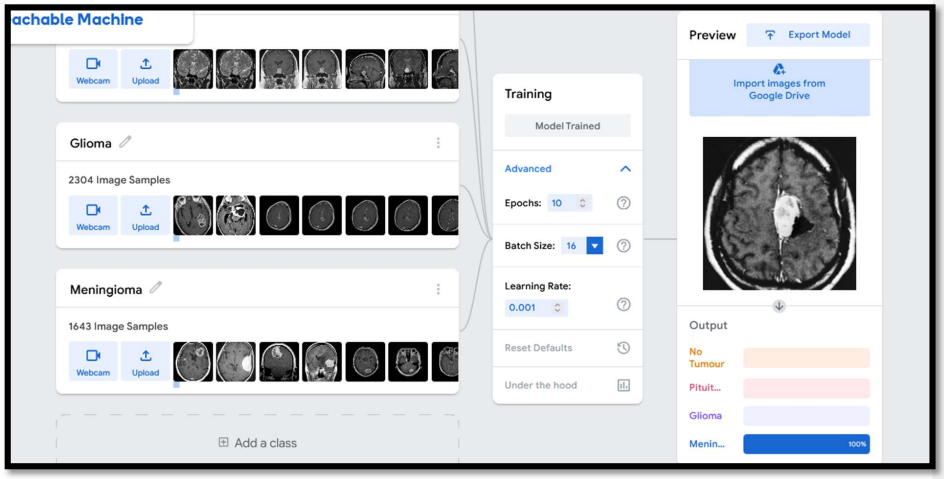


Figure 4.3.6 – Model Testing

Teachables Machines also displays the classification metrics. The system splits it 85/15, For training to testing data.

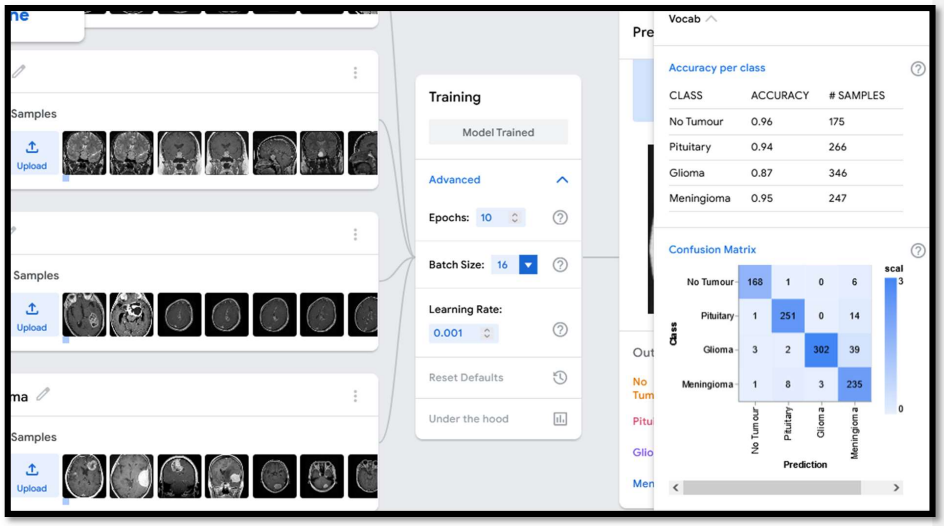


Figure 4.3.7 – Model Metrics

Teachables Machine then gives the option to save our model. The modle is saved as a keras model under the extension of .h5

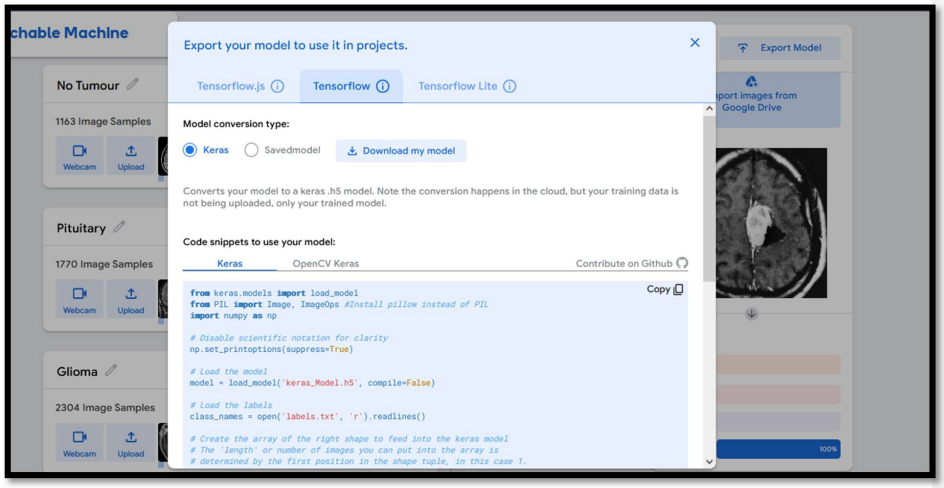


Figure 4.3.8– Model Saving

4.3.4 Deep Learning

CNN, Convolutional Neural Network

Below are the required libraries to run the program.

```
import os
import cv2
import numpy as np
from PIL import Image
from sklearn.model_selection import train_test_split
from keras.utils import normalize
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Activation, Flatten, Dense
```

- os is used to access and retrieve file directory and system information.
- cv2 is a computer vision library used to read the image file.
- Numpy is for matrix array calculations
- PIL is python imaging library, used for converting image to array
- SKLearn is for the many tools required to preprocess the data
- Keras is for the model building and layering

```
image_directory = 'datasets2/'
dataset=[]
label=[]

IMAGE_SIZE = 64

glioma_images = os.listdir(image_directory+'glioma/')
meningioma_images = os.listdir(image_directory+'meningioma/')
pituitary_images = os.listdir(image_directory+'pituitary/')
notumor_images = os.listdir(image_directory+'notumor/')
```

Image_directory stores the file path to dataset images. Two lists are made to store the overall dataset into the program and its labels. A constant of IMAGE_SIZE is set to 64 to uniformly resize the image later on. Each type of image label is given a variable to me stored in.

```
for i , image_name in enumerate(glioma_images) :
        if(image_name.split('.')[1] == 'jpg'):
            image = cv2.imread(image_directory+'glioma/'+image_name)
            image = Image.fromarray(image)
            image = image.resize((IMAGE_SIZE,IMAGE_SIZE))
            dataset.append(np.array(image))
            label.append(0)

for i, image_name in enumerate(pituitary_images):
    if (image_name.split('.')[1] == 'jpg'):
        image = cv2.imread(image_directory + 'pituitary/' + image_name)
        image = Image.fromarray(image)
        image = image.resize((IMAGE_SIZE, IMAGE_SIZE))
        dataset.append(np.array(image))
        label.append(1)

for i, image_name in enumerate(notumor_images):
    if (image_name.split('.')[1] == 'jpg'):
        image = cv2.imread(image_directory + 'notumor/' + image_name)
        image = Image.fromarray(image)
        image = image.resize((IMAGE_SIZE, IMAGE_SIZE))
        dataset.append(np.array(image))
        label.append(2)

for i, image_name in enumerate(meningioma_images):
    if (image_name.split('.')[1] == 'jpg'):
        image = cv2.imread(image_directory + 'meningioma/' + image_name)
        image = Image.fromarray(image)
        image = image.resize((IMAGE_SIZE, IMAGE_SIZE))
        dataset.append(np.array(image))
        label.append(3)
```

The code below extracts the image name, reads the, converts to array and resizes it to the IMAGE_SIZE. The processed data is then appended as 1 cell into the dataset. A label is then added depending of what image it is. 0: Glioma, 1: Pituitary, No Tumour: 2, Meningioma: 3.

```
dataset = np.array(dataset)
label = np.array(label)

x_train, x_test, y_train, y_test = train_test_split(dataset,label,test_size = 0.2)

x_train = normalize(x_train)
x_test = normalize(x_test)
```

The dataset and label lists are converted to numpy array objects.By using, train test split, The dataset and label are split into training and testing data, in a 80:20 ratio. The x_train and x_test is then feature scaled using the normalise function.

```
model= Sequential()

model.add(Conv2D(32,(3,3), input_shape=(IMAGE_SIZE,IMAGE_SIZE,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32,(3,3), kernel_initializer='he_uniform'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(32,(3,3), kernel_initializer='he_uniform'))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(128,activation=tf.nn.relu))
model.add(Dense(6,activation=tf.nn.softmax))
```

Above is the CNN Architecture used for this project. After using model.summary, the output shows us in detail of the neural network.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
activation (Activation)	(None, 62, 62, 32)	0
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	9248
activation_1 (Activation)	(None, 29, 29, 32)	0
max_pooling2d_1 (MaxPooling)	(None, 14, 14, 32)	0
conv2d_2 (Conv2D	(None, 12, 12, 32)	9248
activation_2 (Activation)	(None, 12, 12, 32)	0
max_pooling2d_2 (MaxPooling	(None, 6, 6, 32)	0
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 128)	147584
dense_1 (Dense)	(None, 6)	774

Table 1.0 – CNN Model Summary

A 2D Convolution Layer with a 3x3 kernel and 32 filter size, followed by a ReLU Activation function. ReLU stands for rectified linear activation function which is responsible for converting the sum of the weighted inputs from a node to the node's activations or outputs for that input. 3 Feature Extraction layers before the Fully Connected layer for classification. Softmax was used for the Multiclass Classification which this study needed.

```
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accura
cy'])

model.fit(x_train, y_train, batch_size=16, verbose =1, epochs=10,
          validation_data=(x_test,y_test),shuffle=False)

model.save('CNNBrainTumourClassifier10Epochs')
```

The model is compiled with sparse_categorical_crossentropy for multiclass classification. adam, aka Adaptive Movement Estimation is a combination of two gradient decent methodologies. When fitting the model. 10 epochs is the study's control to ensure fair training and prevent model overfitting or underfitting. The model is then saved.

Group 3

4.4 Model Evaluation

Many different types of machine learning branches were tested out. These models have their own pros and cons when it comes to the different situations that they are applied in. In this study, Our goal is to find out which model works best when classifying brain tumour images. The test data is taken from the test dataset of the entire project. The methods used to get these statistics are sklearn own classification metrics. The testing dataset was used to ensure unbiased results.

Supervised Method

SVM (Support Vector Machine)

Confusion Matrix

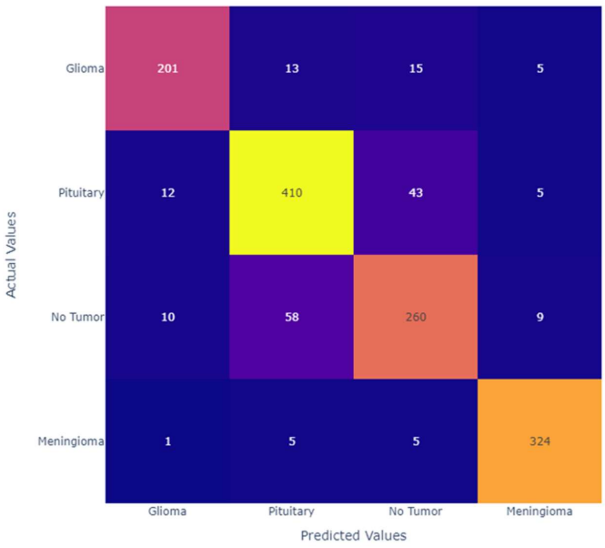


Figure 4.3.1 – SVM Confusion Matrix

	Precision	Recall	F1-score	Support
Glioma	0.9	0.86	0.88	234
Pituitary	0.84	0.87	0.86	478
No Tumour	0.8	0.77	0.79	337
Meningioma	0.94	0.97	0.96	335
Accuracy			0.87	1376
Macro Avg	0.87	0.87	0.87	1376
Weighted Avg	0.87	0.87	0.87	1376

Figure 4.1.2 – SVM Classification Report

The figures above shows that SVM has a 0.87/ 87% accuracy score overall along with a fairly accurate confusion matrix.

Unsupervised Learning

K-Means

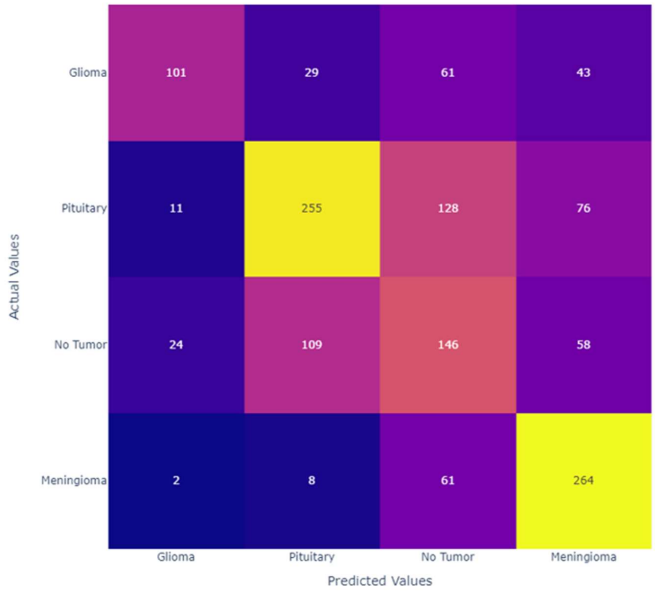


Figure 4.1.3 – K-Means Confusion Matrix

	Precision	Recall	F1-score	Support
Glioma	0.73	0.43	0.54	234
Pituitary	0.64	0.54	0.59	478
No Tumour	0.37	0.43	0.4	337
Meningioma	0.6	0.79	0.78	335
Accuracy			0.56	1376
Macro Avg	0.58	0.55	0.55	1376
Weighted Avg	0.58	0.56	0.56	1376

Figure 4.1.4 – K-Means Classifcation Report

The figures above shows that K-Means has a 0.56 accuracy score/ 56% overall along with a diverse, spread out confusion matrix.

Transfer Learning

Teachable Machines

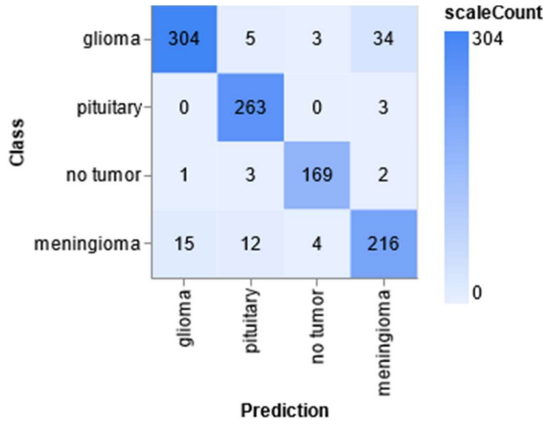


Figure 4.1.5 – Teachable Machines Confusion Matrix

CLASS	ACCURACY	# SAMPLES
glioma	0.88	346
pituitary	0.99	266
no tumor	0.97	175
meningioma	0.87	247

Figure 4.1.6 - Teachable Machines Classification Report

The figures above shows that Teachable Machines has a 0.93 / 93% accuracy score overall along with an accurate confusion matrix.

Deep Learning

CNN – Convolutional Neural Network

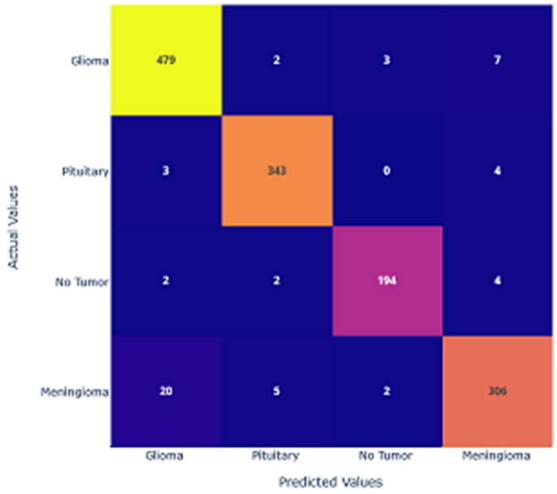


Figure 4.1.7 – CNN Confusion Matrix

	Precision	Recall	F1-score	Support
Glioma	0.95	0.97	0.96	2086
Pituitary	0.98	0.98	0.98	1587
No Tumour	0.97	0.98	0.97	1057
Meningioma	0.96	0.91	0.93	1462
Accuracy			0.96	6192
Macro Avg	0.96	0.96	0.96	6192
Weighted Avg	0.96	0.96	0.96	6192

Figure 4.1.8 - CNN Classification Report

The figures above shows that CNN, Convolutional Neural Network has a 0.96/ 96% accuracy score overall along with an accurate confusion matrix.

Model Evaluation Results

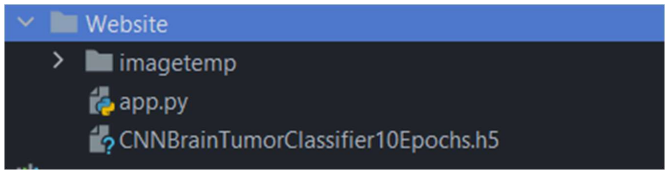
After evaluating the four models, it is concluded that CNN has the highest accuracy score. Followed by Teachable Machines, SVM and K-Means. Both Deep Learning models, CNN and Teachable Machines (Transfer Deep Learning) have achieved good classification scores. While the supervised method, SVM has reputable scores. The unsupervised method, K-Means however did not reach our classification thresholds. From the findings alone,

1. Deep Learning models work extremely well when classifying images.
2. Supervised Learning is able to classify images with respectable ability however cannot outperform the deep learning models.
3. Unsupervised learning is not fit for classifying images.
4. The Meningioma Class has the highest overall F1-Scores compared to the other classes with the machine learning models but not with the deep learning models.

CNN will our be the model of choice for implementation in the industry.

4.5 Explanation Website Code

These are the required files to make the website. The files include a python file, to write the code, a directory to store image files and the brain tumour classification model. To make this website, Python and the Streamlit API will be used to construct the website.




```
import streamlit as st
import os
from keras.preprocessing.image import load_img
import numpy as np
from keras.models import load_model
import datetime
```

These are the libraries required to create the model,

- Streamlit will be the main library used to create the website
- OS is used to attach file paths and run through the directory
- Keras is used to turn images into array and it is used to load the model
- Numpy is used to store the images that has been turned into array
- Datetime is used to get the current date and time.

```
def predict(filename , model):
    imageProcess = load_img(filename , target_size = (64 , 64))
    imageProcess = np.array(imageProcess)
    imageProcess = np.expand_dims(imageProcess, axis=0)
    result = model.predict(imageProcess)
    prob_result = np.round(result * 100, 2).max()
    class_result = np.argmax(result, axis=1)


    if class_result == 0:
        class_result = "Glioma"
    elif class_result == 1:
        class_result = "Pituitary Tumour"
    elif class_result == 2:
        class_result = "No Tumour"
    elif class_result == 3:
        class_result = "Meningioma"
```

```
    return class_result , prob_result
```

This is a method to predict the images, the method will load and turn the images into the appropriate format (array). The parameter `(filename , model)` requires a image file name and the model to predict. After the prediction, the method will return the predicted class and the accuracy of the prediction.

```
def saveUploadedFile(file):
    f = open(os.path.join("imagetemp", file.name), "wb")
    f.write(file.getbuffer())
    return st.success("Saved File:{} to imagetemp".format(file.name))
```

This is a method to save the image on to the computer. The parameter is the image file name and the returned value is a section in the website that prompts the user that the image uploaded has been saved.

```
BASE_DIR = os.path.dirname(os.path.abspath("CNNBrainTumorClassifier10Epochs.h5"))
model = load_model(os.path.join(BASE_DIR,"CNNBrainTumorClassifier10Epochs.h5"))
```

The code above uses functions from OS to get the file path and to join the file path. BASE_DIR contains the main file path to the model. (C:\Users\kelvi\PycharmProjects\MachineLearningTutorial\Website) . The keras function is used to load the model.

```
# start of the website
st.set_page_config(page_title="Brain Tumor Classification")


with st.container():
    st.header("Brain Tumor Classification")
    imagec,textc = st.columns((1,2))
    image = st.file_uploader("Upload an image",type=["png","jpg","jpeg"])
```

The code above starts the beginning of the website. `St.header` creates a header on the website. `St.columns` bis a function used to divide the width of the website, in this case the imagec only uses 1/3 of the width of the website, while textc uses 2/3. The next line uses `st.file_uploader` to create a button prompting the user to upload a file and it only accepts png, jpg and jpeg files that is within 200mb and the image uploaded will in the image variable.

```
    if image is not None:
        with imagec:
            st.image(image, use_column_width=True)
            saveUploadedFile(image)
            imagepath = os.path.join(BASE_DIR,"imagetemp")
            imagepath = os.path.join(imagepath,image.name)
            temp = st.button("Predict")
```

If an image is found in the image variable within the spaces of imagec, the image will be displayed and the image file will be saved using the method defined from above. Using OS function, the file path to the image is made. A button is created using `st.button` the result of the button is saved in temp. When the button is clicked it will return True to the temp variable

```
            with st.spinner("predicting....."):
                if temp == True:
                    classresult, probability = predict(imagepath, model)
```

After the button is clicked a loading spinning icon will appear. Then it will predict the image that is uploaded and the returned value is saved in classresult and probability.

```
                    with textc:
                        x = datetime.datetime.now()
                        st.write("report generated on",x)
                        if classresult == "Glioma" or classresult == "Pituitary
Tumour" or classresult == "Meningioma":
                            st.title("Cancer Detected")
                            st.text("Prediction = {}".format(classresult))
                        else:
                            st.title("Cancer Not Detected")


                        st.text("Confidence = {}".format(probability))
```

Within the width of textc, it will display the current date and time of when the result is generated. It will display the whether the tumor is detected and if a tumor is detected it will display the type of tumor that is detected. Then it will display the confidence of said prediction.

```
with st.container():
    st.write("Note: this is cannot be used for actual diagnosis")
```

The last line is a warning to the users of the website that this cannot be used in actual diagnosis.

In another file there is the CSS part, where the font designs and the background colour is being designed.

```
@import url('https://fonts.googleapis.com/css2?family=Roboto:wght@100&display=swap');


        html, [class*="css"]  {
        font-family: 'Verdana', sans-serif;'
        }
        body {
        background-color: #152238;
        }
```

Chapter 5: System Testing

5.1 System Test

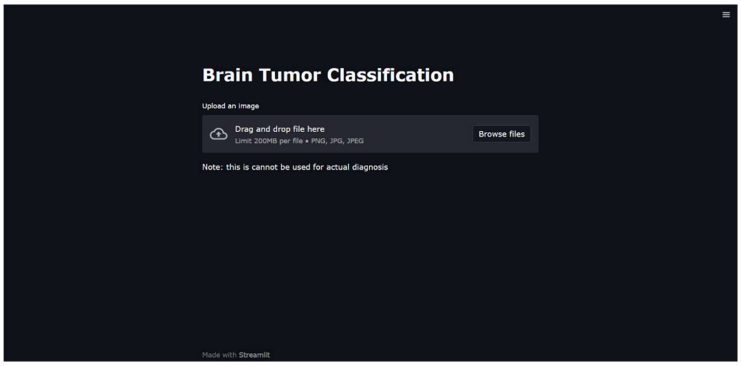


Figure 5.1 - Home Page

In Figure 5.1 shows the home page of the system where the user is able to upload any png, jpg, or jpeg file to be scanned through.

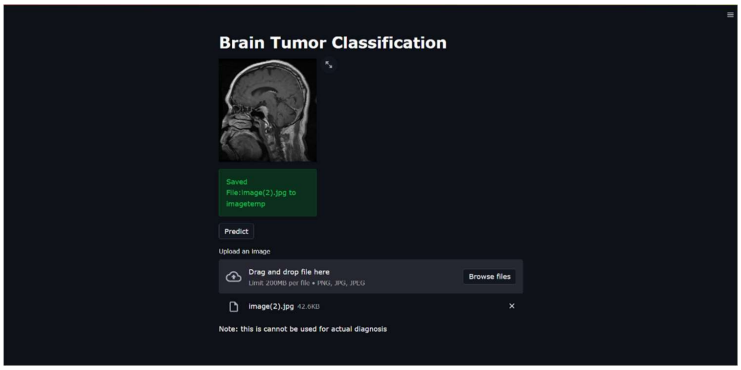


Figure 5.2 - Image with Brain Tumour Uploaded Page

After the image file has been uploaded the system will display the file name along with the image.

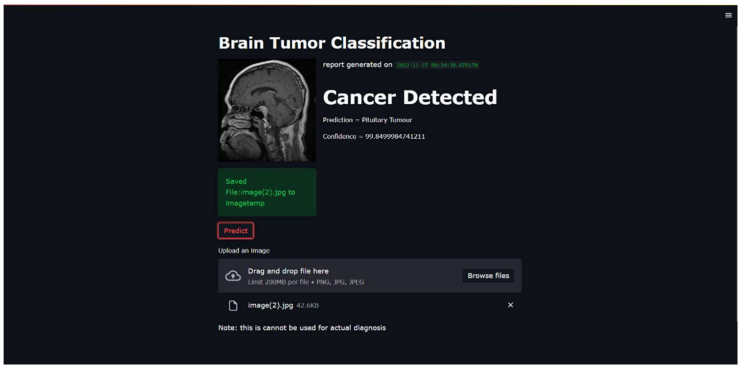


Figure 5.3 - Predicted with Pituitary Tumour Page

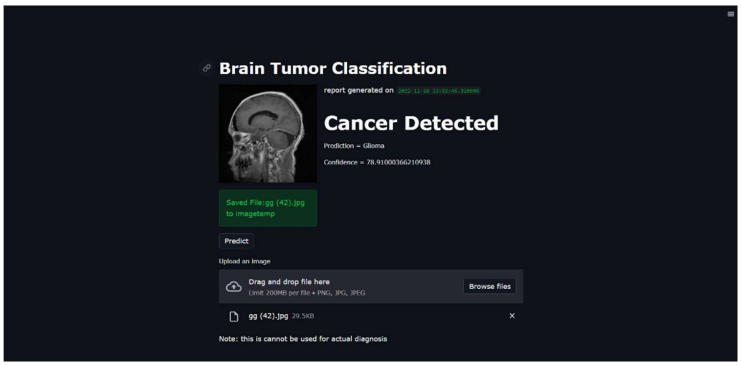


Figure 5.4 - Predicted with Glioma Page

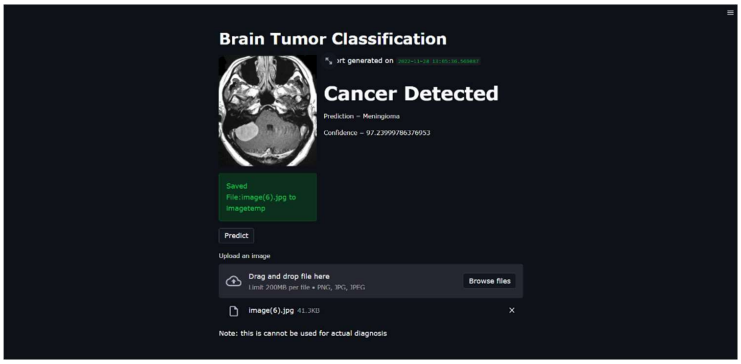


Figure 5.5 - Predicted with Meningioma Page

Once pressing predict, it will predict whether the image file consists of brain tumour or no tumour. In Figure 5.3, 5.4, & 5.5 show the image with cancer.

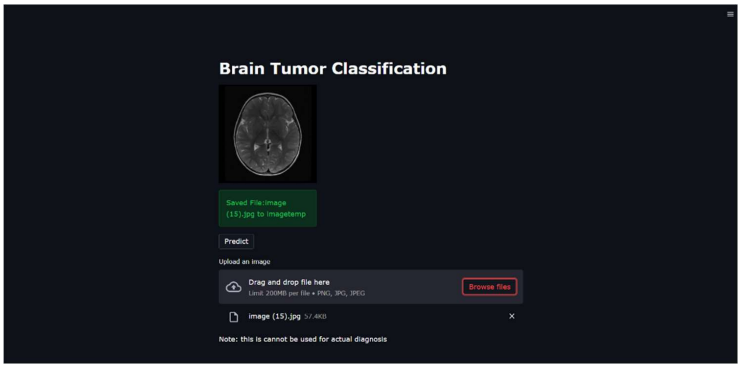


Figure 5.6 - Image without Brain Tumour Uploaded Page

In Figure 5.6 there is a brain scan which is also being uploaded to test whether it has brain cancer or not.

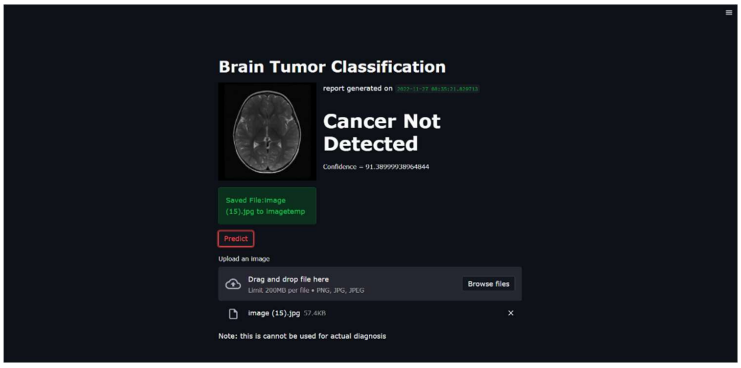


Figure 5.7 - Predicted without Tumour Page

In the Figure 5.7 it shows that there is no brain cancer detected with the confidence level at the bottom of its prediction below.

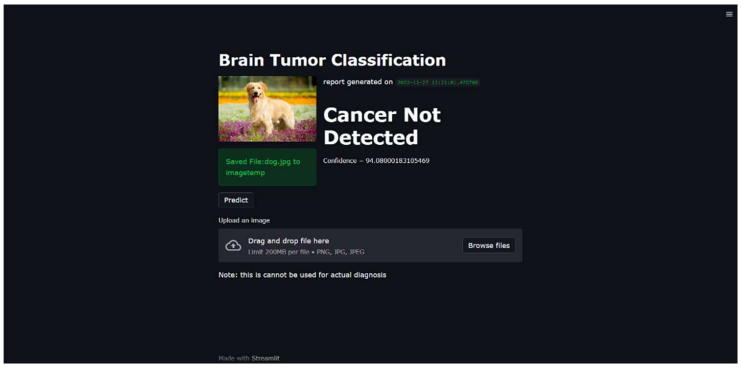


Figure 5.8 - Other Image File Scanned Page

If the user uploads an image that is not a brain scan. It is unable to differentiate, images of brain scans and foreign objects. It assumes the object of a brain scan and predicts. In this case, it shows cancer not detected.

Chapter 6: Conclusion & Recommendations

6.1 Conclusion

For the past few weeks, A study was carried by our capstone group titled Image Classification for Brain Tumour Detection using Machine Learning. An analysis was done to discover the abilities of machine learning models and its capabilities to be implemented within our real-world society. The system developed will also have a front-end for ease of access. The system will allow medical specialists to double check whether the brain scan either has the presence of brain tumour or the absence of brain tumour along with its type. By using python, all the systems were built and coded.

This study has got us plentiful of information and insights that can be applied in the real world. For example, our study concluded, Neural Networks would be the best choice of brain tumour classification. The information gained can also be used in other fields such as veterinary medicine and detection of plant/aquatic diseases.

Overall, our study has successfully produced solid distinct results. However, there are still improvements that can be done in the future.

6.2 Project Limitation

Time constraint is a major limiting factor in this project, this system is

Feature Extraction

A major flaw in our system is that it is lacking in feature extraction. Feature extraction in a project is also known as image segmentation. Feature extraction allows the extraction of region of interest. In our project, feature extraction can pinpoint the size and the location of the tumour. This can come in extremely helpful, however, it was hard to understand and to learn the implementation within the time frame. Therefore, it was not used in this project.

Cannot differentiate between brain and non-brain images

The system cannot identify if the images uploaded is a brain MRI or a foreign object. If an image of a foreign object is uploaded it will proceed as usual.

Low UI/UX Design

Due to our lack of knowledge and familiarity with StreamLit, we could not design a website with a professional looking user interface.

Model Training Accuracy Threshold

Despite being one of our goals, K-Means did not reach our threshold of 60%. After a few tries, K-Means could not get a 60% accuracy rate. A few techniques to optimise and boost its accuracy could be implemented but was out of our understanding.

6.3 Challenges in this Project

In this project, members of the group encountered challenges of their own, which included learning machine learning from scratch, learning the process of KDD/data mining, we also have to learn to use unfamiliar Python libraries such as OpenCV, Keras, Pillow, Tensorflow, Sklearn. Understanding and learning the concepts of image classification and learning also took multiple books and videos. Understanding the theory of machine learning models were especially difficult with the introduction of neurons and neural networks. Coding bugs were extremely prevalent in this project as we are not familiar with the field.

The front-end of this project faced the most issues. The first iteration of the website coded using HTML, CSS and JavaScript would not work. Flask, A popular web framework was then implemented in place of JavaScript however it has come in short. Due to the constraints of time, the website must be made with a machine learning API framework, StreamLit, for front-end proof of concept

6.4 Future Recommendation

For the analysis, reinforced learning can be included when comparing the capability for image classification. It is left out in this project, as we have time constraints. In the future, reinforced deep learning can be trained and utilised in tasks like medical predictions etc. The uniqueness of the human body is unpredictable which means the brain tumours can come in all shapes and sizes. Through the utilisation of reinforced learning, A better classification model could be achieved.

In the future, the system should be available as an application with a complex GUI to pass all the information. There are several factors that can be improved to improve the current system in this project. This system should be evaluated by actual medical expert/personnel for further insight and improvements.

One of the factors we can improve is the accuracy of the model. To do that increasing the amount of training data used to train the model can have a positive impact on the accuracy of the model. Unfortunately, due to the amount of data available online and the limitations of our machine, that could not be achieved with the scope of this project.

Due to the rise of Computer Vision technology, Its capabilities can be adapted to further improve our system. First of all, is feature extraction, Computer Vision or CV is capable of extracting “regions of interest” on the image. This is also called, image segmentation.

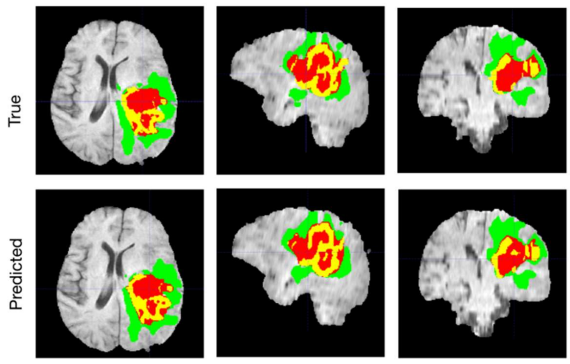


Figure 6.4.1 – Segmentation of Brain Tumours using AI by NVIDIA

The brain images also can be measured via CV technology where an accurate measurement of the region of interest is extracted. Information like these can give a good idea to medical professional of the stage and progression of the cancer.

This system also can have a near-to real time prediction which helps medical professional in the field such as paramedics or home-visitation medical services.

References

Delua, J. (2021). *Supervised vs. unsupervised learning: What's the difference?* [online] IBM. Available at: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning [Accessed 29 Sep. 2022].

Mishra, U. (2021). *Binary and Multi-class Classification in Machine Learning | Analytics Steps*. [online] www.analyticssteps.com. Available at: https://www.analyticssteps.com/blogs/binary-and-multiclass-classification-machine-learning [Accessed 27 Sep. 2022].

IBM Cloud Education (2020). *What is Unsupervised Learning?* [online] www.ibm.com. Available at: https://www.ibm.com/cloud/learn/unsupervised-learning [Accessed 29 Sep. 2022].

www.javatpoint.com. (n.d.). *Supervised Machine learning - Javatpoint*. [online] Available at: https://www.javatpoint.com/supervised-machine-learning [Accessed 29 Sep. 2022].

Tim (2019). *SVM P.2 - How Does A SVM Work?* [online] techwithtim.net. Available at: https://www.techwithtim.net/tutorials/machine-learning-python/svm-2/ [Accessed 30 Sep. 2022].

www.tutorialspoint.com. (n.d.). *ML - Support Vector Machine(SVM) - Tutorialspoin*t. [online] Available at: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_support_vector_machine.htm [Accessed 1 Oct. 2022].

riturajsaha (2021). *Image Classification using Google's Teachable Machine*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/image-classification-using-googles-teachable-machine/ [Accessed 2 Oct. 2022].

Landman, E. (2022). *Finding Duplicate Images with Python*. [online] Medium. Available at: https://towardsdatascience.com/finding-duplicate-images-with-python-71c04ec8051 [Accessed 8 Oct. 2022].

Khan, M. (2017). *KMeans Clustering for Classification*. [online] Medium. Available at: https://towardsdatascience.com/kmeans-clustering-for-classification-74b992405d0a#:~:text=In%20the%20first%20attempt%20only [Accessed 15 Oct. 2022].

Rastogi, S. (2021). *K-Means Clustering and Transfer Learning for Image Classification*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/06/k-means-clustering-and-transfer-learning-for-image-classification/ [Accessed 15 Oct. 2022].

Insight, A. (2018). *An introduction to Reinforcement Learning*. *YouTube*. Available at: https://www.youtube.com/watch?v=JgvyzIkgxF0 [Accessed 26 Nov. 2022].

Błażej Osiński (2018). *What is reinforcement learning? The complete guide - deepsense.ai*. [online] deepsense.ai. Available at: https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/ [Accessed 26 Nov. 2022].

A Combined Deep CNN-LSTM Network for the Detection of Novel Coronavirus (COVID-19) Using X-ray Images - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-typical-architecture-of-the-convolutional-neural-network_fig1_342345782 [accessed 3 Oct, 2022]

An Efficient Alert Aggregation Method Based on Conditional Rough Entropy and Knowledge Granularity - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Scatter-diagram-of-the-main-process-of-the-K-Means-clustering-algorithm-a-k-center_fig2_339891265 [accessed 4 Oct, 2022]

IBM (2020). *What are Convolutional Neural Networks?* [online] www.ibm.com. Available at: https://www.ibm.com/cloud/learn/convolutional-neural-networks [Accessed 3 Oct. 2022].

Jeffares, A. (2019). *K-means: A Complete Introduction*. [online] Medium. Available at: https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c [Accessed 4 Oct. 2022].

Kholsa, S. (2019). *CNN | Introduction to Pooling Layer*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/.

Mishra, M. (2020). *Convolutional Neural Networks, Explained*. [online] Medium. Available at: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939 [Accessed 2 Oct. 2022].

Monitoring Online Tests through Data Visualization - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-Steps-of-a-KDD-process_fig7_220073492 [accessed 5 Oct, 2022]

T, B. (2021). *Comprehensive Guide on Multiclass Classification Metrics*. [online] Medium. Available at: https://towardsdatascience.com/comprehensive-guide-on-multiclass-classification-metrics-af94cfb83fbd [Accessed 9 Oct. 2022].

A New Model for Brain Tumor Detection Using Esemble Transfer Learning and Quantum Variational Classifier. [online] Available at: https://www.researchgate.net/publication/359968682_A_New_Model_for_Brain_Tumor_Detection_Using_Ensemble_Transfer_Learning_and_Quantum_Variational_Classifier [Accessed 11 Oct, 2022]

Explaination-Driven Deep Learning Model for Prediction of Brain Tumour Status Using MRI Image Data. [online] Available at: https://www.frontiersin.org/articles/10.3389/fgene.2022.822666/full [Accessed 11 Oct, 2022]

Brain tumour detection from MRI images using deep learning technique. [online] Available at: https://iopscience.iop.org/article/10.1088/1757-899X/1055/1/012115 [Accessed 14 Oct, 2022]

Brain Tumour Detection Using Deep Learning. [online] Avaliable at: https://ieeexplore.ieee.org/document/9445185/keywords#keywords [Accessed 15 Oct, 2022]

Magnetic resonance imaging and computed tomography in emergency assessment of patients with suspected acute stroke: a prospective comparison. [online] Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1859855/ [Accessed 15 Oct 2022]

Image References

https://static.javatpoint.com/tutorial/machine-learning/images/supervised-machine-learning.png

https://static.javatpoint.com/tutorial/machine-learning/images/unsupervised-machine-learning-1.png

https://www.techwithtim.net/wp-content/uploads/2019/01/svm9.png

https://media.geeksforgeeks.org/wp-content/uploads/20201129131914/20200124919183p27134-660x309.png

https://media.geeksforgeeks.org/wp-content/uploads/20201129130604/Screenshot85.png

https://iopscience.iop.org/article/10.1088/1757-899X/1055/1/012115

https://www.frontiersin.org/articles/10.3389/fgene.2022.822666/full

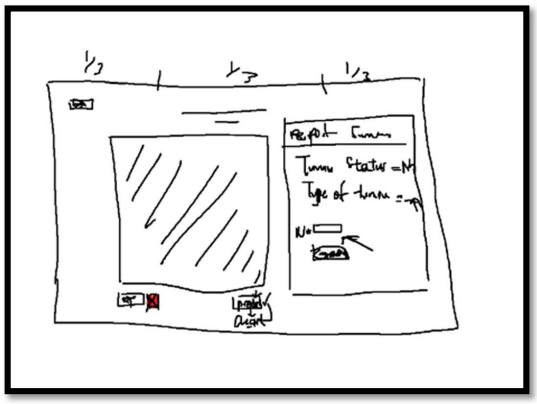
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9023211/

https://iopscience.iop.org/article/10.1088/1742-6596/1937/1/012008

https://miro.medium.com/max/1400/1*Y0TDuXNyywjqqr5l5GkMQQ.png

Appendices

Appendix i. Original Front End UI Design.



Appendix ii. Tasks to be completed by team

```
1. Types of Models to be Tested

    Machine Learning
        Supervised
            SVM(Support Vector Machine)
        Unsupervised Learning
            K-Means

    Transfer Learning

        TeachableMachines
        VGG-16

    Deep Learning
        Convolutional Neural Network

3. Acquire Image Classification Confidence if possible

2. Experiments

    1. Accuracy of Models. (BarChart)
    2. tested against size of dataset vs accuracy. (ScatterPlot)
    3. get number of images each folder and compare numbers.


3. Goals of Webapp
    Able to display Image
    Generate Text Report
    Show Test Conditions for eg:
        Cancer detected...
        globliastoma detected..
    Upload File
    *Run with Python*
```