Application of Neural Network in Breast Cancer Prediction and processing Prostate Cancer images in Python

Samya Praharaj
Tiyasa Dutta
Indian Statistical Institute Kolkata

January 2024

Abstract:

In this article we explore one of the most popular domain of Machine Learning which is based on artificial neural networks known as Deep Learning. Mainly we focus on how Deep learning can be applied for detecting cancer at early stages. Two types of datasets has been considered on Breast cancer and Prostate cancer among which the Prostate cancer is available as the image dataset given in the DICIOM format which actually draws interest as handling medical images in this format in recent days is one of the most important task in the field of medicines. An Artificial Neural Network(ANN) has been applied to the Breast Cancer data for prediction purpose. The study also gives an idea of what are the steps to be followed to read this DICOM image format in a programming language like python and then do the necessaries.

Keywords:

ANN,DICOM,epoch,batch size,HU

1 Introduction:

Deep Learning is a branch of machine learning which is capable of learning complex patterns. It has become very popular in the recent days due to the advances in processing power.Neural Networks builds up the corner stone in Deep Learning fields which are modeled after the structure and function of the human brain. It consists of multiple layers of interconnected nodes. Breast Cancer is one of the precarious conditions that affect women that has no substansive cure till date. Different Deep Learning techniques are used nowadays to predict whether a patient is having cancer at early stages henceforth increasing the chance of survival. Like Breast cancer another worldwide cancer particularly prevailing in men is Prostate Cancer. With the advent of AI detecting this kind of diseases in early stage has become one of the major area of medical science and Machine learning units jointly.

2 Objective:

In this study we mainly have two objectives of which the foremost one is to fit an ANN to a numerical dataset on Breast cancer and check how much accurate the predictions are and the latter one is to learn how to load and pre-process medical image data DCM in Python before further analysis.

3 Description of the Breast Cancer Wisconsin (Diagnostic) Data Set:

The dataset is taken from the official website of Kaggle. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. There are in total 569 observations and 32 features out of which one is categorical. Attribute Information:

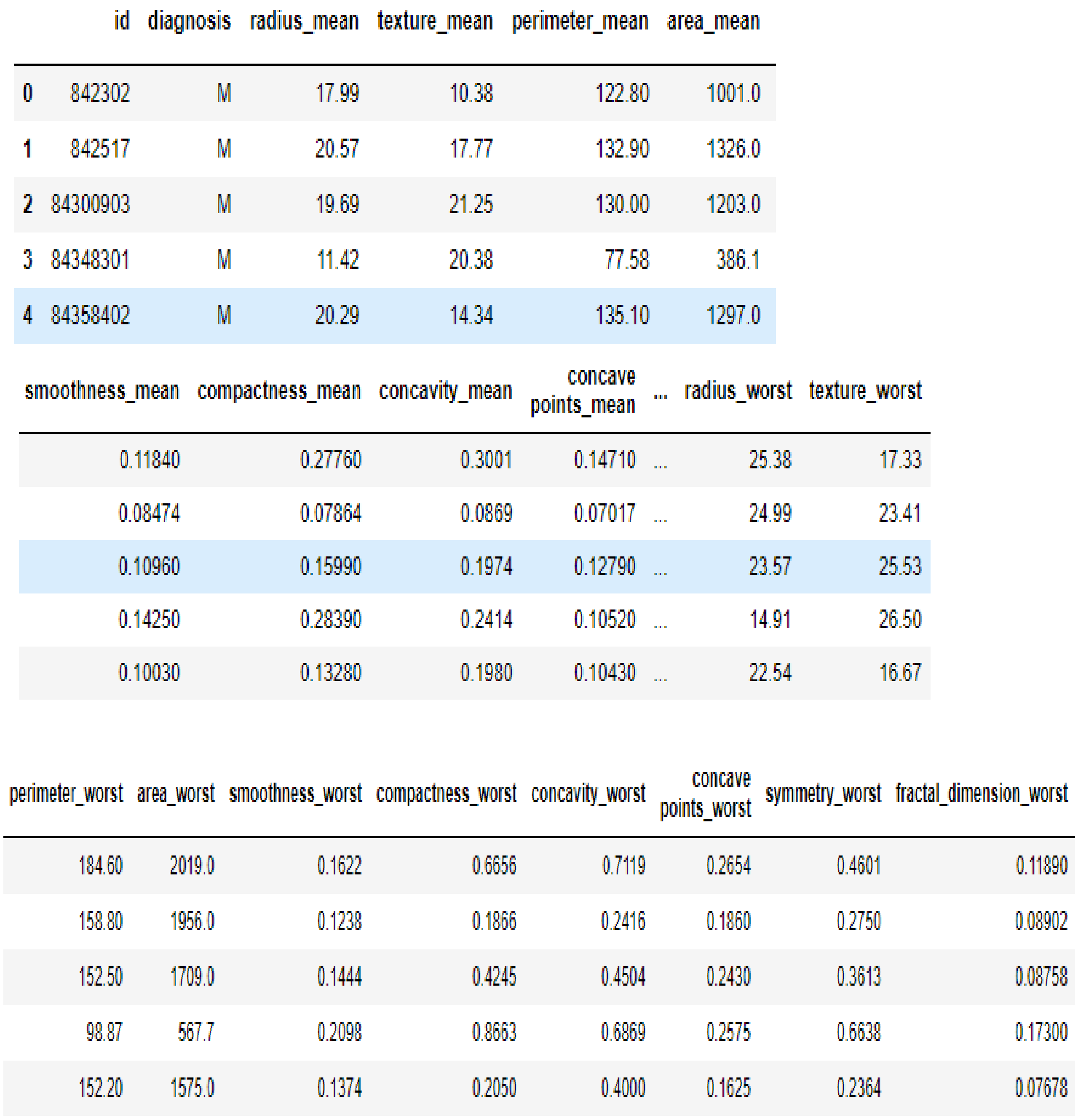
1. ID number
2. Diagnosis (M = malignant, B = benign) 3-32)

Ten real-valued features are computed for each cell nucleus:

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error and ”worst” or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recoded with four significant digits.
Missing attribute values: none
Class distribution: 357 benign, 212 malignant.
Here is an excerpt of the dataset with 5 observations



	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
0	842302	M	17.99	10.38	122.80	1001.0
1	842517	M	20.57	17.77	132.90	1326.0
2	84300903	M	19.69	21.25	130.00	1203.0
3	84348301	M	11.42	20.38	77.58	386.1
4	84358402	M	20.29	14.34	135.10	1297.0

smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	radius_worst	texture_worst
0.11840	0.27760	0.3001	0.14710	...	25.38	17.33
0.08474	0.07864	0.0869	0.07017	...	24.99	23.41
0.10960	0.15990	0.1974	0.12790	...	23.57	25.53
0.14250	0.28390	0.2414	0.10520	...	14.91	26.50
0.10030	0.13280	0.1980	0.10430	...	22.54	16.67

perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890
158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902
152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758
98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300
152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678

Packages needed to be installed for the analysis:

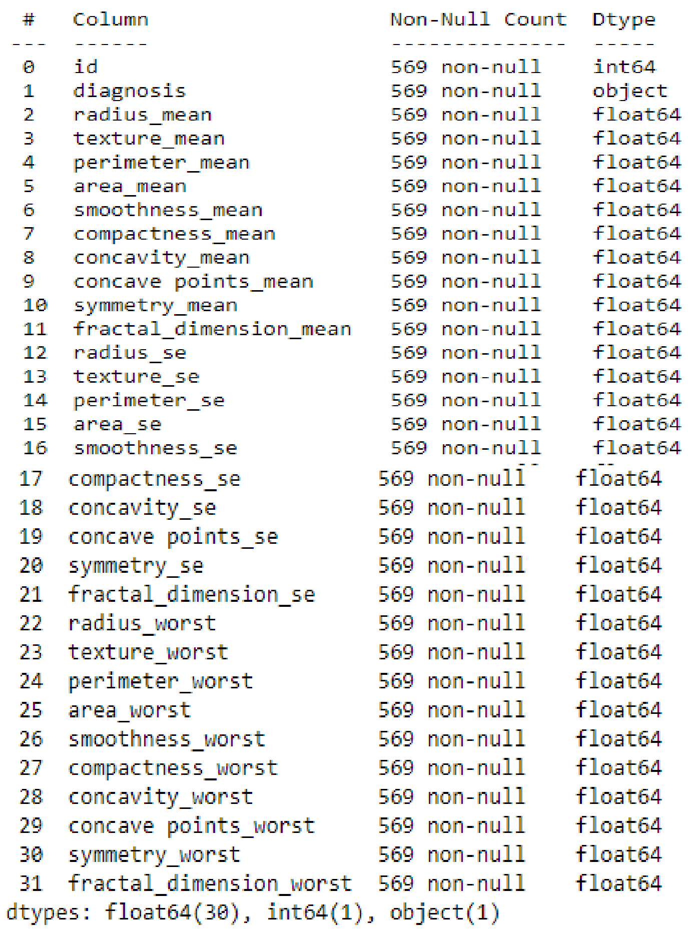
The following pacakges must be installed for the following purposes

- **pandas** : for reading the dataset .
- **matplotlib.pyplot :** this package is mainly used for having plots like histogram,scatter plot etc.
- **numpy :** for performing all matrix,algebraic and numerical operations.
- **tensorflow :** used for fitting ANN to the data.

4 Analysis of the Breast Cancer Wisconsin (Diagnostic) Data Set:

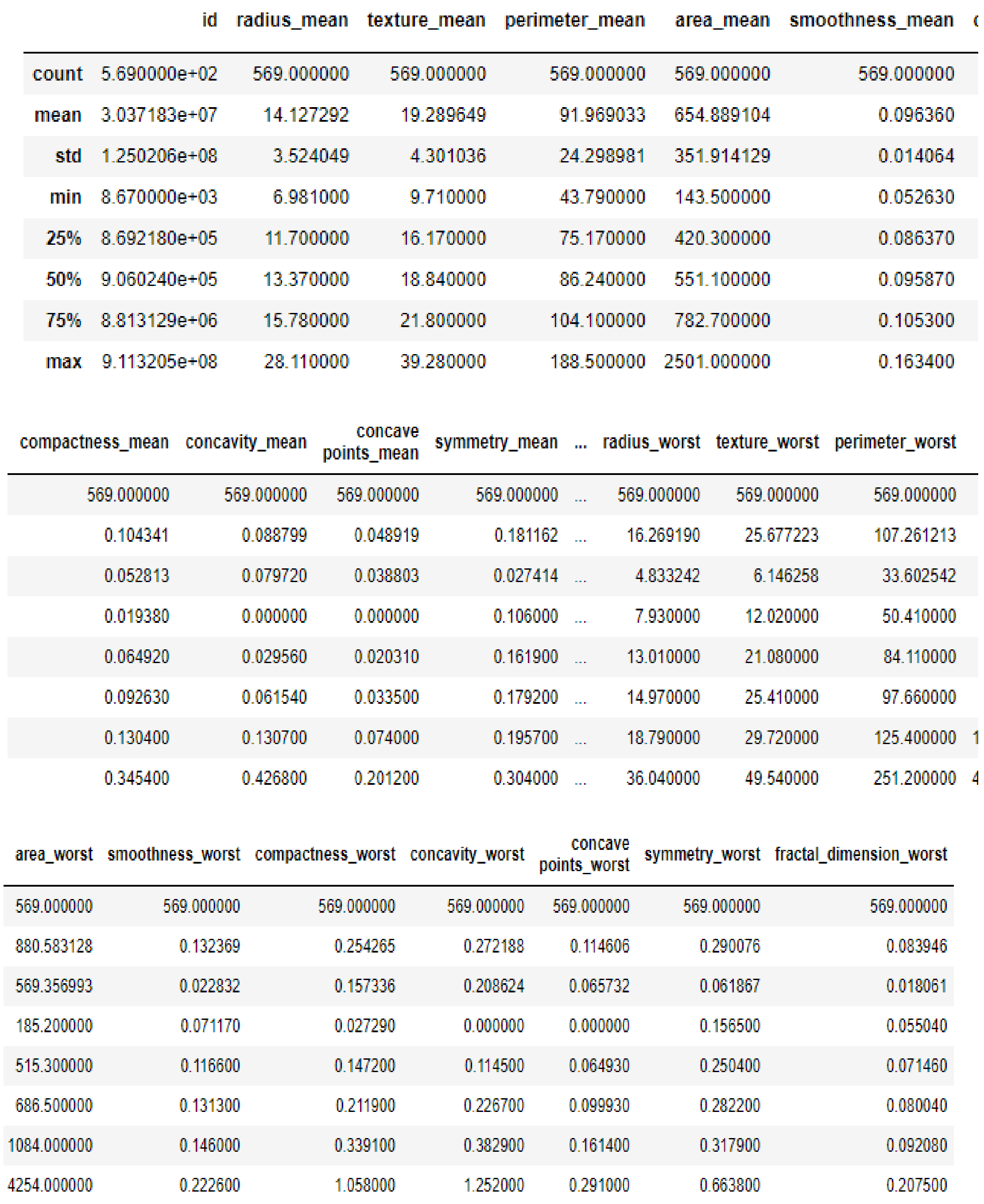
Exploratory Data Analysis:

We will perform a small exploratory analysis over the dataset. Firstly we try to view what types of features we are dealing with , so here is the following:



```
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), int64(1), object(1)
```

So we observe that we have 30 continous covariates which will be treated as the predictor variables when we fit the ANN to the data for prediction purpose. Secondly we will calculate some summary measures over the dataset in python and it is as follows:



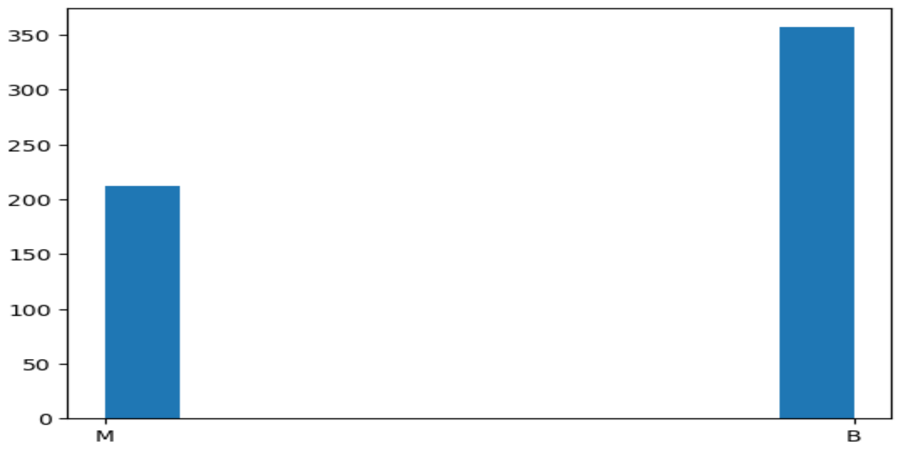
	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400

compactness_mean	concavity_mean	concave points_mean	symmetry_mean	...	radius_worst	texture_worst	perimeter_worst
569.000000	569.000000	569.000000	569.000000	...	569.000000	569.000000	569.000000
0.104341	0.088799	0.048919	0.181162	...	16.269190	25.677223	107.261213
0.052813	0.079720	0.038803	0.027414	...	4.833242	6.146258	33.602542
0.019380	0.000000	0.000000	0.106000	...	7.930000	12.020000	50.410000
0.064920	0.029560	0.020310	0.161900	...	13.010000	21.080000	84.110000
0.092630	0.061540	0.033500	0.179200	...	14.970000	25.410000	97.660000
0.130400	0.130700	0.074000	0.195700	...	18.790000	29.720000	125.400000
0.345400	0.426800	0.201200	0.304000	...	36.040000	49.540000	251.200000

area_worst	smoothness_worst	compactness_worst	concavity_worst	concave points_worst	symmetry_worst	fractal_dimension_worst
569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
880.583128	0.132369	0.254265	0.272188	0.114606	0.290076	0.083946
569.356993	0.022832	0.157336	0.208624	0.065732	0.061867	0.018061
185.200000	0.071170	0.027290	0.000000	0.000000	0.156500	0.055040
515.300000	0.116600	0.147200	0.114500	0.064930	0.250400	0.071460
686.500000	0.131300	0.211900	0.226700	0.099930	0.282200	0.080040
1084.000000	0.146000	0.339100	0.382900	0.161400	0.317900	0.092080
4254.000000	0.222600	1.058000	1.252000	0.291000	0.663800	0.207500

The summary features like count,mean,standard deviation,min,max and the

quantiles of the features are reported .
Next we have a histogram of the response variable which is the diagnosis.



Since its a discrete variable and suppose the two types "Malignant" and "Benign" are coded as 1 and 0 respectively so the plot basically gives the frequency of each type.
Then we look for the strongly correlated variables and we see radius mean and perimeter mean are the two variables awhich are highly positively correlated with a correlation of 0.99875 which is very close to 1 and the justification of this is quiet obvious.

Fitting the Neural Network:

The **Methodology** is given as follows:

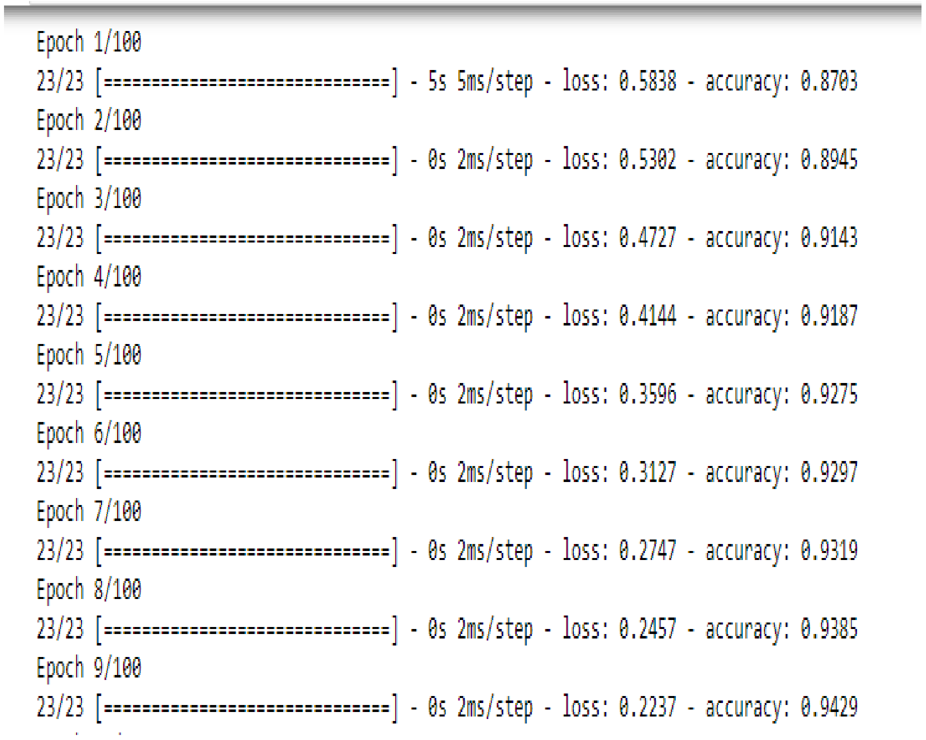
1. The categorical variable "diagnosis is encoded as 1 and 0 for later analysis.
2. 80 % of the data set is taken to be the training data and remaining 20% to be the test data.
3. The continuous covariates are all standardised for meaningful interpretations through out the further analysis.
4. A 2 layer Neural network is fitted to the data.
5. The input layer conists of all the covariates and each hidden layer is made up of 6 units and the input features are carried on to the irst hidden layer by the "**relu**" activation function.
6. From the second hidden layer the sigmoid function is used to generate the output layer.
7. Since we have two classes there is only one unit on the output layer.

8. The "**adam**" optimizer is being used and since its a two class classification we use the binary cross entropy as the loss function.

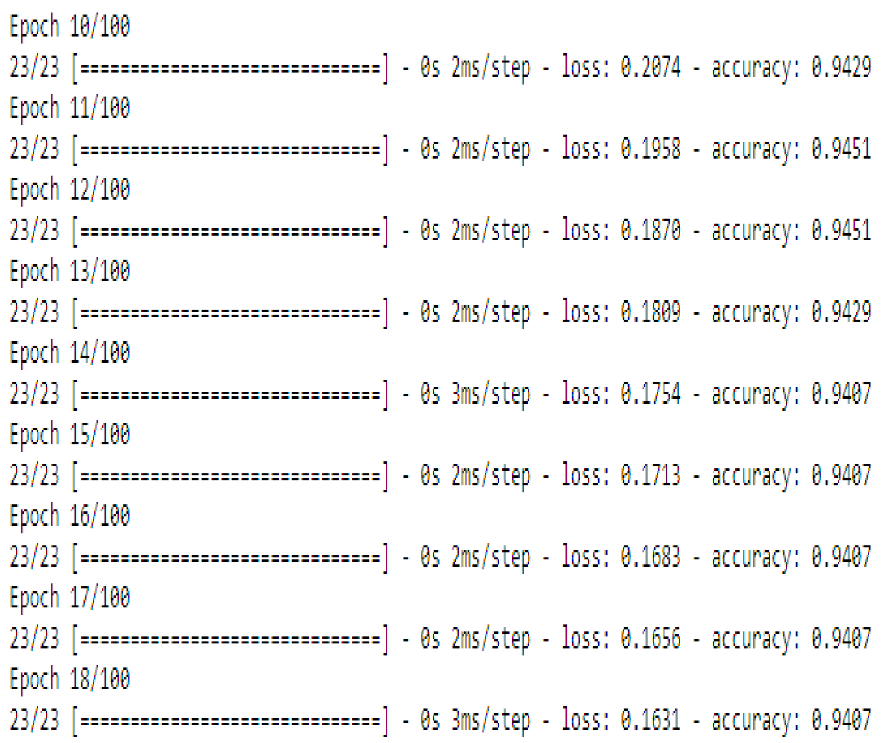
9. The metrics used here is "**accuracy**" that is the number of true positives divided by the total number of positives.

10. The batch size and the epoch were taken to be 20 and 100.

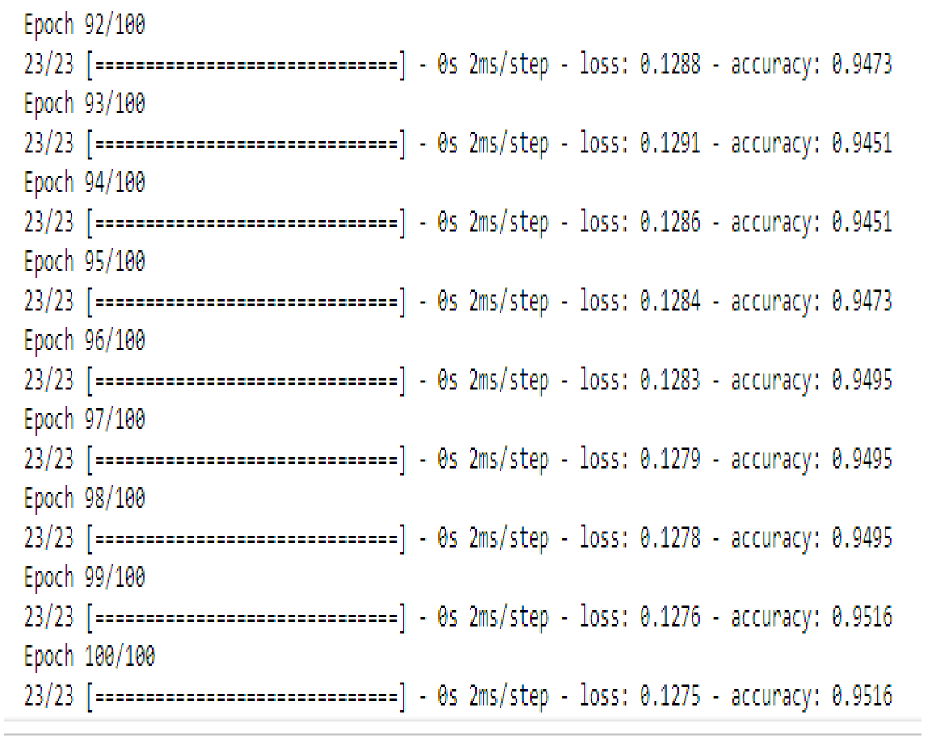
11. The output is given as follows:



```
Epoch 1/100
23/23 [==============================] - 5s 5ms/step - loss: 0.5838 - accuracy: 0.8703
Epoch 2/100
23/23 [==============================] - 0s 2ms/step - loss: 0.5302 - accuracy: 0.8945
Epoch 3/100
23/23 [==============================] - 0s 2ms/step - loss: 0.4727 - accuracy: 0.9143
Epoch 4/100
23/23 [==============================] - 0s 2ms/step - loss: 0.4144 - accuracy: 0.9187
Epoch 5/100
23/23 [==============================] - 0s 2ms/step - loss: 0.3596 - accuracy: 0.9275
Epoch 6/100
23/23 [==============================] - 0s 2ms/step - loss: 0.3127 - accuracy: 0.9297
Epoch 7/100
23/23 [==============================] - 0s 2ms/step - loss: 0.2747 - accuracy: 0.9319
Epoch 8/100
23/23 [==============================] - 0s 2ms/step - loss: 0.2457 - accuracy: 0.9385
Epoch 9/100
23/23 [==============================] - 0s 2ms/step - loss: 0.2237 - accuracy: 0.9429
```



```
Epoch 10/100
23/23 [==============================] - 0s 2ms/step - loss: 0.2074 - accuracy: 0.9429
Epoch 11/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1958 - accuracy: 0.9451
Epoch 12/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1870 - accuracy: 0.9451
Epoch 13/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1809 - accuracy: 0.9429
Epoch 14/100
23/23 [==============================] - 0s 3ms/step - loss: 0.1754 - accuracy: 0.9407
Epoch 15/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1713 - accuracy: 0.9407
Epoch 16/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1683 - accuracy: 0.9407
Epoch 17/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1656 - accuracy: 0.9407
Epoch 18/100
23/23 [==============================] - 0s 3ms/step - loss: 0.1631 - accuracy: 0.9407
```



```
Epoch 92/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1288 - accuracy: 0.9473
Epoch 93/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1291 - accuracy: 0.9451
Epoch 94/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1286 - accuracy: 0.9451
Epoch 95/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1284 - accuracy: 0.9473
Epoch 96/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1283 - accuracy: 0.9495
Epoch 97/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1279 - accuracy: 0.9495
Epoch 98/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1278 - accuracy: 0.9495
Epoch 99/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1276 - accuracy: 0.9516
Epoch 100/100
23/23 [==============================] - 0s 2ms/step - loss: 0.1275 - accuracy: 0.9516
```

Here we have printed upto 19 epochs from the first and the last 8 epochs and we try to explain the output:

Explanation:

- It is basically the training progress of a neural network using the Keras library, specifically for a classification task.
- The training process is divided into epochs, where each epoch represents one pass through the entire training dataset. In this case, we have trained the model for 100 epochs.
- The loss is a measure of how well the model is performing. It represents the error between the predicted values and the actual values. During training, the goal is to minimize this loss. In our output, the loss decreases over epochs, indicating that the model is improving.
- Accuracy is a measure of how well the model is classifying the data. It represents the percentage of correctly classified instances. Here the model's accuracy is also improving over epochs, reaching 95.16% in the last epoch.
- The time taken for each epoch is displayed. For example, the first epoch took 5 seconds, and subsequent epochs took less time (0-2ms), indicating that the model is being trained relatively quickly.

5 Description of the Prostate Cancer Image data:

Prostate cancer is a common form of cancer that occurs in the prostate, a small gland in the male reproductive system. The prostate's primary function is to produce seminal fluid that nourishes and transports sperm. Prostate cancer occurs when abnormal cells in the prostate gland begin to grow uncontrollably, forming tumors. We need to note firstly what is a DCM format of an image.
A DCM file, or DICOM file, stands for "Digital Imaging and Communications in Medicine"; and is a standardized format used for the storage, exchange, and management of medical images and related information. Here's a short note on DCM files:

- Purpose: DCM files are designed to store medical images, such as X-rays, MRIs, CT scans, and ultrasound, along with associated patient information and other data, in a standardized and interoperable format.
- Standardization: DICOM is a globally recognized standard developed by the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA). It ensures that medical images and information can be exchanged between different healthcare systems and devices seamlessly.

- Key Features: **Metadata**: DICOM files include extensive metadata, known as DICOM tags, which provide essential information about the image, patient, equipment, and more.
- **Modality**: Each DICOM file specifies the medical imaging modality (e.g., CT, MRI, ultrasound) and the type of image it contains.
- **Interoperability**: DICOM files are platform-independent and can be viewed and manipulated using various DICOM-compatible software and hardware systems.
- **Security**: DICOM supports security features like encryption and access control, ensuring the privacy and integrity of patient data.

This dataset contains MRI scans of prostate cancer along with other important (meta) data of patients who have cancer. This dataset contains 20K+ DCM files of 26 patients across 26 different studies.

6 Pre-processing of the Image data:

The steps involves: Transforming to HU, Removing Noises,Segmentation and masking.

6.1 Transforming to HU:

We know that image data can be essentially thought of as a matrix where entry to each matrix has a particular numeric value assigned. What is the scale of these numbers ? HU is a scale universally acknowledged as something which helps in quantitative analysis.
Definition: Housefield Units (HU), also known as CT numbers or CT values, are a measure of radiodensity used in computed tomography (CT) imaging. They quantify the attenuation of X-rays as they pass through different tissues, structures, or materials in the human body. Here's a simplified illustration:

- Air: HU around -1000 (relatively low attenuation, dark pixels).
- Water: HU around 0 (baseline reference for HU, middle gray).
- Soft Tissues (e.g., organs): HU positive values (lighter shades, varying by tissue).
- Bones: HU higher positive values (bright white, strong attenuation).

Need for HU :

- **Tissue Differentiation**: HU enables the clear differentiation of various tissues and structures within the body. This differentiation is crucial for diagnostic purposes, as it allows radiologists to distinguish between normal and abnormal anatomical features.

- **Quantitative Analysis**: HU provides a basis for quantitative analysis in CT imaging. It allows for measurements of tissue density, which is valuable in applications such as tumor characterization, assessing lung function, and monitoring disease progression.
- **From Practical Point of View** :
  - Pixels in CT scans generally are in HU. This is NOT true for all type of tests.
  - MRI scans for example need to be converted to HU. Our dataset contains MRI scans and hence we need to do this.
  - The HU in MRI (h1) can be computed as it is a linear function of the pixel value (p1) as follows : h1 = slope1 * p 1+ intercept1. The exact value of slope1 and intercept1 of image 1 is available in the extra data available in DCM file.

6.2 Noise Cancellation :

Definition: Noise in medical images refers to random or unwanted variations in pixel values that obscure or distort the true anatomical or pathological information within the image. It is an inherent aspect of imaging processes and can have a detrimental impact on image quality and diagnostic accuracy.
Sources of Noise in Medical Images:

- **Electronic Noise**: This type of noise arises from electronic components within the imaging system, such as detectors and amplifiers.
- **Photon Noise**: It occurs due to the random nature of photon interactions with the patient's tissues and the detector.
- **Patient Motion**: In cases of patient motion during image acquisition, motion artifacts can introduce noise.

Noise Reduction Techniques:

1. **Filtering**: Various filtering techniques, such as Gaussian filtering and median filtering, can be applied to reduce noise while preserving important image details.
2. **Iterative Reconstruction:** Advanced reconstruction algorithms, like statistical iterative reconstruction, aim to reduce noise while maintaining image quality.
3. **Increased Dose:** Sometimes, increasing the radiation dose or the signal-to-noise ratio during image acquisition can reduce noise, but this must be balanced with patient safety concerns.

6.3 Segmentation AND Masking :

In the context of noise cancellation in image processing, "segmentation" and "masking"; refer to two key steps used to identify and isolate the regions of interest or areas that contain the desired information within an image. These steps are essential for effectively removing noise or unwanted elements from an image while retaining the relevant content. Here's a brief explanation of each:
Segmentation:

- **Definition**: Masking involves creating a binary mask that acts as a filter or overlay on an image. This binary mask consists of 0s (typically black) and 1s (typically white), where 0s correspond to areas to be excluded or "masked out", and 1s correspond to areas to be retained or "unmasked".
- **Purpose**: Masking is used to specify which regions or elements of the image should be preserved (unmasked) and which should be suppressed (masked out) during further processing. It helps focus on the regions of interest and eliminate noise or undesired components.
- **Method**: To create a mask, you can use image processing techniques or thresholding to define the areas to be masked (set to 0) and those to be unmasked (set to 1). The mask can then be applied to the original image by element-wise multiplication, effectively eliminating or reducing the noise.

To be noted: In the context of noise cancellation, the segmentation step is used to identify regions that contain the relevant information, while the masking step creates a binary mask that isolates those regions. By masking out the areas that are not part of the region of interest, you can focus on the essential content and reduce or eliminate noise in the image, ultimately enhancing the quality and clarity of the desired information. These steps are commonly used in various image processing and computer vision tasks. There are also other steps like tilting and cropping involved. But those mostly depend on data at hand.

7 Analysis of the image dataset:

Packages needed to be installed:

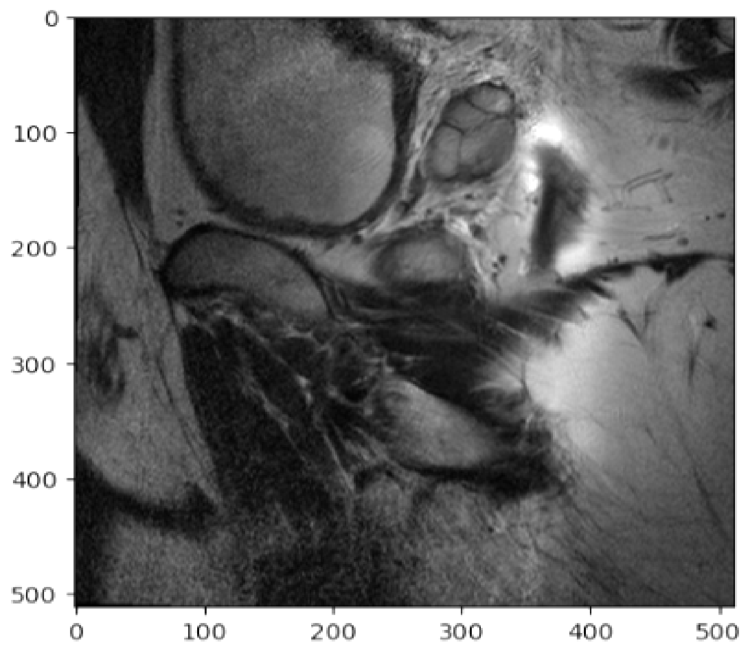
The packages needs to be installed in python for the analysis along with their functions are given below:

- **pydicom**: pydicom is a Python library for working with DICOM (Digital Imaging and Communications in Medicine) files, which are the standard format for medical images. It allows you to read, manipulate, and analyze medical images and their associated metadata.

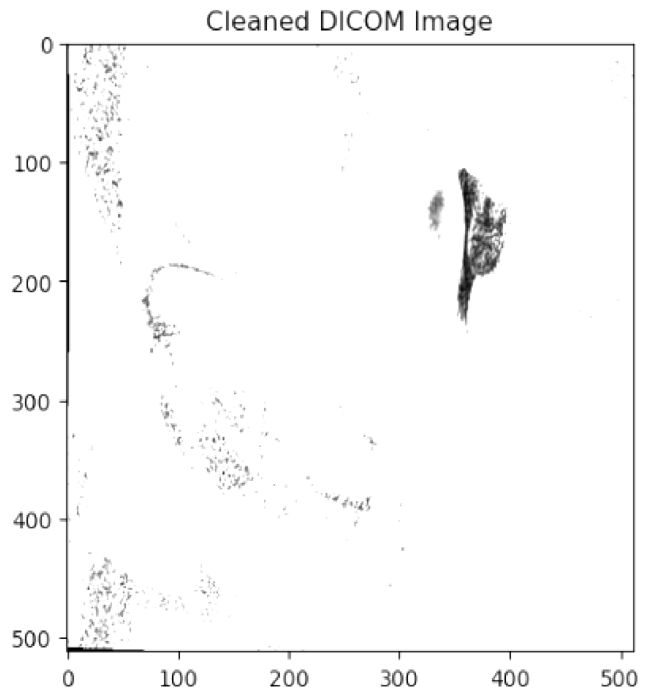
- **skimage:** It seems you intended to import scikit-image, which is a collection of algorithms for image processing. This library provides a wide range of functions for tasks such as image filtering, segmentation, and feature extraction. It is often used in conjunction with numpy and matplotlib for image analysis and visualization.
- **scipy:** scipy is a scientific computing library in Python. It builds upon numpy and provides additional functionality for various scientific and engineering applications, including signal processing, statistics, and optimization.
- **scipy.ndimage.morphology:** This specific module from scipy is used for morphological image processing, which includes operations like dilation, erosion, and other transformations used to process and manipulate images. These operations are often used in tasks such as noise reduction and feature extraction.

Loading the images in Python extracting it from the DCM format:

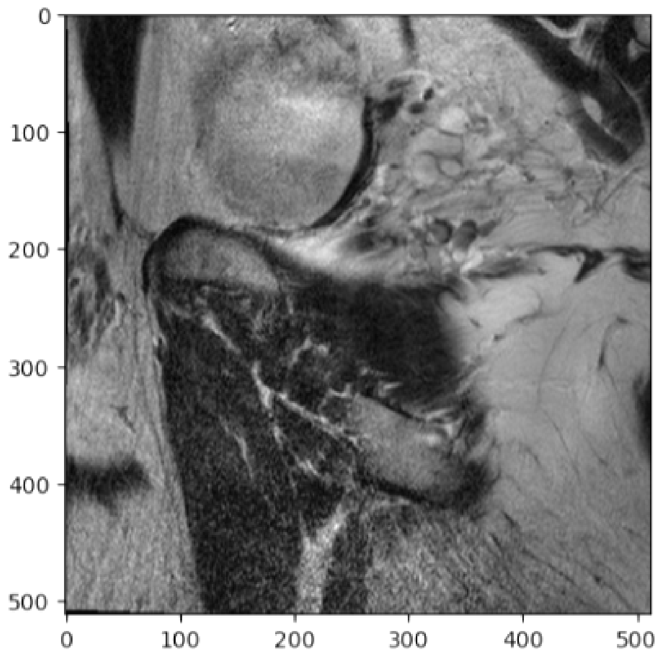
The medical image is loaded and it looks as follows :

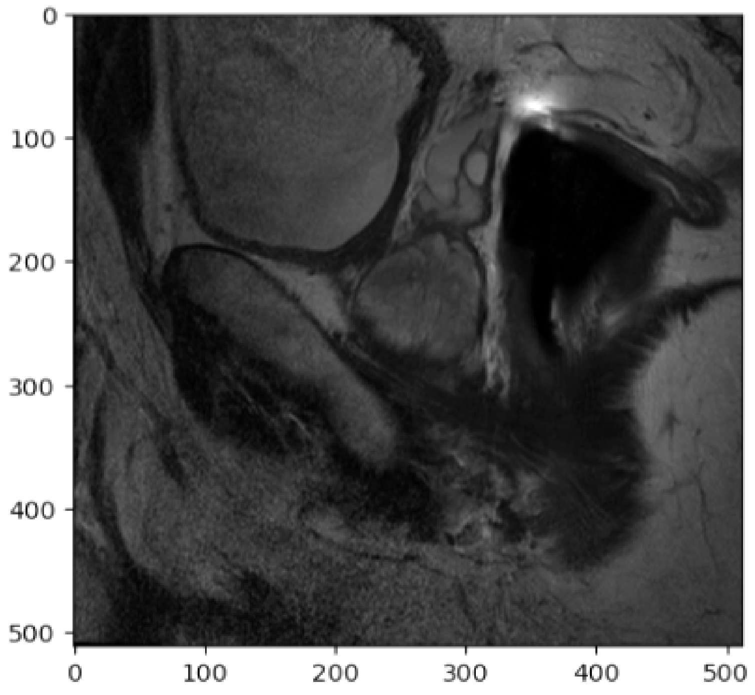


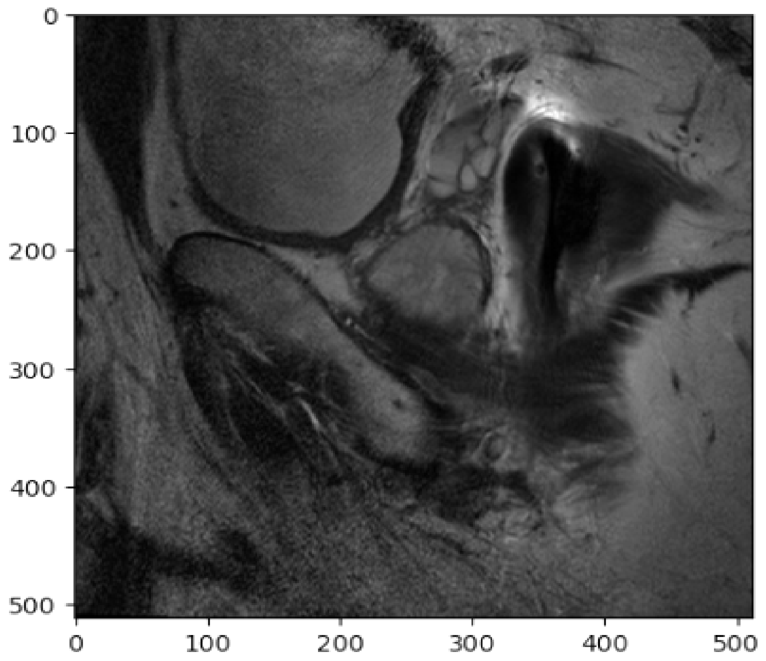
Here firstly the pixel data is extracted from the DCM format and then they are converted to HU to generate the plot (the details is provided in the Appendix part). Next we remove the noise and we have a cleaned DICOM image



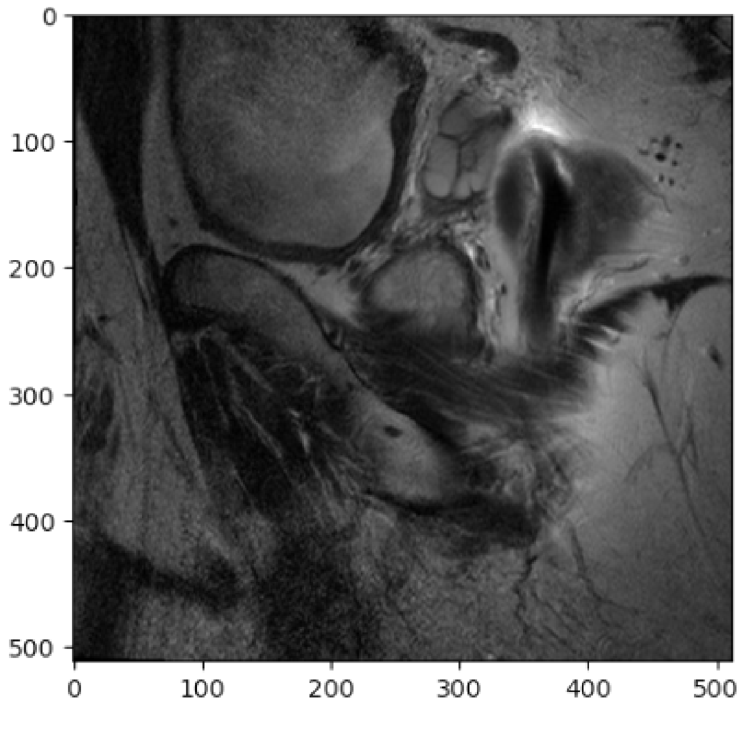
Here we present the 5 actual pictures of the same patient over different periods of the disease MRI images (In decreasing stage)


0
100
200
300
400
500
0
100
200
300
400
500


0
100
200
300
400
500
0
100
200
300
400
500

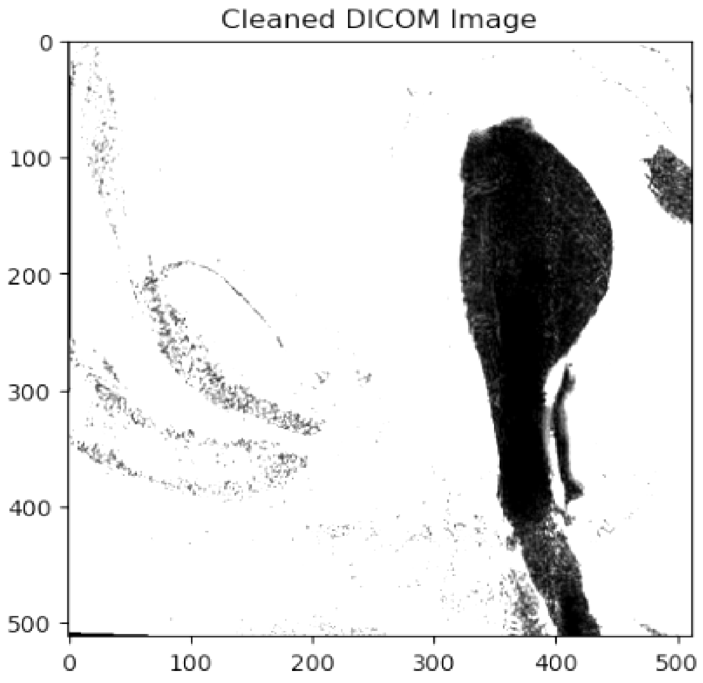


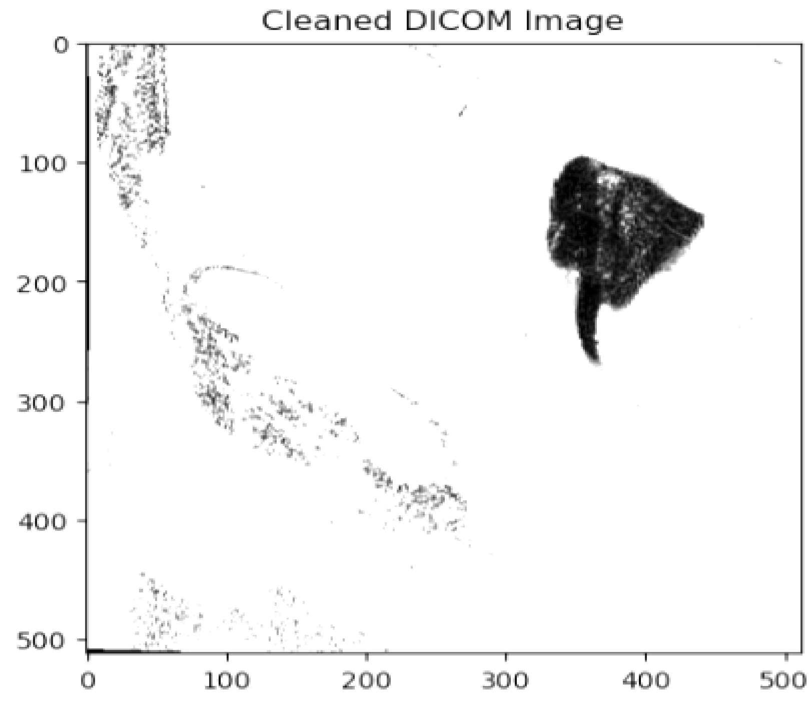
0
100
200
300
400
500
0
100
200
300
400
500

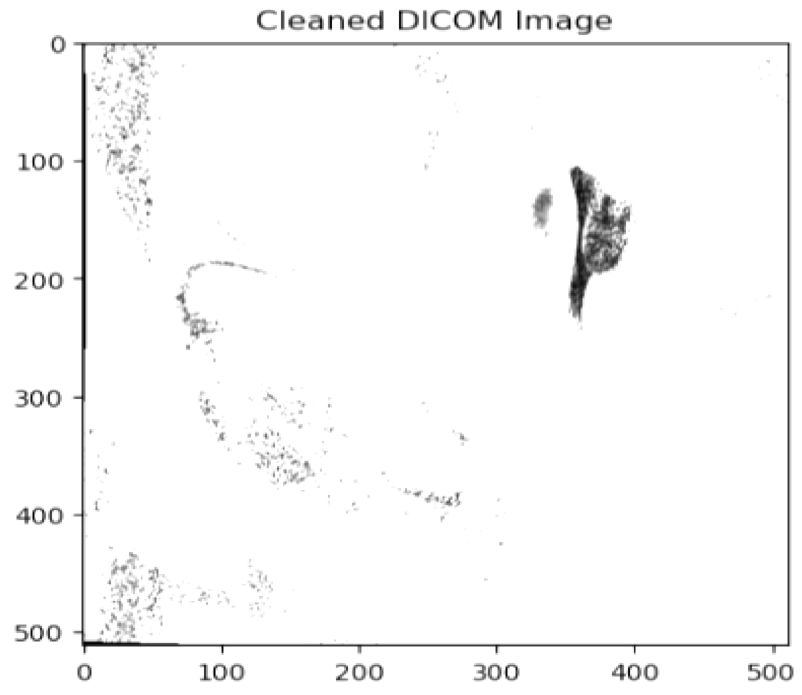


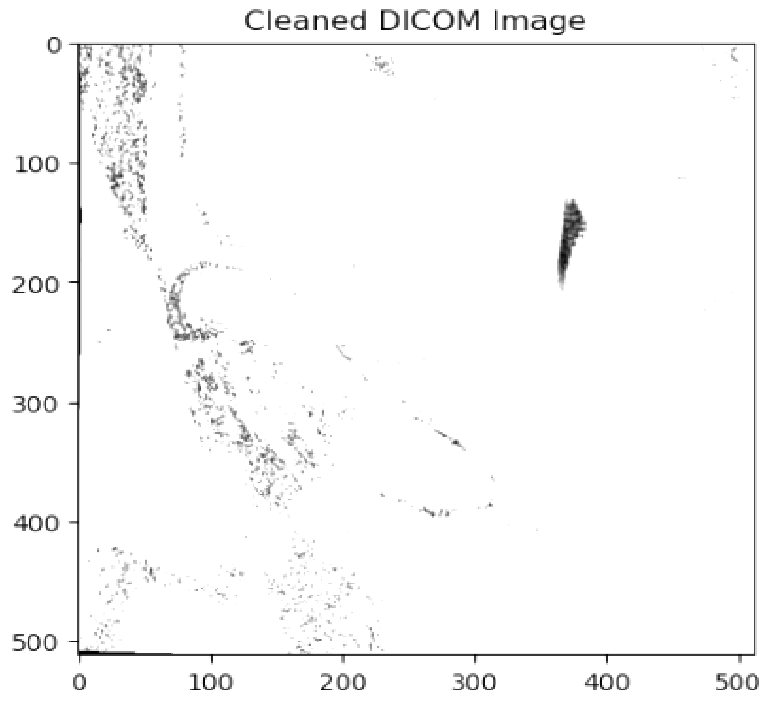
0
100
200
300
400
500
0
100
200
300
400
500

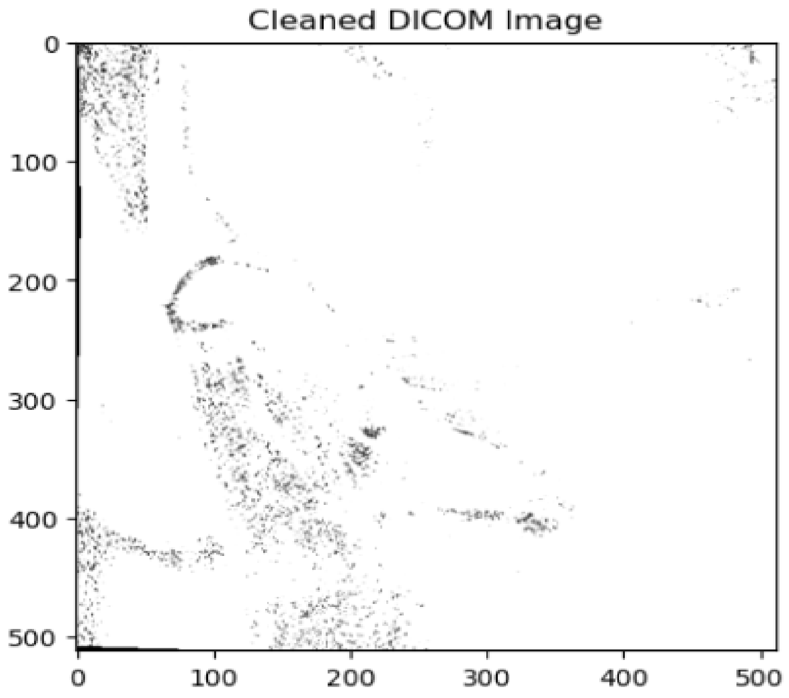
We also have the corresponding noise cancelled images as follows:




Cleaned DICOM Image
0
100
200
300
400
500
0
100
200
300
400
500


Cleaned DICOM Image
0
100
200
300
400
500
0
100
200
300
400
500


Cleaned DICOM Image
0
100
200
300
400
500
0
100
200
300
400
500


Cleaned DICOM Image
0
100
200
300
400
500
0
100
200
300
400
500

So from the images above we can comment that since we have shown how to process the images step by step and its in a decreasing manner it implies that the areas that should be masked out are more as the time period passes on .

8 Conclusion :

In conclusion, our study focused on the application of neural network models for predicting breast cancer. Leveraging a dataset comprising various features related to breast cancer patients, we successfully trained a neural network model with 100 epochs. The model demonstrated promising performance, achieving an accuracy of 95.16 % on the training data.While these results are encouraging, it is crucial to validate the model's generalization performance on independent datasets to ensure its reliability in real-world scenarios. Further refinement of the model and exploration of hyperparameter tuning may contribute to even better predictive accuracy.
On the other hand our image dataset analysis of prostate cancer using Python has yielded valuable insights and potential avenues for further research. Leveraging state-of-the-art image processing and machine learning techniques, we successfully explored and extracted meaningful information from the dataset.Our work sets the foundation for continued exploration into the intersection of medical imaging and machine learning for prostate cancer diagnosis. Future research should focus on expanding the dataset,and classification problems.

9 Appendix:

The code snippets of the entire study is given below:

9.1 Code snippets for the Breast Cancer Analysis:

```
#importing necessary librarires
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
#loading the data set into python
d=pd.read_csv(r'C:\Users\Tiyasa Dutta\Downloads\Breast Cancer.csv')
#having a view of the dataset
d.head(n=5)
d.shape
d.info()
#calculating the summary of the data
d.describe()
d.isnull().sum()
d.value_counts("diagnosis")
#histogram of the reponse variable
```

```
plt.hist(x="diagnosis",data=d)
plt.show
#finding the strongest correlation
d.corr(method='pearson')
from sklearn.preprocessing import LabelEncoder
# Create a LabelEncoder instance
label_encoder = LabelEncoder()
# Fit and transform the 'species' column
d['diagnosis_encoded'] = label_encoder.fit_transform(d['diagnosis'])
d1=d.drop(columns=['diagnosis'])

# Display the modified DataFrame
print(d1)
X=d1.iloc[:,range(1,10)]
Y=d1.iloc[:,31]
from sklearn.model_selection import train_test_split
#spliting the data into train and test sets
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.2,random_state=20)
print(X_train)
print(Y_train)
#perform feature scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
#initialising ANN
ann=tf.keras.models.Sequential()
#adding first hidden layer
ann.add(tf.keras.layers.Dense(units=6,activation="relu"))
#adding 2nd hidden layer
ann.add(tf.keras.layers.Dense(units=6,activation="relu"))
#adding output layer
ann.add(tf.keras.layers.Dense(units=1,activation="sigmoid"))
#compiling ANN
ann.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
#Fitting ANN
ann.fit(X_train,Y_train,batch_size=20,epochs=100)
```

9.2 Code snippets for processing the image dataset of Prostate Cancer:

```
#loading the DICOM images
fp1 = r'C:\Users\Dell\OneDrive\Desktop\PROSTATE_MRI\PROSTATE-MRI\MIP-PROSTATE-01-0001
\04-30-2006-NA-MRI Prostate ERC-40831\401.000000-T2 TSE sag-01264\1-01.dcm'
mi1 = pydicom.read_file(fp1)
```

```
fp2 = r'C:\Users\Dell\OneDrive\Desktop\PROSTATE_MRI\PROSTATE-MRI\MIP-PROSTATE-01-0001
\04-30-2006-NA-MRI Prostate ERC-40831\401.000000-T2 TSE sag-01264\1-02.dcm'
mi2 = pydicom.read_file(fp2)
fp3 = r'C:\Users\Dell\OneDrive\Desktop\PROSTATE_MRI\PROSTATE-MRI\MIP-PROSTATE-01-0001\
   04-30-2006-NA-MRI Prostate ERC-40831\401.000000-T2 TSE sag-01264
   \1-03.dcm'
mi3 = pydicom.read_file(fp3)
fp4 = r'C:\Users\Dell\OneDrive\Desktop\PROSTATE_MRI\PROSTATE-MRI\MIP-PROSTATE-01-0001
\04-30-2006-NA-MRI Prostate ERC-40831\401.000000-T2 TSE sag-01264\1-04.dcm'
mi4 = pydicom.read_file(fp4)
fp5 = r'C:\Users\Dell\OneDrive\Desktop\PROSTATE_MRI\PROSTATE-MRI\MIP-PROSTATE-01-0001
\04-30-2006-NA-MRI Prostate ERC-40831\401.000000-T2 TSE sag-01264\1-10.dcm'
mi5 = pydicom.read_file(fp5)
#extracting the pixels
img1 = mi1.pixel_array
img2 = mi2.pixel_array
img3 = mi3.pixel_array
img4 = mi4.pixel_array
img5 = mi5.pixel_array
plt.imshow(img1,cmap='gray')
#transforming to HU
def transform_to_hu(mi,img):
intercept = mi.RescaleIntercept
slope = mi.RescaleSlope
hu_image = img*slope + intercept
return hu_image
def window_image(img,window_center,window_width):
img_min = window_center - window_width
img_max = window_center + window_width
window_image = img.copy()
window_image[window_image &lt; img_min] = img_min
window_image[window_image&gt;img_max] = img_max
return window_image

def remove_noise(fp, display=False):
mi = pydicom.read_file(fp)
img = mi.pixel_array

hu_image = transform_to_hu(mi, img)
prt_image = window_image(hu_image, 40, 80)

# Use the dilation function from scipy.ndimage instead of morphology.dilation
segmentation = ndimage.morphology.binary_dilation(prt_image,
structure=numpy.ones((1, 1)))
labels, label_nb = ndimage.label(segmentation)
```

```
label_count = numpy.bincount(labels.ravel().astype(int))
label_count[0] = 0

mask = labels == label_count.argmax()

# Continue using morphology.dilation here
mask = ndimage.morphology.binary_dilation(mask, numpy.ones((1, 1)))
mask = ndimage.morphology.binary_fill_holes(mask)
mask = ndimage.morphology.binary_dilation(mask, numpy.ones((3, 3)))
masked_image = mask * prt_image
return masked_image
cleaned_image = remove_noise(fp3, display=True)

# Display the cleaned image if 'display' is set to True
if display:
plt.imshow(cleaned_image, cmap='gray')
plt.title('Cleaned DICOM Image')
plt.show()
```

10 References:

- Deep Learning Based Methods for Breast Cancer Diagnosis: A Systematic Review and Future Direction-Kwangsoo Kim