Analysis of radiogenomic data by neural networks for tumor detection in the body

Summary

A tumor is a common type of cancer. It can manifest itself in the body of a person in two cases, benign and malignant. How does it enter the body and what can it do in the body?! We will give radiogenomic analysis. In this research, our aim is to analyze tumor data by radiogenomic with the help of neural network. Radiogenomics can understand the type of cancer and how far it has progressed in the human body by analyzing data from the patient's MRI. We are going to take this data to the prediction stage with the help of neural network (ANN). Since Radiogenomic is not able to regression the data, and it cannot give us a more efficient prediction in the field of the resulting tumor in the body. We have to consider a training model for this work. We to have better results with high quality of data with this training model. In this model, which gives us most of the states 0 and 1, the closer we are to stage 0, in this case, our tumor will move towards the benign state, and the more it moves towards 1, the state it will be malignant.

But it should also be noted that radiogenomic and neural networks have a close relationship. And that both of them use ML to analyze and predict their experimental data. And this proximity provides us with the connection between these two directions in our use of radiogenomic data by neural networks.

Definitions:

1. Radiogenomics:

1.1: Radiogenomics refers to the analytical process, which aims to link cancer imaging features with genomic data.

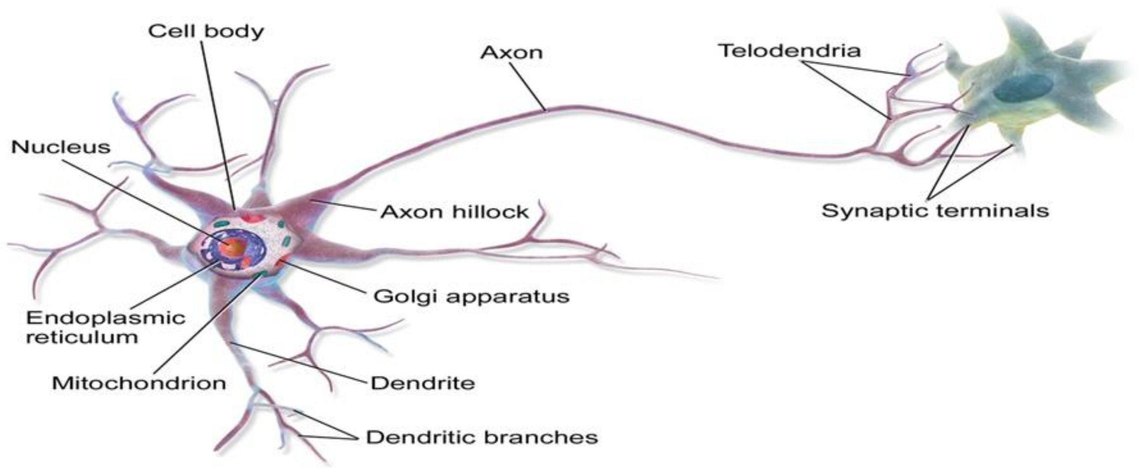
1.2: Radiogenomic represents the evolution related to radiology-pathology from the tissue level to the subcellular level.

1.3: Radiogenomic, in its simplest form, can be said to refer to the relationship between imaging features of a disease and its gene, expression patterns, gene mutations and other features related to the patient's gene.

In this section, we present three different and comprehensible definitions of radiogenomic. The point of similarity in these three definitions is the patient's gene. All three definitions emphasize the genome resulting from the tumor disease in the individual's body. So the turning point in the calculations related to Radiogenomic data will be related to the genomicity of the individual. Of course, we intend to provide the definitions related to the neural network in this part of the research.

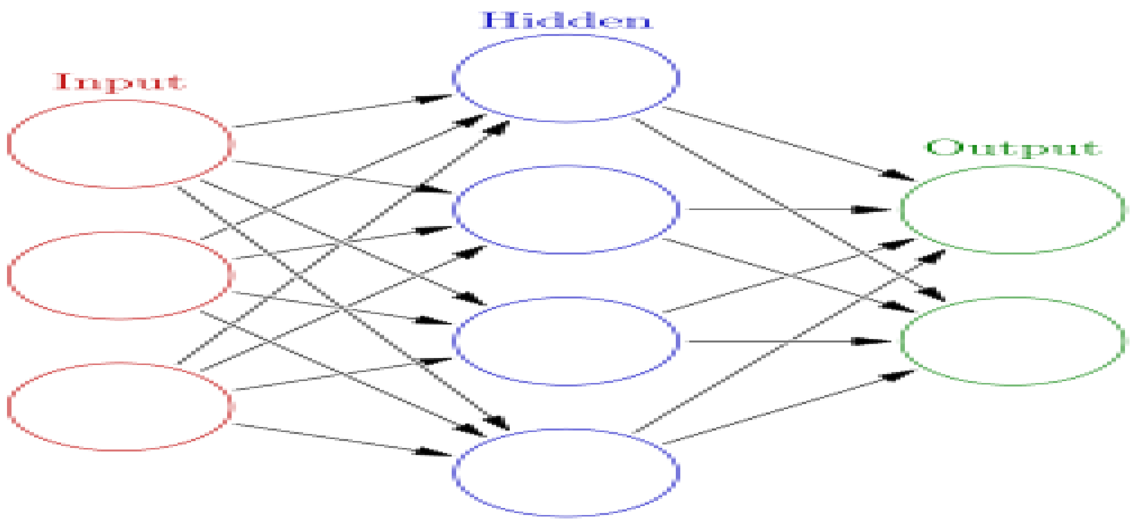
2. Neural networks (NN):

2.1: A neural networks is a set of interconnected neurons that are placed in the body to perform certain operations.



The figure related to the connection of neurons

2.2: Neural networks are one or more layers of neurons stacked on each other. Each neuron's output is another neuron's input.



The figure related to the layers of the neural network

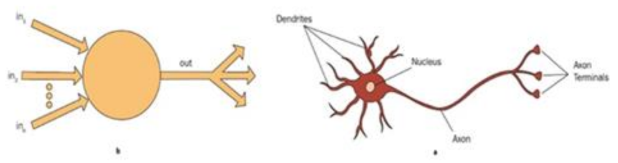
2.3: Neural networks are composed of neurons communicating with each other to coordinate other body parts with each other.

3. Artificial neural networks (ANN):

3.1: Artificial neural networks are a type of network inspired by the biological brain of living organisms, especially humans.

3.2: Artificial neural networks, consisting of a type of neuron and node with net weights, are suitable for forming the output.

3.3: Neural network refers to a set of sub-branches of artificial intelligence derived from the human biological brain, which is usually based on calculations.

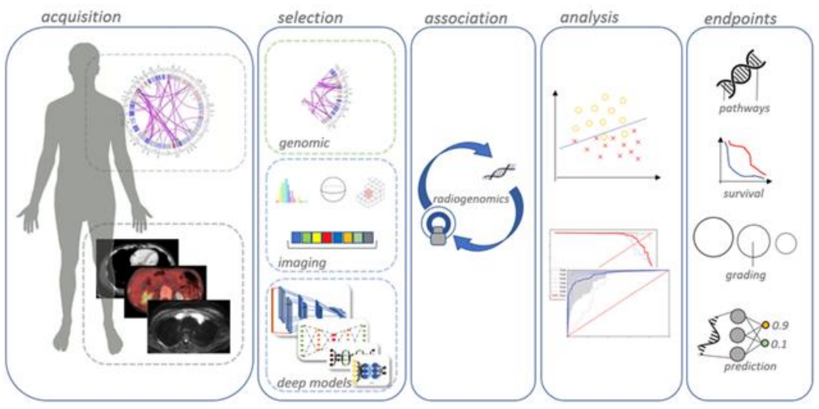


In this section, we discussed the definitions of three important branches of our research. That these three branches will play a key role in this research. In order to use our artificial network for experimental results, we must know the important structure and function of the human neural network and understand how it works. In the following, we will The concepts and examples of a person's tumor and investigate the causes of a patient's tumor with the help of analyzing Radiogenomics data by neural network discuss.

Basic concepts:

<< Radiogenomic:

Radiogenomic can be used more in the fields of imaging disease genes for cancer treatment. It is mostly related to genes and gene patterns in cancer patients, which in its simplest form can be called (genomic features) or (genomics). Radiogenomic is also a type of radiation therapy imaging to improve and treat cancer. With the difference that in this imaging it analyzes the genes of the disease. In radiogenomic, it also predicts the initial results with available information. Imaging information that gives us the results of genomic completion and the state of the tumor inside the patient's body. Many engineers in the field of medical science consider radiogenomic as an emerging and new operating system with high efficiency in order to better identify cancer at the molecular level, allowing early detection of cancer in a patient. Between imaging, it expresses genomics and clinical knowledge according to the data regardless of any general interpretation. It can be said that radiogenomic in a casual way can talk with the gene data taken from the patient. In principle, radio genomics is much more efficient for reducing radiation therapy from the aspects of damage it brings to the body. It puts the person in a normal state after chemotherapy. And it causes rapid recovery after the completion of radiation therapy due to chemotherapy.



Radiogenomic figure

In this form, the image from the initial stage taken by MRI is analyzed by radiogenomic data. As the image shows, they are analyzed as data and put into a recognizable pattern to predict the results of a disease or genomic.

>> Neural networks:

Neural networks are a method to predict through the available data. They can provide their prediction through a model that we define ourselves. This prediction through ML, which is a separate discussion. and related to ANN gives the neural network.

Neural networks should be considered as boxes related to neurons, because each neural network is made up of several neurons, which are placed together layer by layer.

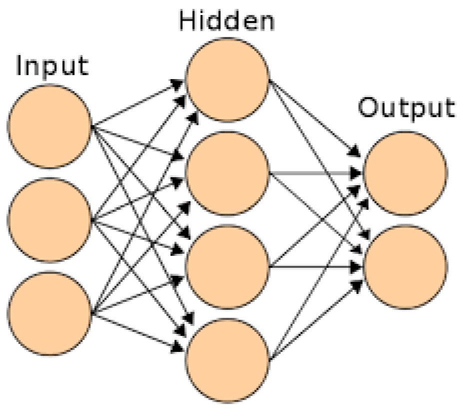
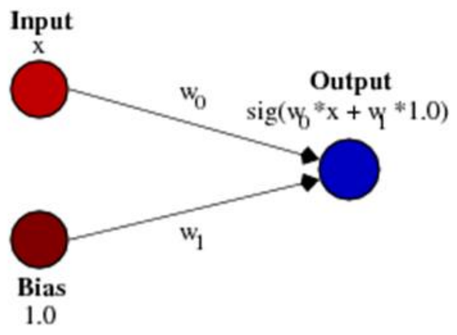


Figure of a simple neural network

This is a simple form of neural network. It has three input, hidden and output layers.

The input layer has inputs from 1 to n. In this neural network, after entering the network, the input data enters the first hidden layer called weights, and a weight is assigned to each input based on its value. After the weighting stage Inputs, bias is responsible for processing the input data and analyzes them and provides an output according to our model. This output will be the prediction we want. The bias layer plays a key role in data analysis. Any data that is entered is analyzed, then clustered or calculated. In each case, it considers or prepares the data from the raw state to a usable state for output.

Total related to the bias layer



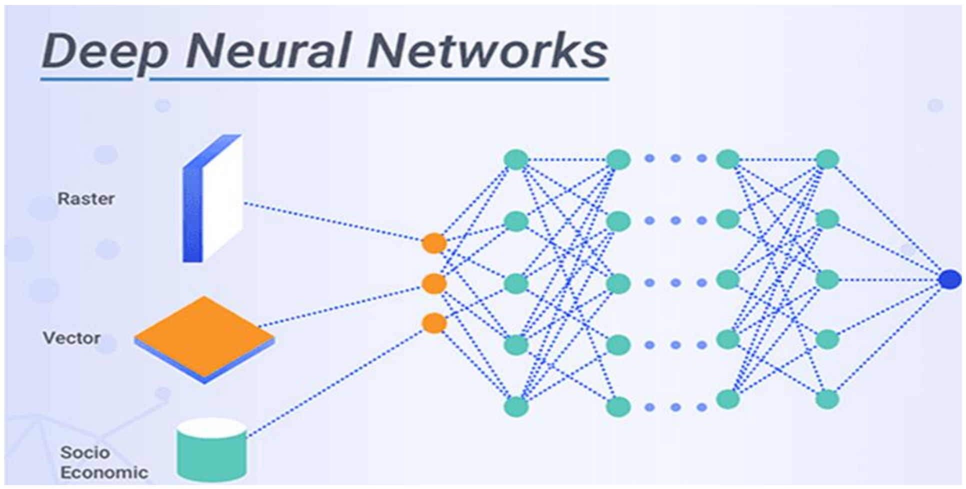
In this figure, we have an input named x and a bias that performs a process on the input x. As it was said, the bias is a processor on the inputs. In this simple network, it performs a series of operations with the help of the sigmoid function and the result is sends to the output.

In addition to training data in neural networks, we can also use DNN, which means deep learning.

In this method, ML considers a pattern for ANN through Datasets to force our DNN to make a suitable decision or prediction based on this pattern.

We are looking at a deep learning neural network. In this image, as you can see, we have three inputs: raster, vector, and free. These three inputs look a bit complicated, because they are not of the same data type. Rather, it must be analyzed from several types of data in this network. That is, our bias is facing a variety of data.

Figure of a DNN

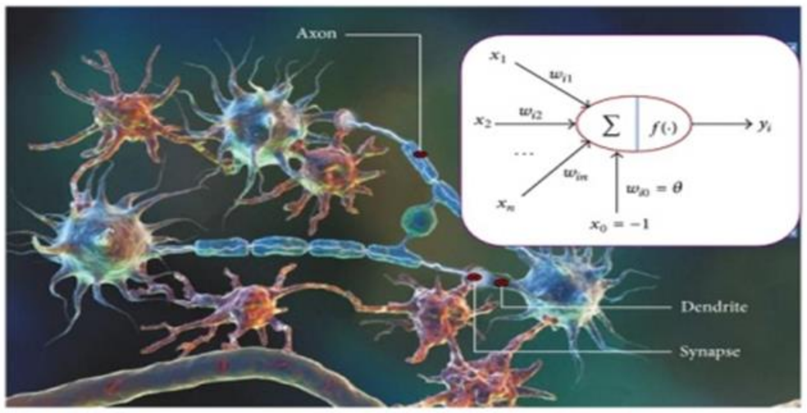


First, data must be separated and it can perform its analysis according to the type of data.

The neural network can be considered a human biological brain network. A neuron inside this network acts as a logical mathematical function. It considers all the data as a fuzzy logic. And it decides and acts according to this fuzzy logic. It means that it considers the codes 0 and 1 for itself and based on these codes, it is always on or off.

Sometimes, or even most of the time, we deal with real data from the real world, and our networks should be able to analyze this volume of data faster and more cost-effectively. we must provide networks that can provide forecast with better quality for us with forecasting stock market risk, better weather forecasting, cancer diagnosis, tumor, etc.

Figure related to the biological network of the brain



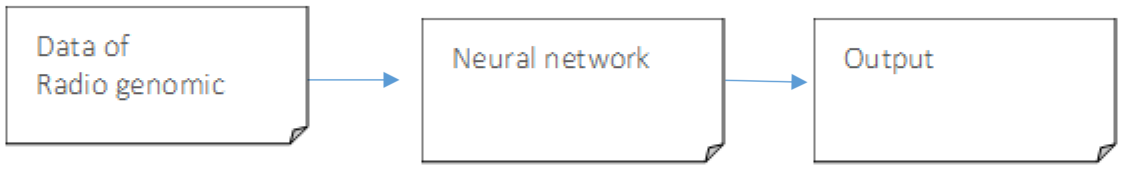
This image is related to the biological of the human brain. The function is considered for this network, which performs a series of operations on this data.

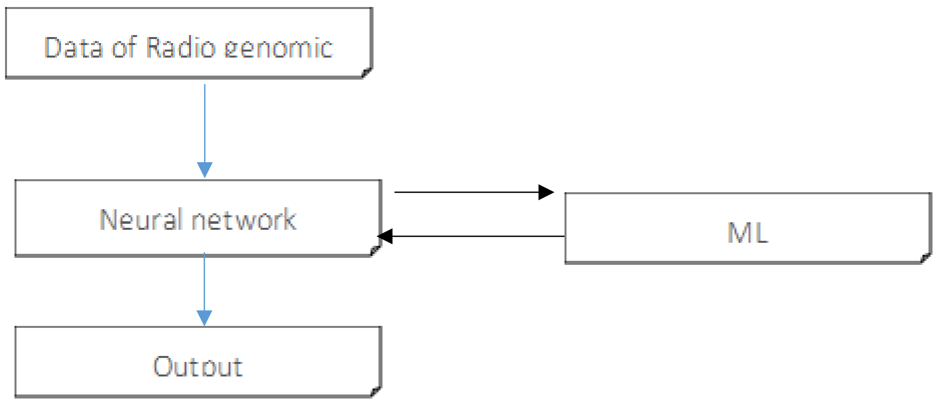
Target:

Purpose of neural networks:

Neural networks are called an intelligent computer network because they can make an intelligent decision based on the criteria they have from the data. This intelligent decision is sometimes supervised and sometimes unsupervised.

For example, in tumor diagnosis, we present the data obtained from radio genomics to the neural network. Before analyzing the data, the neural network examines the data with the help of ML, which has a pattern for itself. Reads



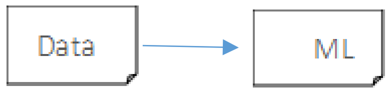


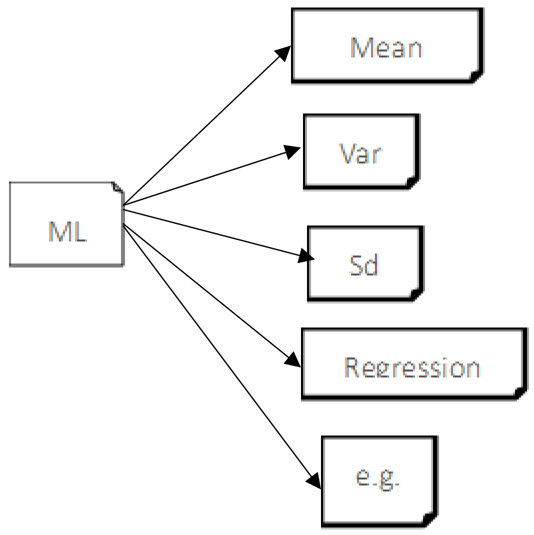
As you can see in the flowchart related to this process, after entering the ANN and before the output, the data enters the Ml and takes a pattern and enters the ANN again for processing.

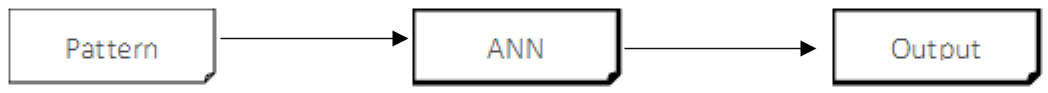
So, the first goal of our study is ML. That means we have to consider a series of models for our neural network before dealing with data processing.

We should always try to ensure that our model provides a high-quality plan of predicting the data results. And for this, it requires that we always have data with ideal weights, that is, in the data collection phase. Let's use the best algorithms.

The main goal of ANN is a prediction with a certain probability of the data in any image or object that exists. When our ML can be better, neural networks can provide a more accurate study of ML. In some cases, we need to analyze large data. In this case, we cannot work only by analyzing a series of photos or objects. At this stage, we will deal with random data in wide ranges. Big data will need accurate clustering, so that a suitable data pattern can be created from them for the pattern used in the neural network. Of ML, that is, we must use algorithms in ML. In fact, ML takes all the raw data. Our intended ML provides us with a specific pattern with the help of a series of functions including Mean, Var, Sd, Regression and other functions.







After our model is extracted, it enters the neural network stage to predict the results or a desired output.

The purpose of all these steps is to create a suitable output. An output that can turn chance into a possibility of certainty.

Both reduce the risk of radiation therapy in some medical operations and reduce the percentage of error resulting from human vision and perception.

Radiogenomic:

The main goal of radiogenomic can be considered to help cure cancer in people suffering from it. Also, the diagnosis of this disease was introduced with the help of genetic data and computer imaging. Since radiation therapy causes dangerous side effects in patients and even in Most of them do not cure cancer. Radiogenomic tries to analyze the characteristics and radiological appearance and the relationship between the genome of the tumor. The question that arises in the relationship between genomes is: What is the practical importance of radiogenomic discoveries?

To answer this question, radiogenomic works with the results of imaging. The results obtained from the early stages, such as: the outlook of a patient with cancer, are among these results. Also, the main goal of radiogenomic can be considered as predicting the patient's condition. That is, with the help of the available results and by examining the results with the help of computer imaging, it informs the treating doctor of the rate of recovery or the rate of cancer progression.

The purpose of examining the results of radiogenomic data:

1. It tells the doctor how far the cancer has progressed.

Of course, it should be said that radiation therapy also works like this. However, radiation therapy causes serious damage to a person's genetic cells and creates a new type of toxicity in the body. But radiogenomic is not like that and it does this without chemical intervention.

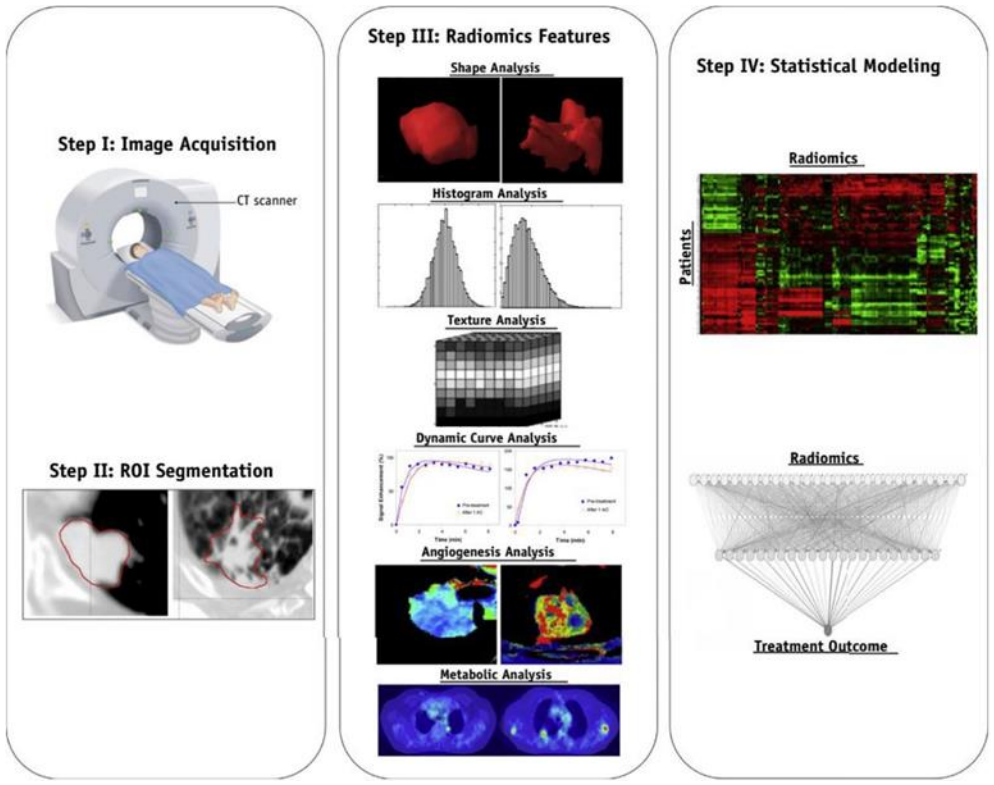


Figure related to cancer investigation by radio genomics

In this form, MRI imaging is done first. Then it is converted into computer data for analysis by radiogenomic, so that the doctor can find out the progress of cancer with the help of ML.

2. It identifies the exact point of cancer.

The main problem in the field of medicine can be seen in the accurate identification of cancer. We cannot understand the primary and exact location of cancer without the help of radiogenomic. Only with MRI imaging can we identify the location of cancer activity.

3. It provides us with detailed information about cancer.

Analysis of radiogenomic data with neural networks:

It is only with radiogenomic that we can turn MRI and C-TScan images into experimental data. But it should be noted that when analyzing the data, we will often have more dispersion than the standard average. Because will not have management taken place. And our data will be more scattered.

The lack of data management in Radiogenomic causes the data dispersion to be higher than our standard average. The higher the dispersion, the higher the quality prediction for ours data cannot with be considered.

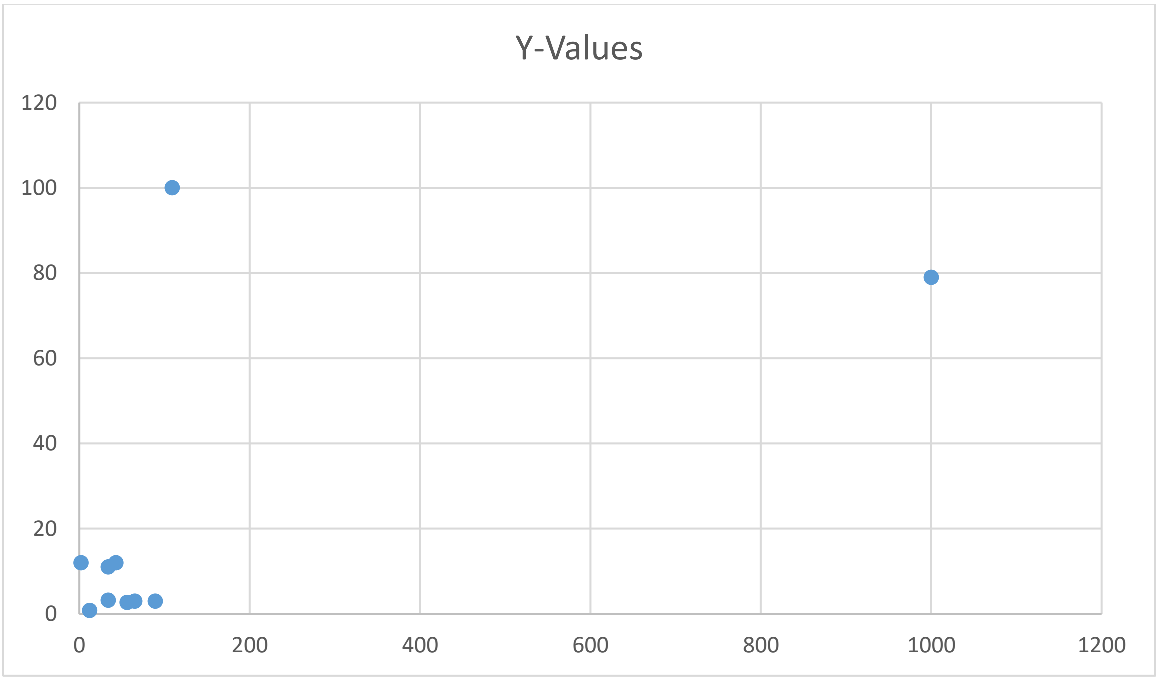


Diagram of a Radio genomics data

In this graph, as you can see, we don't see a relatively good data dispersion. Our prediction will face a high percentage of error. To solve this problem, we are looking for the design of an ANN in this project so that we can analyze this data after analysis. Convert usable data in ML to an ANN.

It should be noted that the reason we use ANN is that we don't want our data to be far away from the average of our criteria. Because the further the data goes, then it will move away from the state of 0 and 1. It should to be close to 0 and 1 no matter what.

In fact, after creating a data analysis with the help of Radiogenomic, we will convert them into an understandable pattern for an ANN. so that we can use a linear diagram to convert them into an educational data for teaching the diagnosis of viral infection of the patient in question.

In this project, we have considered the MRI image of four brain regions L1, L2, L3, and L4. In this image, we have analyzed the data taken by Radiogenomic as follows.

L1 = 471

L2 = 30

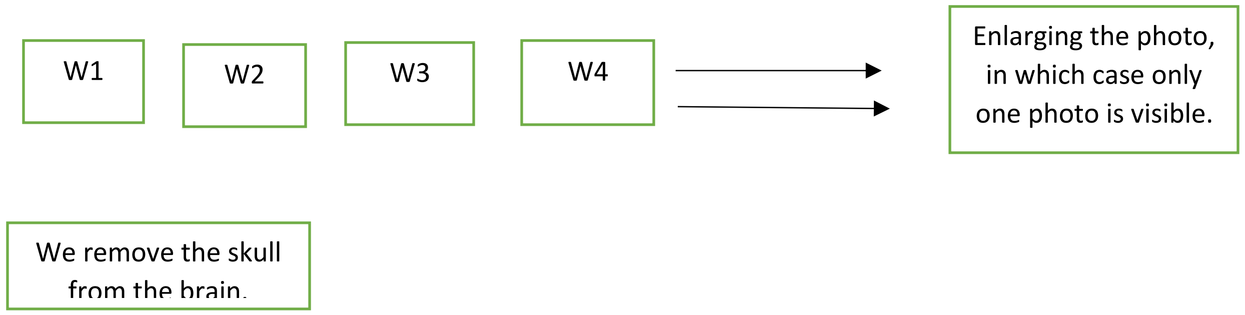
L3 = 500

L4 = 845

Image cropping: 10 crops are used for each submitted image. After this step, we increase the size of the image a little. We will remove the skull from the brain. So that we can examine the tumor better.

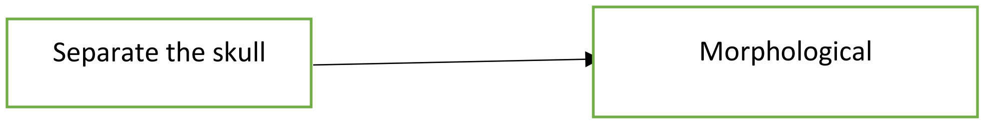
To get better data. By cutting the skull, we actually want to convert the MRI image into data with certain weights.

We have four layers of the photo.



W1
W2
W3
W4
Enlarging the photo, in which case only one photo is visible.
We remove the skull from the brain.

After the only remaining brain, we perform morphological operations on the resulting image of the brain. We aim the whole skull at the brain, then we to explore objects on the brain. (Here the objects are the tumor.) Then, after finding the tumor (if there is one), we put the brain on it.



After the Morphological operation, we extract data from them, such as (mean standard, standard deviation, and properties). In this project, we will extract data features through python codes due to its simplicity and expandability. We have four images or four input data. Well, with Python code, we calculate their average criteria as below.

```
Import numpy as np

datas = [471, 30, 500 , 845]

data_mean = np.mean(datas)

print (data_mean)
```

By running the code, our average criterion will be equal to 461.5.

To calculate the std or standard deviation of four photos with specific data, we used the following Python code for this task.

```
Import numpy as np

datas = [471,30,500,845]

std_data = np.std(datas)

print (std_data)
```

The standard deviation of our MRI imaging in this calculation was equal to 289.325.

For us, the amount of data dispersion is very important. Because with this data dispersion, we want to find out whether the brain tumor is benign or malignant. So, in this case, we also calculate the amount of data variance.

```
Import numpy as np

datas = [471,30,500,845]

var_data = np.var(datas)

print (var_data)
```

Well when we calculated the standard deviation the result gave us a value of 83709.25.

Now, with the help of the standard mean and standard deviation of our experimental data, we extract a high-quality prediction.

```
Import numpy as np

datas = np.random.uniform(461.5,289.325,4)

print (datas)
```

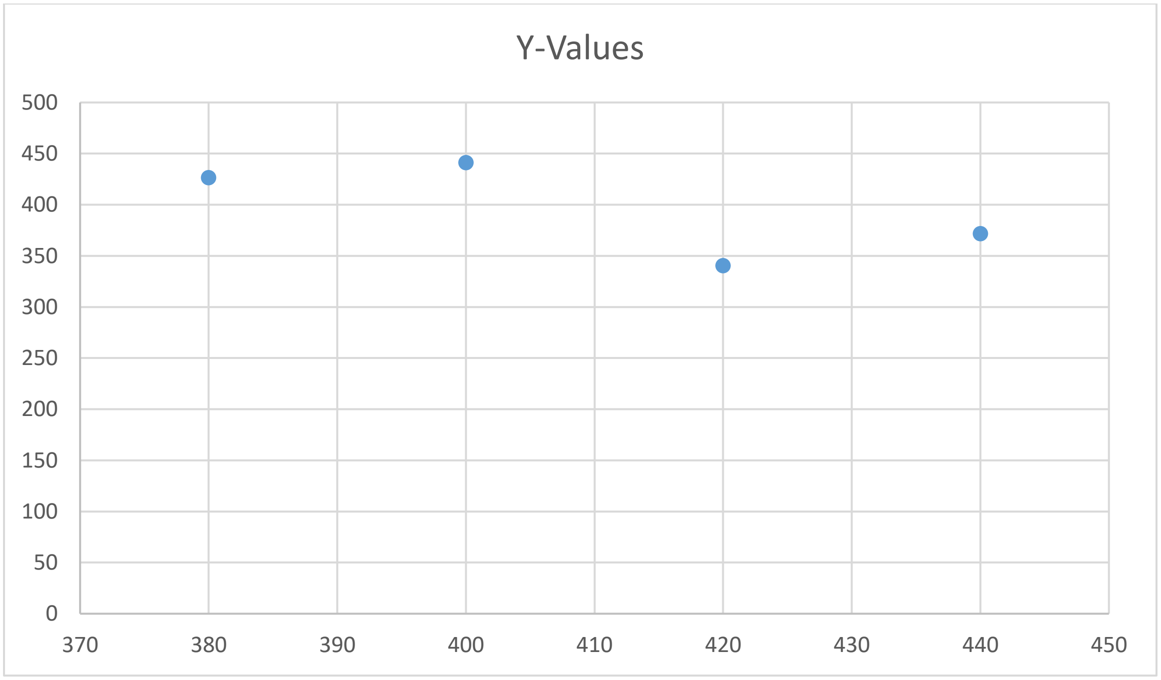
Our resulting data will be in the form of a one-dimensional array as follows.

```
[370.5129, 340.29, 441.025, 426.47]
```

We can now plot the dispersion of our data with the help of Histogram. We will have an initial raw data of the image, and another data called test. Now we draw this distribution with the help of Python.

```
Import numpy as np

Import matplotlib.pyplot as plt

data_raw = [471 , 30 , 500 , 845]

 data_test = [371.5129 , 340.29 , 441.025 , 426.47]

plt.scatter(data_raw , data_test)

plt.show()
```

The general shape of the diagram resulting from the program for the dispersion of experimental data will be as follows.



It shows the amount of dispersion of the data on the graph. It shows more dispersion upwards. And it shows us the news of a high malignant tumor.

Now we can consider an average criterion for each of our data compared to the experimental data. If we consider this for our data, our table will be complete as below.

input_raw	input_test	mean	std	var
471	371.5129	421.25645	49.74355	2474.4207
30	340.29	185.145	155.145	24069.971
500	441.025	470.645	29.354	861.716
845	426.47	637.235	207.765	43166.2952

In this table, we calculated the properties and characteristics of each image according to the criteria of raw and experimental data to compare the properties of each photo with the general properties of the tests.

If we check the properties between the raw and experimental data with our whole data, we will see that the amount of dispersion will be much higher than normal.

We created a suitable criterion for our neural network model by examining the generalities. We saw that we obtained a very high dispersion. If we draw a regression line for these axes, we will see that our dispersion value will be higher than expected.

Now, after we have calculated the properties between the experimental and raw data, in general, we get an intermediate value of the properties.

The benchmark median for our data set:

```
Import numpy as np

data_mean = [421.25645 , 185.145 , 470.645 , 637.235]

data_std = [49.74355 , 155.145 , 29.354 , 207.765]

data_var = [2474.4207 , 24069.971 , 861.716 , 43166.2952]

data_mean_median = np.median(data_mean)

print ("data_mean_median = " , data_mean_median)

data_std_median = np.median(data_std)

print ("data_std_median =" , data_std_median)

data_var_median = np.median(data_var)

print ("data_var_median = " , data_var_median)
```

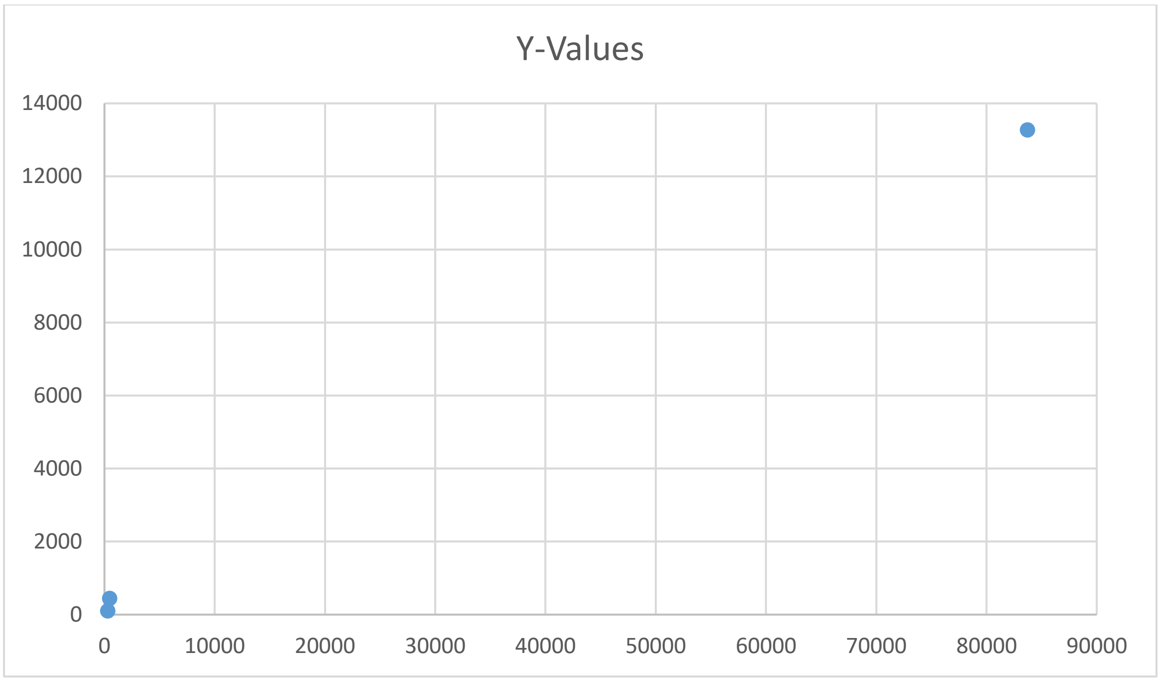
After we have calculated the result of all medians, we will calculate the amount of dispersion with the general properties of the result.

```
data_mean_median = 445.95

data_std_median = 102.44

data_var_median = 13272.195
```

Now that we have calculated the properties of two data_raw and data_test, now we consider data properties in the general state of a scatter and prediction for our results. That is, we will consider a specific pattern for ANN design.

```
import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

feature_all = [461.5 , 289.325 , 83709.25]

feature_raw_test = [445.95 , 102.44 , 13272.195]

slope , intercept , r , p , std_err = stats.linregression(feature_all ,
feature_raw_test)

def myfunc(feature_all):

        return slope * feature_all + intercept

mymodel = list(map(myfunc , feature_all))

plt.scatter(feature_all , feature_raw_test)

plt.plot(feature_all , mymodel)

plt.show()
```

The diagram shows an upward equilibrium state representing the tumor in an advanced and malignant state.



Now that we have separated our data and considered a suitable model for our ANN, we go to our neural network. In this research, we have tried to use the from to scratcher mode.

```
import numpy as np

from scipy.special import expit as activation_function

from scipy.stats import truncnorm

def truncated_normal(mean=0, sd=1, low=0, upp=10:

    return truncnorm(low - mean) / sd, (upp - mean) / sd, loc=mean,
scale=sd)

class Nnetwork:
```

```
def __init__(self,

            no_of_in_nodes1,

            no_of_in_nodes2,

            no_of_in_nodes3,

            no_of_in_nodes4,

            no_of_in_nodes5,

            no_of_in_nodes6,

            no_of_out_nodes1,

            no_of_out_nodes2,

            no_of_out_nodes3,

            no_of_out_nodes4,

            no_of_out_nodes5,

            no_of_out_nodes6,

            no_of_hidden_nodes1,

            no_of_hidden_nodes2,

            no_of_hidden_nodes3,

            no_of_hidden_nodes4,

            no_of_hidden_nodes5,
            no_of_hidden_nodes6,
            learning_rate):
   self.no_of_in_nodes1 = no_of_in_nodes1
       self.no_of_in_nodes2 = no_of_in_nodes2
       self.no_of_in_nodes3 = no_of_in_nodes3
       self.no_of_in_nodes4 = no_of_in_nodes4
       self.no_of_in_nodes5 = no_of_in_nodes5
       self.no_of_in_nodes6 = no_of_in_nodes6
       self.no_of_out_nodes1 = no_of_out_nodes1
       self.no_of_out_nodes2 = no_of_out_nodes2
       self.no_of_out_nodes3 = no_of_out_nodes3
       self.no_of_out_nodes4 = no_of_out_nodes4
       self.no_of_out_nodes5 = no_of_out_nodes5
       self.no_of_out_nodes6 = no_of_out_nodes6
       self.no_of_hidden_nodes1 = no_of_hidden_nodes1
       self.no_of_hidden_nodes2 = no_of_hidden_nodes2
       self.no_of_hidden_nodes3 = no_of_hidden_nodes3
       self.no_of_hidden_nodes4 = no_of_hidden_nodes4
       self.no_of_hidden_nodes5 = no_of_hidden_nodes5
       self.no_of_hidden_nodes6 = no_of_hidden_nodes6
       self.learning_rate = learning_rate
       self.create_weight_matrices()

   def create_weight_matrices(self):
       rad = 1 / np.sqrt(self.no_of_in_nodes1)
       X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
       self.weights_in_hidden1 = X.rvs((self.no_of_hidden_nodes1,
                                       self.no_of_in_nodes1))
       rad = 1 / np.sqrt(self.no_of_in_nodes2)
       X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
       self.weights_in_hidden2 = X.rvs((self.no_of_hidden_nodes2,
                                       self.no_of_in_nodes2))
       rad = 1 / np.sqrt(self.no_of_in_nodes3)
       X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
       self.weights_in_hidden3 = X.rvs((self.no_of_hidden_nodes3,
                                       self.no_of_in_nodes3))
       rad = 1 / np.sqrt(self.no_of_in_nodes4)

       X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)

       self.weights_in_hidden4 = X.rvs((self.no_of_hidden_nodes4,

                                       self.no_of_in_nodes4))

       rad = 1 / np.sqrt(self.no_of_in_nodes5)

       X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)

       self.weights_in_hidden5 = X.rvs((self.no_of_hidden_nodes5,

                                       self.no_of_in_nodes5))

       rad = 1 / np.sqrt(self.no_of_in_nodes6)

       X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)

       self.weights_in_hidden6 = X.rvs((self.no_of_hidden_nodes6,

                                       self.no_of_in_nodes6

   rad = 1 / np.sqrt(self.no_of_hidden_nodes1)

       X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)

       self.weights_hidden_out1 = X.rvs((self.no_of_out_nodes1,

                                        self.no_of_hidden_nodes1))

       rad = 1 / np.sqrt(self.no_of_hidden_nodes2)

       X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
```

```
self.weights_hidden_out2 = X.rvs((self.no_of_out_nodes2,
                                  self.no_of_hidden_nodes2))
rad = 1 / np.sqrt(self.no_of_hidden_nodes3)
X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
self.weights_hidden_out3 = X.rvs((self.no_of_out_nodes3,
                                  self.no_of_hidden_nodes3))
rad = 1 / np.sqrt(self.no_of_hidden_nodes4)
X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
self.weights_hidden_out4 = X.rvs((self.no_of_out_nodes4,
                                  self.no_of_hidden_nodes4))
rad = 1 / np.sqrt(self.no_of_hidden_nodes5)
X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
self.weights_hidden_out5 = X.rvs((self.no_of_out_nodes5,
                                  self.no_of_hidden_nodes5))
rad = 1 / np.sqrt(self.no_of_hidden_nodes6)
X = truncated_normal(mean=0, sd=1, low=-rad, upp=rad)
self.weights_hidden_out6 = X.rvs((self.no_of_out_nodes6,
                                  self.no_of_hidden_nodes6))
```

```
def train(self, input_vector1 , input_vector2 , input_vector3 ,
input_vector4 , input_vector5 , input_vector6 , target_vector1 ,
target_vector2 , target_vector3 , target_vector4 , target_vector5 ,
target_vector6):
        pass
    def run(self, input_vector1 , input_vector2 , input_vector3 ,
input_vector4 , input_vector5 , input_vector6):
        input_vector1 = np.array(input_vector1, ndmin=6).T
        input_vector2 = np.array(input_vector2, ndmin=6).T
        input_vector3 = np.array(input_vector3, ndmin=6).T
        input_vector4 = np.array(input_vector4, ndmin=6).T
        input_vector5 = np.array(input_vector5, ndmin=6).T
        input_vector6 = np.array(input_vector6, ndmin=6).T
input_hidden1 = activation_function(self.weights_in_hidden1 @
input_vector1)
        input_hidden2 = activation_function(self.weights_in_hidden2 @
input_vector2)
        input_hidden3 = activation_function(self.weights_in_hidden3 @
input_vector3)
```

```
        input_hidden4 = activation_function(self.weights_in_hidden4 @
input_vector4)

        input_hidden5 = activation_function(self.weights_in_hidden5 @
input_vector5)

        input_hidden6 = activation_function(self.weights_in_hidden6 @
input_vector6)

        output_vector1 = activation_function(self.weights_hidden_out1 @
input_hidden1)

        output_vector2 = activation_function(self.weights_hidden_out2 @
input_hidden2)

        output_vector3 = activation_function(self.weights_hidden_out3 @
input_hidden3)

        output_vector4 = activation_function(self.weights_hidden_out4 @
input_hidden4)

        output_vector5 = activation_function(self.weights_hidden_out5 @
input_hidden5)

        output_vector6 = activation_function(self.weights_hidden_out6 @
input_hidden6)

        output = np.array([output_vector1 , output_vector2 ,
output_vector3 , output_vector4 , output_vector5 , output_vector6],
ndmin=6)
```

```
    return output
no_of_in_nodes1 = 1
no_of_in_nodes2 = 1
no_of_in_nodes3 = 1
no_of_in_nodes4 = 1
no_of_in_nodes5 = 1
no_of_in_nodes6 = 1
no_of_out_nodes1 = 1
no_of_out_nodes2 = 1
no_of_out_nodes3 = 1
no_of_out_nodes4 = 1
no_of_out_nodes5 = 1
no_of_out_nodes6 = 1
no_of_hidden_nodes1 = 1
no_of_hidden_nodes2 = 1
no_of_hidden_nodes3 = 1
no_of_hidden_nodes4 = 1
no_of_hidden_nodes5 = 1
```

```
no_of_hidden_nodes6 = 1

simple_network = Nnetwork(no_of_in_nodes1,

                no_of_in_nodes2,

                no_of_in_nodes3,

                no_of_in_nodes4,

                no_of_in_nodes5,

                no_of_in_nodes6,

                no_of_out_nodes1,

                no_of_out_nodes2,

                no_of_out_nodes3,

                no_of_out_nodes4,

                no_of_out_nodes5,

                no_of_out_nodes6,

                no_of_hidden_nodes1,

                no_of_hidden_nodes2,

                no_of_hidden_nodes3,

                no_of_hidden_nodes4,

                no_of_hidden_nodes5,
                no_of_hidden_nodes6,
                learning_rate=1)
print(simple_network.run((2 , 8) , (7 , 9) , (3 , 9) , (1 , 3) , (3 , 6) , (9 ,
1)))
```

By implementing the ANN model, the result that our data presented is in the form of an array as follows.

```
[.7857 , -.336 , .64 , -.916 , -.78 , .19]
```

By examining the obtained results, it becomes clear that our data has followed an irregular upward trend. This shows that our tumor in the patient's body is growing insignificantly. That is, from a benign state to It has a malignant state. So our neural network says, if the tumor is not treated as soon as possible, it will be sent to a malignant state.