

BREAST CANCER DETECTION MODEL

A MINOR PROJECT REPORT

Submitted by

Akriti Priya (RA2011003010344)

Debosmita Sarkar (RA2011003010371)

Ayesha Ali(RA2011003010350)

Under the guidance of

Dr. P Selvaraj , Associate Professor

In a partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY



SRM Nagar, Kattankulathur, Chengalpattu District

MAY 2023



BONAFIDE CERTIFICATE

Certified to be the bonafide record of work done by Ayesh Ali (RA2011003010350), Akriti Priya (RA2011003010344), Debosmita Sarkar (RA2011003010371) of Computer Science and Engineering , B.Tech Degree course in the Mini-Project “18CSC204J-Artificial Intelligence” in SRM Institute of Science and Technology ,Kattankulathur during the academic year 2022-2023.

Faculty Incharge

Head of the Department

Date:

TABLE OF CONTENTS

| S.NO | TITLE | PAGE NO. |
|-------------|--|-----------------|
| 1. | Abstract | 4 |
| 2. | Introduction | 5 |
| 3. | Proposed Architecture | 6 |
| 4. | Used ML models explanation | 9 |
| 5. | Algorithm/ Full executable code for the Problem | 10 |
| 6. | Explanation of Source code with example | 13 |
| 7. | Execution screenshots | 17 |
| 8. | Conclusion | 26 |
| 9. | References | 27 |

ABSTRACT

Breast cancer is the most common cancer among women worldwide, and early detection is crucial for improving survival rates. Machine learning techniques have emerged as effective tools for analyzing complex medical data and detecting breast cancer. This paper presents an abstract on breast cancer detection using supervised machine learning models such as logistic regression, decision tree, and random forest.

The dataset used in this project contains various features such as age, tumor size, tumor grade, and histology. The first step in the analysis is data pre-processing, which involves handling missing values, scaling the data, and encoding categorical variables. After pre-processing, the dataset is split into training and testing sets to evaluate the performance of the models.

Logistic regression is a linear model that predicts the probability of an event occurring. It is a simple and fast algorithm that is commonly used for binary classification problems. Decision trees, on the other hand, are non-parametric models that can handle both categorical and continuous data. They are easy to interpret and can capture complex relationships between features. Random forest is an ensemble method that combines multiple decision trees to improve the predictive performance of the model.

The models are trained on the training set and evaluated on the testing set using various performance metrics such as accuracy, precision, recall, and F1-score. The results show that all the models achieved high accuracy, with random forest performing the best. The models also provide feature importance rankings, which can help identify the most important features in the dataset.

In conclusion, supervised machine learning models such as logistic regression, decision tree, and random forest can effectively detect breast cancer and provide valuable insights into the most important features. These models have the potential to improve early detection and contribute to better treatment outcomes for breast cancer patients.

INTRODUCTION

According to World health organization, Breast cancer is the most frequent cancer among women and it is the second dangerous cancer after lung cancer. In 2018, from the research it is estimated that total 627,000 women lost their life due to breast cancer that is 15% of all cancer deaths among women. In case of any symptom, people visit to oncologist. Doctors can easily identify breast cancer by using Breast ultrasound, Diagnostic mammogram, Magnetic resonance imaging (MRI), Biopsy. Based on these test results, doctor may recommend further tests or therapy. Early detection is very crucial in breast cancer. If chances of cancer are predicted at early stage then survivability chances of patient may increase. An alternate way to identify breast cancer is using machine learning algorithms for prediction of abnormal tumor. Thus, the research is carried out for the proper diagnosis and categorization of patients into malignant and benign groups.

Three types of tumors are as follows:-

- a. **Benign tumors** are not cancerous they cannot spread or they can grow very slowly. And if doctors remove them, then they cannot cause any harm to the human body.
- b. In **Premalignant tumors** the cells are not cancerous but they have potential to become malignant.
- c. **Malignant cells** are cancerous and they can spread rapidly in body.

In machine learning, cancer classification can be done using benign or malignant cells could significantly improve our ability for prognosis in cancer patients, poor progress has been made for their application in the clinics. However, before gene expression profiling can be used in clinical practice, studies with larger data samples and more adequate validation are needed. And thus, our aim is to develop a prediction system that can predict chances of breast cancer on a huge data.

The dataset used in this project is from Breast Cancer Wisconsin (Diagnostic) Data Set (<https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data>)

PROPOSED ARCHITECTURE

Role Of Machine Learning In Detection Of Breast Cancer

A mammogram is an x-ray picture of the breast. It can be used to check for breast cancer in women who have no signs or symptoms of the disease. It can also be used if you have a lump or other sign of breast cancer.

Screening mammography is the type of mammogram that checks you when you have no symptoms. It can help reduce the number of deaths from breast cancer among women ages 40 to 70. But it can also have drawbacks. Mammograms can sometimes find something that looks abnormal but isn't cancer. This leads to further testing and can cause you anxiety. Sometimes mammograms can miss cancer when it is there. It also exposes you to radiation. You should talk to your doctor about the benefits and drawbacks of mammograms. Together, you can decide when to start and how often to have a mammogram.

Now while its difficult to figure out for physicians by seeing only images of x-ray that weather the tumor is toxic or not training a machine learning model according to the identification of tumour can be of great help.

Breast cancer detection is a widely researched area in machine learning, and there are several approaches that can be used to build a predictive model.

One possible architecture for a breast cancer detection project using supervised learning algorithms like logistic regression, decision tree, and random forest could include the following steps:

- **Data Collection and Preprocessing:** The first step in any machine learning project is to collect and preprocess the data. For breast cancer detection, the data could include features such as age, family history of cancer, mammogram results, and biopsy reports. The data should be cleaned, normalized, and transformed as needed to ensure that it is suitable for use with the chosen algorithms.
- **Feature Selection:** Once the data is preprocessed, the next step is to select the most relevant features to be used for building the model. This can be done using techniques like correlation analysis and feature importance ranking.
- **Model Selection:** With the features selected, the next step is to choose the appropriate algorithm(s) to use for building the predictive model. For breast cancer detection, common algorithms include logistic regression, decision trees, and random forests. Each algorithm has its strengths and weaknesses, so it's important to experiment with different algorithms and hyperparameters to find the best approach.
- **Model Training:** After selecting the algorithm(s), the next step is to train the model using the preprocessed data. This involves splitting the data into training and testing sets and using the training set to fit the model.
- **Model Evaluation:** Once the model is trained, it's important to evaluate its performance using metrics like accuracy, precision, recall, and F1 score. A confusion matrix can also be used to visualize the performance of the model.
- **Model Tuning:** Based on the evaluation results, the model may need to be fine-tuned to improve its performance. This could involve adjusting hyperparameters, using different feature selection techniques, or trying out different algorithms.

- **Model Deployment:** Once the final model has been developed and evaluated, it can be deployed for use in breast cancer detection. This could involve creating a web or mobile application or integrating the model into an existing healthcare system.

Overall, building a breast cancer detection model using supervised learning algorithms like logistic regression, decision tree, and random forest can be a complex process, but by following these steps, it's possible to develop an accurate and reliable predictive model that can be used to improve patient outcomes.

EXPLANATION OF VARIOUS MACHINE LEARNING MODELS USED IN BREAST CANCER DETECTION

SUPERVISED MACHINE LEARNING MODELS

Supervised Learning is a type of system in which both input and desired output data are provided. Input and output data are labelled for classification to provide a learning basis for future data processing. Supervised systems provide the learning algorithms with known quantities to support future judgments.

1. Decision Tree

Decision tree is the most powerful and popular tool for classification and prediction in machine learning. The Decision trees algorithm consists of two parts: nodes and rules (tests). A Decision tree is like tree structure, where node denotes a test on an attribute, Branch represents an outcome of the test, and each leaf node holds a class label. For detecting breast cancer, its leaf nodes are categorised as benign and malignant. And then certain rules are established to check tumor is benign or malignant.

2. Random Forest Classification

Random forest algorithm is a supervised classification algorithm. In this classifier, the higher the number of trees in the forest gives the high accuracy results.

- a. Random forest algorithm can use for both classification and the regression task and can handle the missing values too. For this dataset, we have already handled missing values of attributes. Random forest classifier doesn't over fits the model, if it includes many trees.
- b. It can also work on categorical values too. In this case, we had categorical data as B & M representing benign & malignant which is further converted to numeric data as 0 & 1 respectively.

3. Logistic Regression:

A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. It is generally used when the dependent variable is binary and provides a constant output

Source Code

#Description: This program detects breast cancer, based off of data.

importing libraries

import numpy

import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns

reading data from the file

df=pd.read_csv("data.csv")

df.head()

df.info()

return all the columns with null values count

df.isna().sum()

return the size of dataset

df.shape

remove the column

df=df.dropna(axis=1)

shape of dataset after removing the null column

df.shape

describe the dataset

df.describe()

Get the count of malignant<M> and Benign cells

df['diagnosis'].value_counts()

label encoding(convert the value of M and B into 1 and 0)

from sklearn.preprocessing import LabelEncoder

```
labelencoder_Y = LabelEncoder()
df.iloc[:,1]=labelencoder_Y.fit_transform(df.iloc[:,1].values)

df.head()

sns.pairplot(df.iloc[:,1:5],hue="diagnosis")

# get the correlation
df.iloc[:,1:32].corr()

# visualize the correlation
plt.figure(figsize=(10,10))
sns.heatmap(df.iloc[:,1:10].corr(),annot=True,fmt=".0%")

# split the dataset into dependent(X) and Independent(Y) datasets
X=df.iloc[:,2:31].values
Y=df.iloc[:,1].values
```

In [18]:

```
# splitting the data into training and test dataset
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20,random_state=0)
```

In [19]:

```
# feature scaling
from sklearn.preprocessing import StandardScaler
X_train=StandardScaler().fit_transform(X_train)
X_test=StandardScaler().fit_transform(X_test)

# models/ Algorithms

def models(X_train,Y_train):
    #logistic regression
    from sklearn.linear_model import LogisticRegression
    log=LogisticRegression(random_state=0)
    log.fit(X_train,Y_train)

    #Decision Tree
```

```
from sklearn.tree import DecisionTreeClassifier
tree=DecisionTreeClassifier(random_state=0,criterion="entropy")
tree.fit(X_train,Y_train)
```

```
#Random Forest
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
forest=RandomForestClassifier(random_state=0,criterion="entropy",n_estimators=10)
```

```
forest.fit(X_train,Y_train)
```

```
print('[0]logistic regression accuracy:',log.score(X_train,Y_train))
```

```
print('[1]Decision tree accuracy:',tree.score(X_train,Y_train))
```

```
print('[2]Random forest accuracy:',forest.score(X_train,Y_train))
```

```
return log,tree,forest
```

```
model=models(X_train,Y_train)
```

```
# testing the models/result
```

```
from sklearn.metrics import accuracy_score
```

```
from sklearn.metrics import classification_report
```

```
for i in range(len(model)):
```

```
    print("Model",i)
```

```
    print(classification_report(Y_test,model[i].predict(X_test)))
```

```
    print('Accuracy : ',accuracy_score(Y_test,model[i].predict(X_test)))
```

```
# prediction of random-forest
```

```
pred=model[2].predict(X_test)
```

```
print('Predicted values:')
```

```
print(pred)
```

```
print('Actual values:')
```

```
print(Y_test)
```

```
from joblib import dump
```

```
dump(model[2],"Cancer_prediction.joblib")
```

EXPLANATION OF CODE

- **START PROGRAMMING**

The first thing before writing a single line of code is to put in a description in comments of what the code does. This way anyone can look back on our code and know exactly what it does.

- **import the packages/libraries to make it easier to write the program. Next we will load the data, and print the first 7 rows of data.**

NOTE: Each row of data represents a patient that may or may not have cancer.

- **Explore the data and count the number of rows and columns in the data set. There are 569 rows of data which means there are 569 patients in this data set, and 33 columns which mean there are 33 features or data points for each patient.**

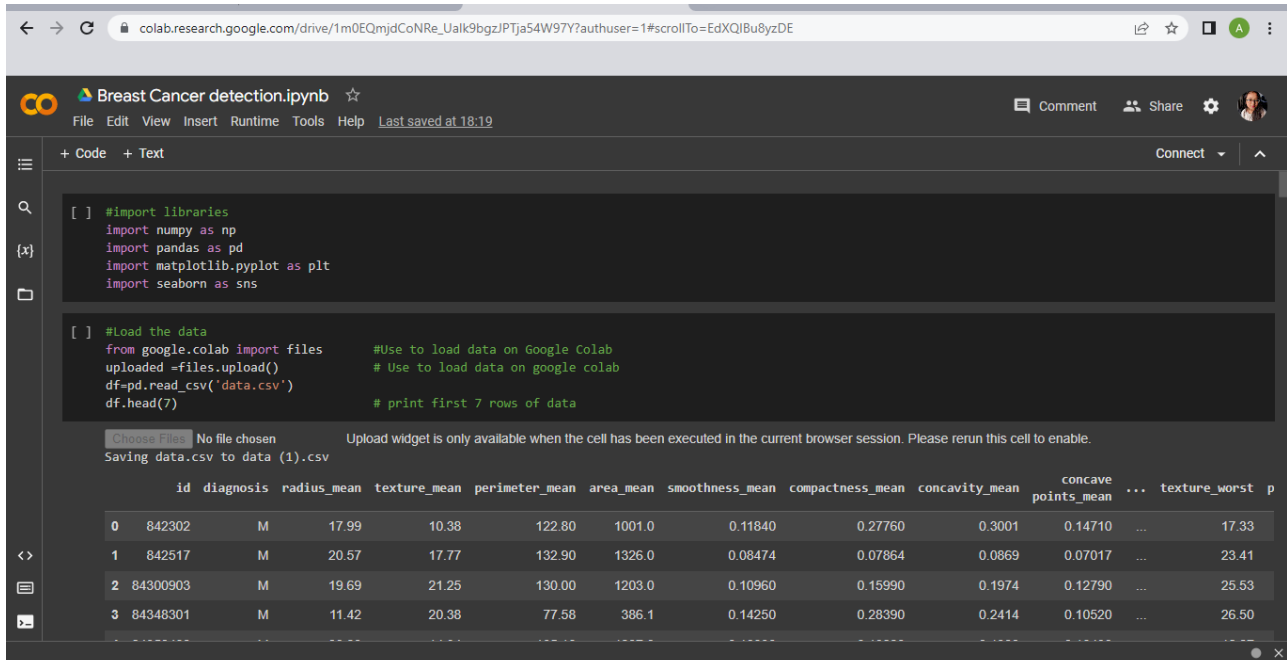
- **Continue exploring the data and get a count of all of the columns that contain empty (NaN, NAN, na) values. Notice none of the columns contain any empty values except the column named ‘*Unnamed: 32*’, which contains 569 empty values (the same number of rows in the data set, this tells me this column is completely useless)**
- **Remove the column ‘*Unnamed: 32*’ from the original data set since it adds no value.**
- **Get the new count of the number of rows and columns.**
- **Get a count of the number of patients with Malignant (M) cancerous and Benign (B) non-cancerous cells.**
- **Look at the data types to see which columns need to be transformed / encoded. We can see from the data types that all of the columns/features are numbers except for the column ‘*diagnosis*’, which is categorical data represented as an object in python.**
- ***Encode the categorical data. Change the values in the column ‘diagnosis’ from M and B to 1 and 0 respectively, then print the results.***

- *Create a pair plot. A “pairs plot” is also known as a scatter plot, in which one variable in the same data row is matched with another variable’s value.*
- *Print the new data set which now has only 32 columns. Print only the first 5 rows.*
- *Get the correlation of the columns.*
- *Visualize the correlation by creating a heat map.*
- *Now We are done exploring and cleaning the data.*
- *Now We have to set up our data for the model by first splitting the data set into a feature data set also known as the independent data set (X), and a target data set also known as the dependent data set (Y).*
- *Split the data again, but this time into 75% training and 25% testing data sets.*
- *Scale the data to bring all features to the same level of magnitude, which means the feature / independent data will be within a specific range for example 0–100 or 0–1.*
- *Create a function to hold many different models (e.g. Logistic Regression, Decision Tree Classifier, Random Forest Classifier) to make the classification. These are the models that will detect if*

a patient has cancer or not. Within this function I will also print the accuracy of each model on the training data.

- *Create the model that contains all of the models, and look at the accuracy score on the training data for each model to classify if a patient has cancer or not.*

OUTPUT SCREENSHOTS



colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgzPTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE

Breast Cancer detection.ipynb

File Edit View Insert Runtime Tools Help Last saved at 18:19

+ Code + Text

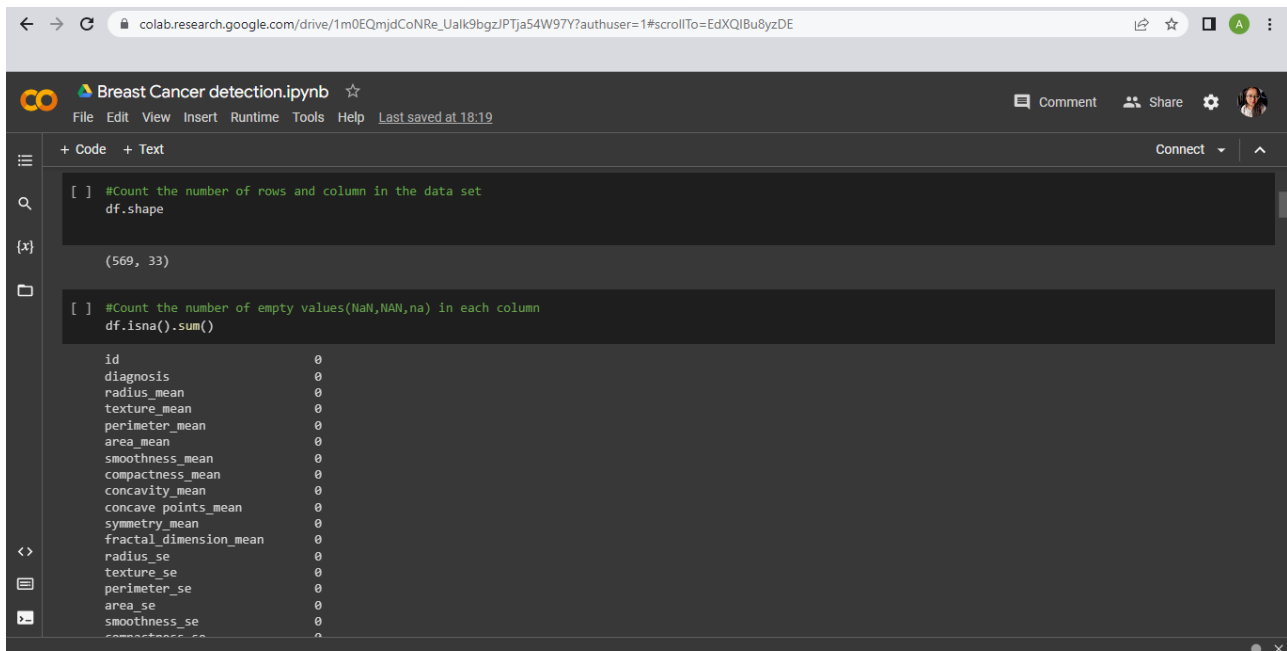
```
[ ] #import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

[ ] #Load the data
from google.colab import files #Use to load data on Google Colab
uploaded =files.upload()       # Use to load data on google colab
df=pd.read_csv('data.csv')
df.head(7)                     # print first 7 rows of data
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving data.csv to data (1).csv

| | id | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean | concavity_mean | concave points_mean | ... | texture_worst | p |
|---|----------|-----------|-------------|--------------|----------------|-----------|-----------------|------------------|----------------|---------------------|-----|---------------|---|
| 0 | 842302 | M | 17.99 | 10.38 | 122.80 | 1001.0 | 0.11840 | 0.27760 | 0.3001 | 0.14710 | ... | 17.33 | |
| 1 | 842517 | M | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.0869 | 0.07017 | ... | 23.41 | |
| 2 | 84300903 | M | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.1974 | 0.12790 | ... | 25.53 | |
| 3 | 84348301 | M | 11.42 | 20.38 | 77.58 | 386.1 | 0.14250 | 0.28390 | 0.2414 | 0.10520 | ... | 26.50 | |



colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgzPTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE

Breast Cancer detection.ipynb

File Edit View Insert Runtime Tools Help Last saved at 18:19

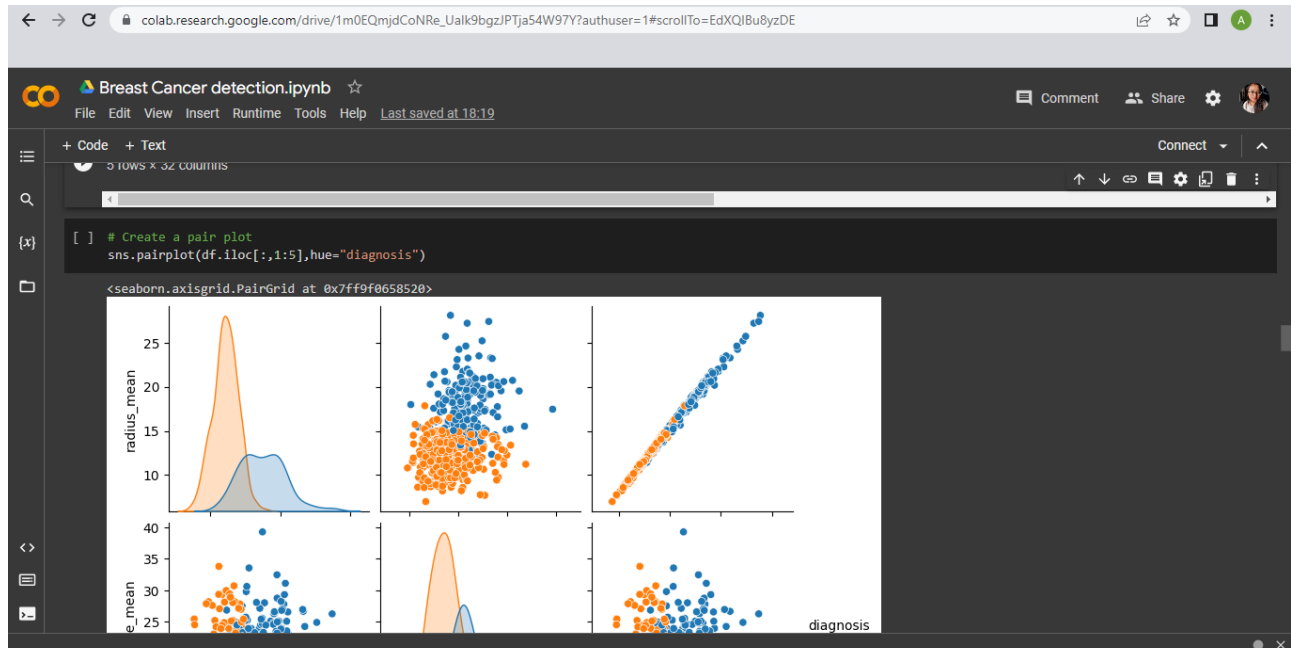
+ Code + Text

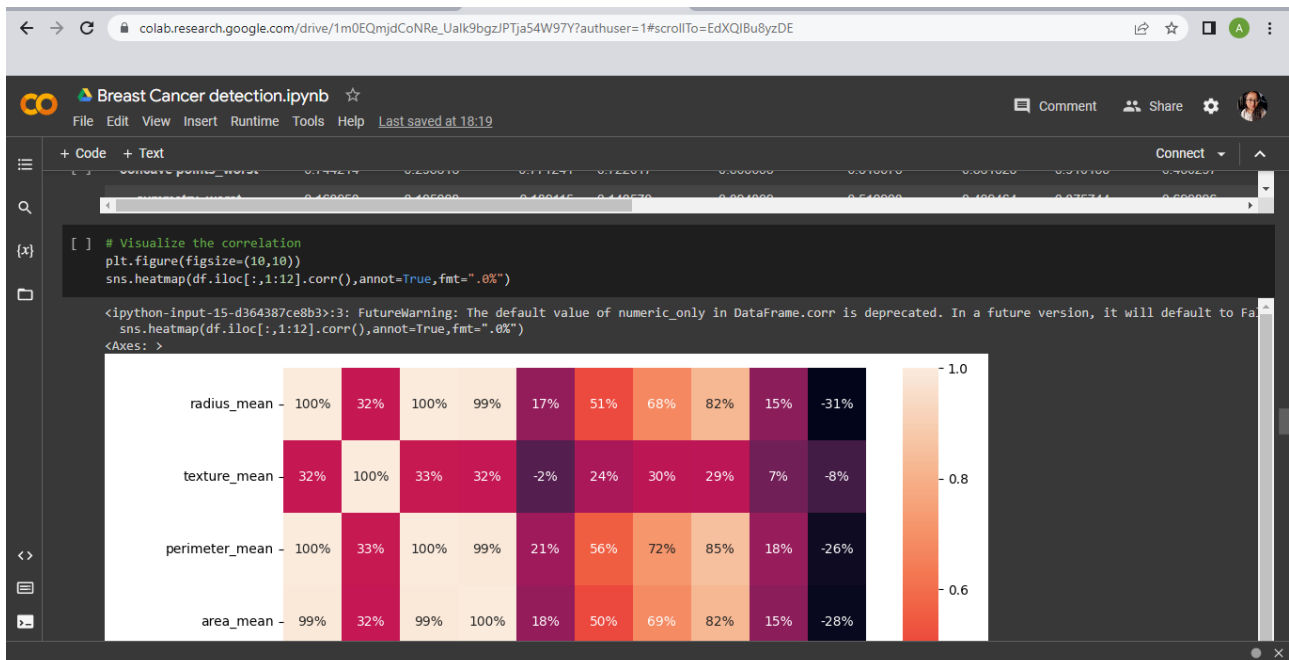
```
[ ] #Count the number of rows and column in the data set
df.shape

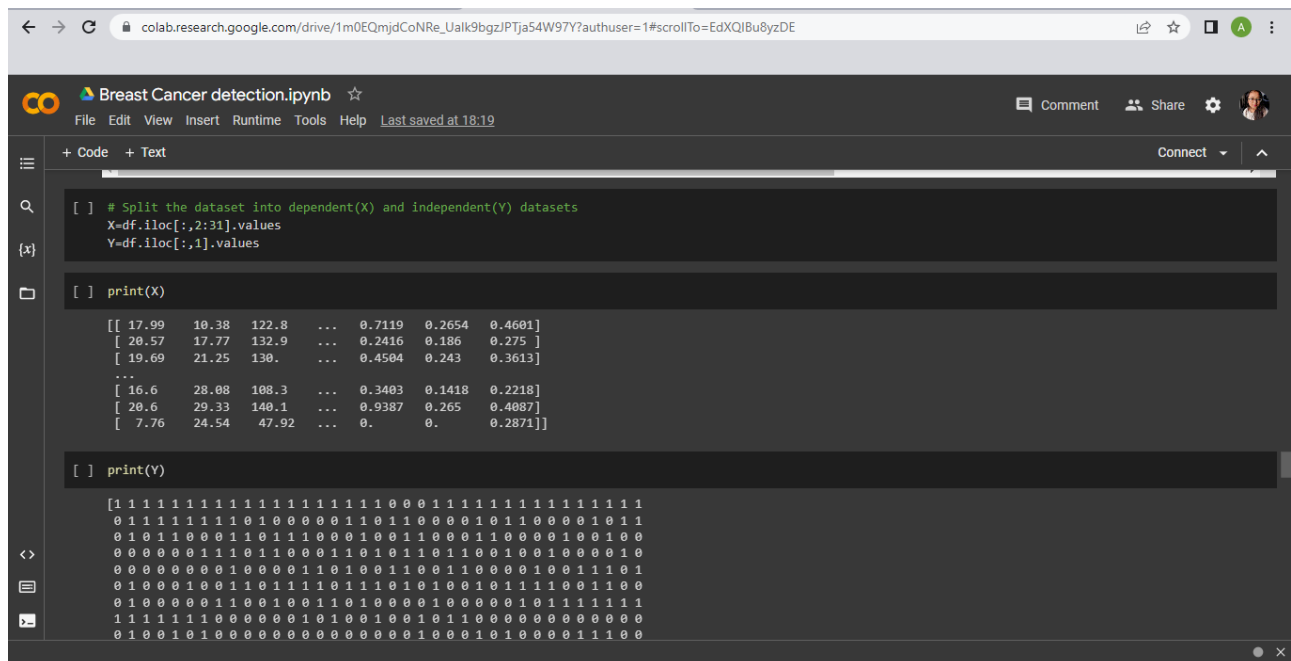
(569, 33)

[ ] #Count the number of empty values(NaN,NAN,na) in each column
df.isna().sum()
```

| | |
|------------------------|---|
| id | 0 |
| diagnosis | 0 |
| radius_mean | 0 |
| texture_mean | 0 |
| perimeter_mean | 0 |
| area_mean | 0 |
| smoothness_mean | 0 |
| compactness_mean | 0 |
| concavity_mean | 0 |
| concave points_mean | 0 |
| symmetry_mean | 0 |
| fractal_dimension_mean | 0 |
| radius_se | 0 |
| texture_se | 0 |
| perimeter_se | 0 |
| area_se | 0 |
| smoothness_se | 0 |







```

[ ]
0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 1 1
0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0
0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1
0 1 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1 1 0 0 1 1 0 0
0 1 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1
1 1 1 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0
0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 1
0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 1 1 1 0]

[ ] # splitting the dataset into training and test dataset 80% training and 20% testing
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.20, random_state=0)

[ ] # feature scaling
from sklearn.preprocessing import StandardScaler
X_train= StandardScaler().fit_transform(X_train)
X_test=StandardScaler().fit_transform(X_test)

```

```

X_train= StandardScaler().fit_transform(X_train)
[ ] X_test=StandardScaler().fit_transform(X_test)

[ ] #models / algorithms
def models(X_train,Y_train):
    #Logistic regression
    from sklearn.linear_model import LogisticRegression
    log=LogisticRegression(random_state=0)
    log.fit(X_train,Y_train)

    # Decision Tree
    from sklearn.tree import DecisionTreeClassifier
    tree=DecisionTreeClassifier(random_state=0,criterion="entropy")
    tree.fit(X_train,Y_train)

    # Random forest
    from sklearn.ensemble import RandomForestClassifier
    forest=RandomForestClassifier(random_state=0,criterion="entropy",n_estimators=10)
    forest.fit(X_train,Y_train)

    print('[0]logistic regression accuracy :',log.score(X_train,Y_train))
    print('[1]Decision tree accuracy :',tree.score(X_train,Y_train))
    print('[2] Random forest accuracy :',forest.score(X_train,Y_train))

```

colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgz/PTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE

Breast Cancer detection.ipynb

File Edit View Insert Runtime Tools Help Last saved at 18:19

+ Code + Text

```
[ ] return log,tree,forest
```

```
[ ] model= models(X_train,Y_train)
```

```
[0]logistic regression accuracy : 0.9912087912087912
[1]Decision tree accuracy : 1.0
[2] Random forest accuracy : 0.9978021978021978
```

```
[ ] # testing the models/result
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

for i in range(len(model)):
    print("Model",i)
    print(classification_report(Y_test,model[i].predict(X_test)))
    print('Accuracy : ',accuracy_score(Y_test,model[i].predict(X_test)))
```

```
Model 0
      precision    recall  f1-score   support

0     0.96      0.99      0.97         67
1     0.98      0.94      0.96         47
```

colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgz/PTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE

Breast Cancer detection.ipynb

File Edit View Insert Runtime Tools Help Last saved at 18:19

+ Code + Text

```
[ ] Model 0
      precision    recall  f1-score   support

0     0.96      0.99      0.97         67
1     0.98      0.94      0.96         47

accuracy          0.96         114
macro avg         0.97         0.96         0.96         114
weighted avg      0.97         0.96         0.96         114

Accuracy : 0.9649122807017544
Model 1
      precision    recall  f1-score   support

0     0.94      0.96      0.95         67
1     0.93      0.91      0.92         47

accuracy          0.94         114
macro avg         0.94         0.94         0.94         114
weighted avg      0.94         0.94         0.94         114

Accuracy : 0.9385964912280702
Model 2
      precision    recall  f1-score   support

0     0.96      1.00      0.98         67
1     1.00      0.94      0.97         47
```

colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgz/PTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE

Breast Cancer detection.ipynb

File Edit View Insert Runtime Tools Help Last saved at 18:19

+ Code + Text

```
[ ] weighted avg      0.94      0.94      0.94      114
[ ] Accuracy : 0.9385964912280702
[ ] Model 2
[ ] precision recall f1-score support
[ ]      0      0.96      1.00      0.98      67
[ ]      1      1.00      0.94      0.97      47
[ ] accuracy
[ ] macro avg      0.98      0.97      0.97      114
[ ] weighted avg      0.97      0.97      0.97      114
[ ] Accuracy : 0.9736842105263158
```

```
[ ] from sklearn.metrics import confusion_matrix
[ ] for i in range(len(model)):
[ ]     cm = confusion_matrix(Y_test, model[i].predict(X_test))
[ ]
[ ]     TN = cm[0][0]
[ ]     TP = cm[1][1]
[ ]     FN = cm[1][0]
[ ]     FP = cm[0][1]
[ ]
[ ]     print(cm)
[ ]     print("Model[{}] Testing Accuracy = {}".format(i, (TP + TN) / (TP + TN + FN + FP)))
[ ]     print()# Print a new line
```

colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgz/PTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE

Breast Cancer detection.ipynb

File Edit View Insert Runtime Tools Help Last saved at 18:19

+ Code + Text

```
[ ] from sklearn.metrics import confusion_matrix
[ ] for i in range(len(model)):
[ ]     cm = confusion_matrix(Y_test, model[i].predict(X_test))
[ ]
[ ]     TN = cm[0][0]
[ ]     TP = cm[1][1]
[ ]     FN = cm[1][0]
[ ]     FP = cm[0][1]
[ ]
[ ]     print(cm)
[ ]     print("Model[{}] Testing Accuracy = {}".format(i, (TP + TN) / (TP + TN + FN + FP)))
[ ]     print()# Print a new line
```

```
[[65  2]
 [ 3 44]]
Model[0] Testing Accuracy = "0.956140350877193!"

[[62  5]
 [ 3 44]]
Model[1] Testing Accuracy = "0.9298245614035088!"

[[66  1]
 [ 2 45]]
Model[2] Testing Accuracy = "0.9736842105263158!"
```

colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgz/PTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE

Breast Cancer detection.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 18:19

+ Code + Text

```
[ ] #Show other ways to get the classification accuracy & other metrics

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

for i in range(len(model)):
    print("Model ",i)
    #Check precision, recall, f1-score
    print( classification_report(Y_test, model[i].predict(X_test)) )
    #Another way to get the models accuracy on the test data
    print( accuracy_score(Y_test, model[i].predict(X_test)))
    print()#Print a new line
```

| Model 0 | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.96 | 0.97 | 0.96 | 67 |
| 1 | 0.96 | 0.94 | 0.95 | 47 |
| accuracy | | | 0.96 | 114 |
| macro avg | 0.96 | 0.95 | 0.95 | 114 |
| weighted avg | 0.96 | 0.96 | 0.96 | 114 |

0.956140350877193

colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgz/PTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE

Breast Cancer detection.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 18:19

+ Code + Text

```
[ ]
0.956140350877193

Model 1
precision recall f1-score support
0 0.95 0.93 0.94 67
1 0.90 0.94 0.92 47
accuracy 0.93 0.93 0.93 114
macro avg 0.93 0.93 0.93 114
weighted avg 0.93 0.93 0.93 114

0.9298245614035088

Model 2
precision recall f1-score support
0 0.97 0.99 0.98 67
1 0.98 0.96 0.97 47
accuracy 0.97 0.97 0.97 114
macro avg 0.97 0.97 0.97 114
weighted avg 0.97 0.97 0.97 114

0.9736842105263158
```



```
colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgzPTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE
```

Breast Cancer detection.ipynb

File Edit View Insert Runtime Tools Help Last saved at 18:19

+ Code + Text

```
[ ] # prediction of random-forest
pred=model[2].predict(X_test)
print('Predicted values:')
print(pred)
print('Actual values:')
print(Y_test)
```

Predicted values:

```
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0
1 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0
1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
1 1 0]
```

Actual values:

```
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0
1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
1 1 0]
```

```
[ ] from joblib import dump
dump(model[2],"Cancer_prediction.joblib")
```

```
['Cancer_prediction.joblib']
```

```
colab.research.google.com/drive/1m0EQmjdCoNRe_Ualk9bgzPTja54W97Y?authuser=1#scrollTo=EdXQlBu8yzDE
```

Breast Cancer detection.ipynb

File Edit View Insert Runtime Tools Help Last saved at 18:19

+ Code + Text

```
[ ] print(Y_test)
```

Predicted values:

```
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0
1 0 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 0
1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
1 1 0]
```

Actual values:

```
[1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 0 1 0 1 0
1 0 1 1 0 1 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 0 1
1 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 1 0
1 1 0]
```

```
[ ] from joblib import dump
dump(model[2],"Cancer_prediction.joblib")
```

```
['Cancer_prediction.joblib']
```

CONCLUSION

In this minor project, different types of models are reviewed and their accuracies are computed and compared with each other, so that the best cancer prediction model can be used by doctors in real life to identify breast cancer relatively faster than previous methods. Above examined study, proposed that the Random Forest Classification algorithm is proficiently utilized and efficient for detection of breast cancer as compared to Decision tree and Logistic Regression algorithms.

REFERENCES

- ‘WHO | Breast cancer’, *WHO*.
<http://www.who.int/cancer/prevention/diagnosis-screening/breast-cancer/en/>
- H. Asri, H. Mousannif, H.A. Moatassime, T. Noel
‘Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis’
Procedia Computer Science, 83 (2016), pp. 1064-1069
- <https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data>
- Fabian Pedregosa, *et al.*
“Scikit-learn: Machine Learning in Python”
Journal of Machine Learning Research., 12 (2011), pp. 2825-2830