BREAST CANCER PREDICTION
END TERM REPORT

by

AMAN PANDEY
&
ADITYA RAJ JHA

Section: KM118
Roll Numbers: RKM118B51 & RKM118A17

Department of Intelligent Systems,

School of Computer Science Engineering,

Lovely Professional University, Jalandhar

November, 2022

STUDENT DECLARATION

This is to declare that this report has been written by us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. We aver that if any part of the report is found to be copied, we are shall take full responsibility for it.

A Pandey

Aman Pandey
RKM118B51

Aditya

Aditya Raj Jha
RKM118A17

Place: Lovely Professional University, Jalandhar
Date: 08th November 2022

TABLE OF CONTENTS

BONAFIDE CERTIFICATE

Certified that this project report “Breast cancer prediction using machine learning” is the bonafide work of Aman Pandey and Aditya Raj verma who carried out the project work under my supervision.

Dr. Dhanpratap Singh
Associate professor
25706
Intelligence System 1

INTRODUCTION

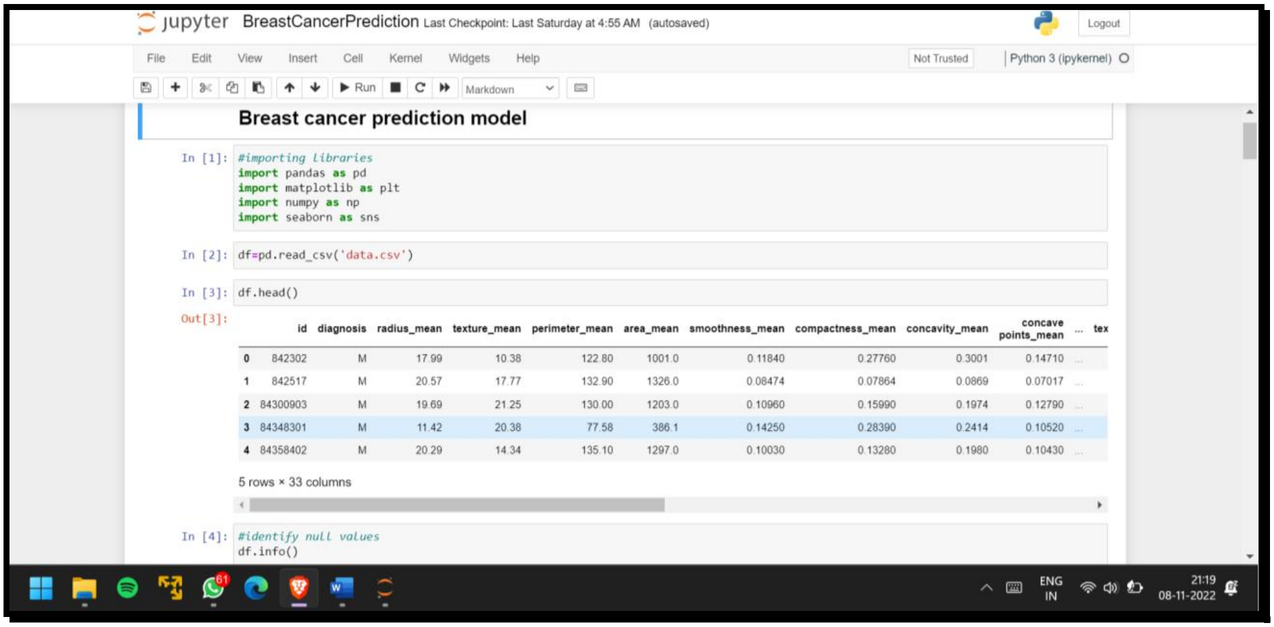
Breast cancer has become the most recurrent type of health issue among women especially for women in middle age. Early detection of breast cancer can help women cure this disease and death rate can be reduced. In the present-day scenario, to observe breast cancer mammograms are used and they are known be the most effective scanning technique. In this prediction model the detection of cancer cells is done by machine learning technique.

Machine learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available. The process of learning begins with observations or data, such as examples, direct experience, or instruction, to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly

Combining multiple risk factors in modelling for breast cancer prediction could help the early diagnosis of the disease with necessary care plans. Collection, storage, and management of different data and intelligent systems based on multiple factors for predicting breast cancer are effective in disease management.

SYNTAX AND SCREENSHOTS

Importing Libraries and csv file


jupyter BreastCancerPrediction Last Checkpoint: Last Saturday at 4:55 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
ENG IN
21:19
08-11-2022

Breast cancer prediction model

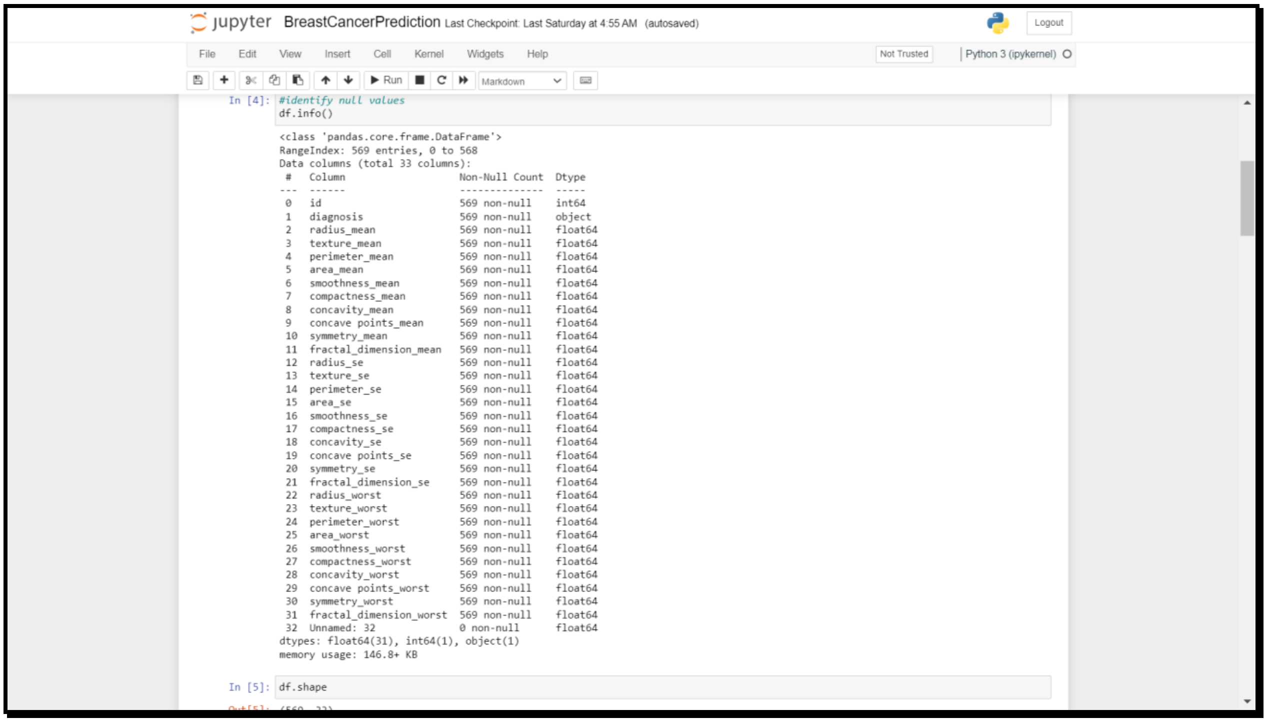
```
In [1]: #importing libraries
        import pandas as pd
        import matplotlib as plt
        import numpy as np
        import seaborn as sns

In [2]: df=pd.read_csv('data.csv')

In [3]: df.head()
Out[3]:
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	tex
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 33 columns

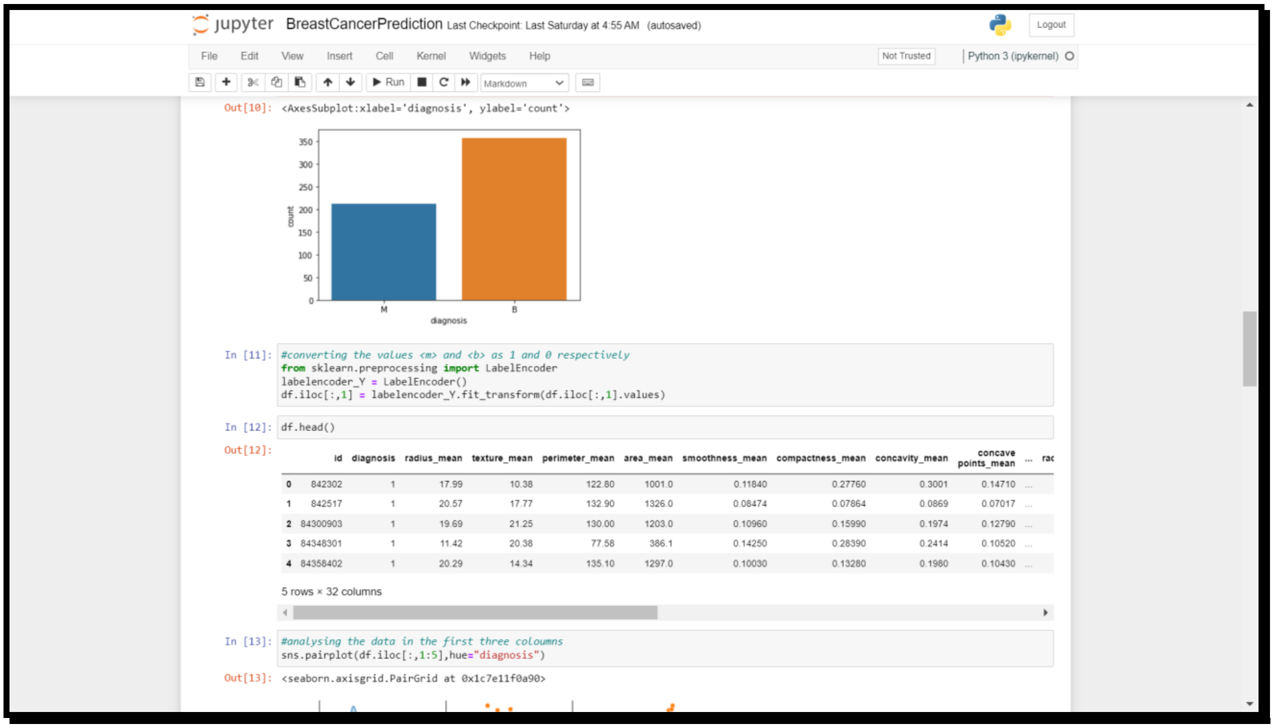
```
In [4]: #identify null values
        df.info()
```

Checking the null values resent in attributes


jupyter BreastCancerPrediction Last Checkpoint: Last Saturday at 4:55 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [4]: #identify null values
        df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   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
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB

In [5]: df.shape
```

Representing malignant and benignant samples in for of graphs, and changing ‘M’ and ‘B’ into 1 and 0 respectively to for a correlation.


jupyter BreastCancerPrediction Last Checkpoint: Last Saturday at 4:55 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown
350
300
250
200
150
100
50
0
count
M
B
diagnosis

```
Out[10]: <AxesSubplot:xlabel='diagnosis', ylabel='count'>
```

```
In [11]: #converting the values <m> and <b> as 1 and 0 respectively
         from sklearn.preprocessing import LabelEncoder
         labelencoder_Y = LabelEncoder()
         df.iloc[:,1] = labelencoder_Y.fit_transform(df.iloc[:,1].values)
```

```
In [12]: df.head()
```

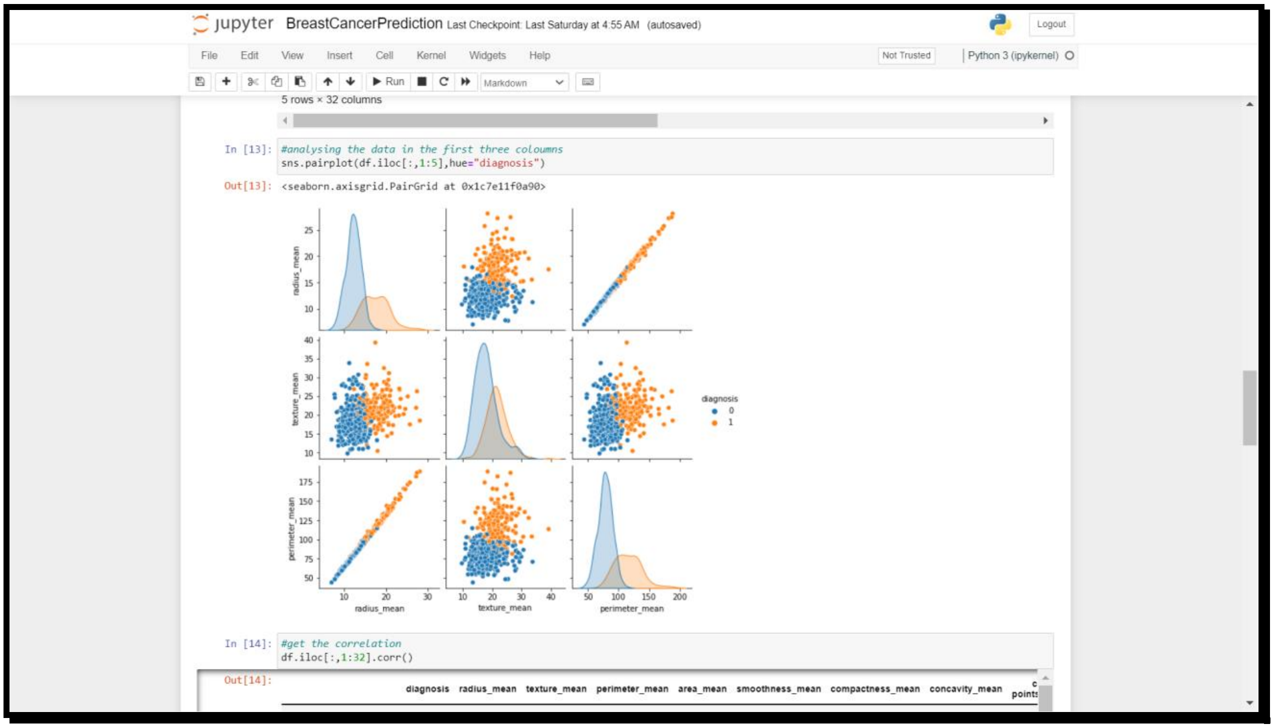
Out[12]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	rac
0	842302	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 32 columns

```
In [13]: #analysing the data in the first three coloumns
         sns.pairplot(df.iloc[:,1:5],hue="diagnosis")
Out[13]: <seaborn.axisgrid.PairGrid at 0x1c7e11f0a90>
```

Generating a pairplot



5 rows × 32 columns

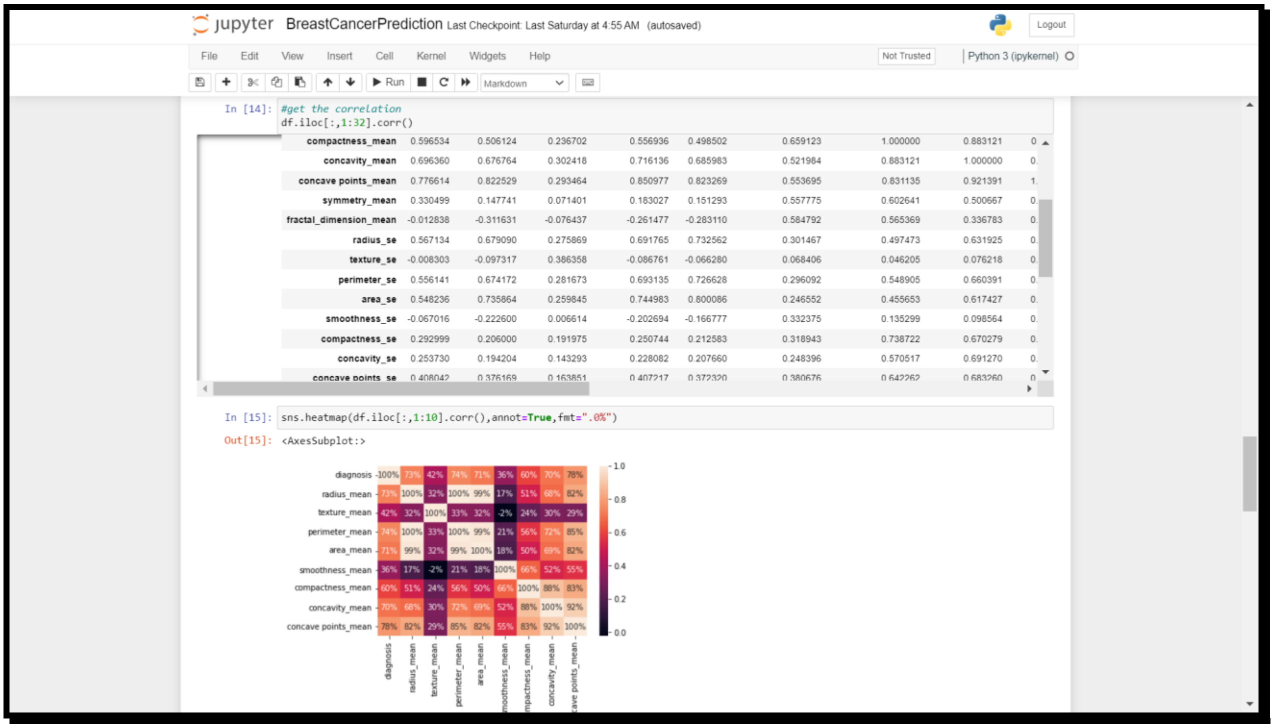
```
In [13]: #analysing the data in the first three coloumns
         sns.pairplot(df.iloc[:,1:5],hue="diagnosis")
Out[13]: <seaborn.axisgrid.PairGrid at 0x1c7e11f0a90>
```

```
In [14]: #get the correlation
         df.iloc[:,1:32].corr()
```

Out[14]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	c points

Checking the correlation among all attributes



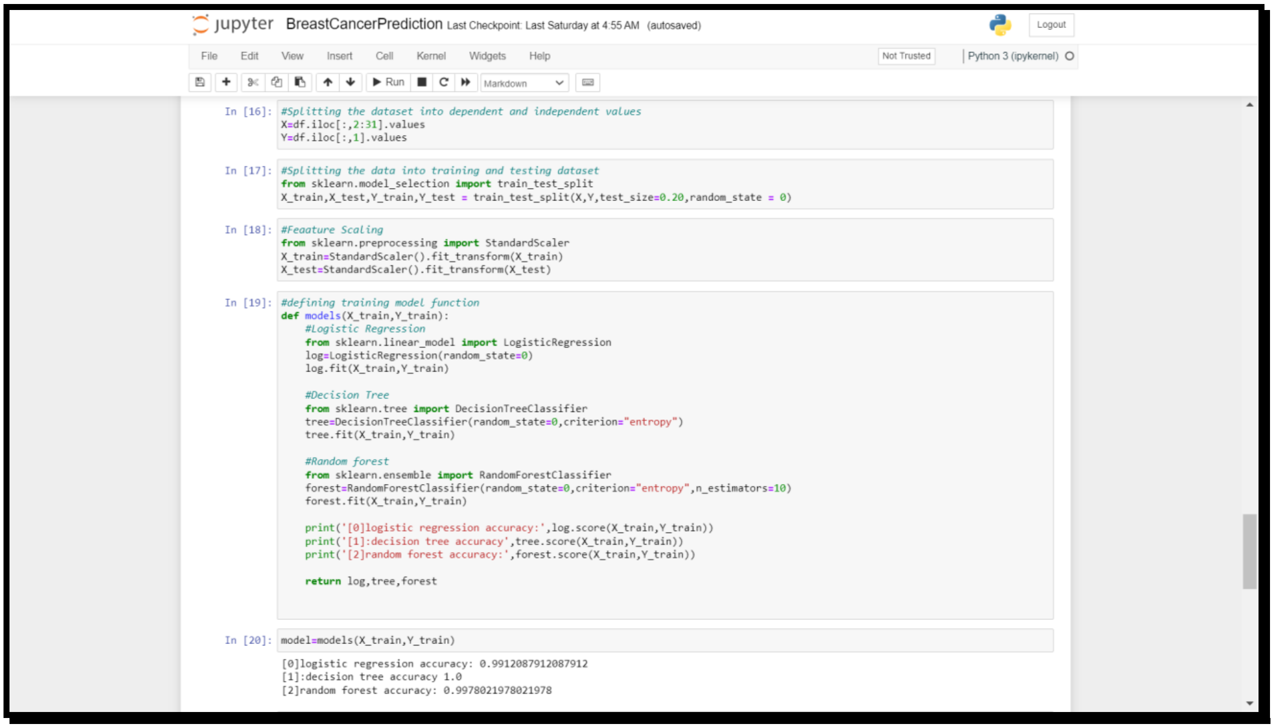
```
In [14]: #get the correlation
         df.iloc[:,1:32].corr()
```

compactness_mean	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121	0
concavity_mean	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000	0.
concave points_mean	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391	1.
symmetry_mean	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667	0.
fractal_dimension_mean	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783	0.
radius_se	0.567134	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925	0.
texture_se	-0.008303	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218	0.
perimeter_se	0.556141	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391	0.
area_se	0.548236	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.617427	0.
smoothness_se	-0.067016	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564	0.
compactness_se	0.292999	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722	0.670279	0.
concavity_se	0.253730	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270	0.
concave points_se	0.408042	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262	0.683260	0

```
In [15]: sns.heatmap(df.iloc[:,1:10].corr(),annot=True,fmt=".0%")

Out[15]: <AxesSubplot:>
```

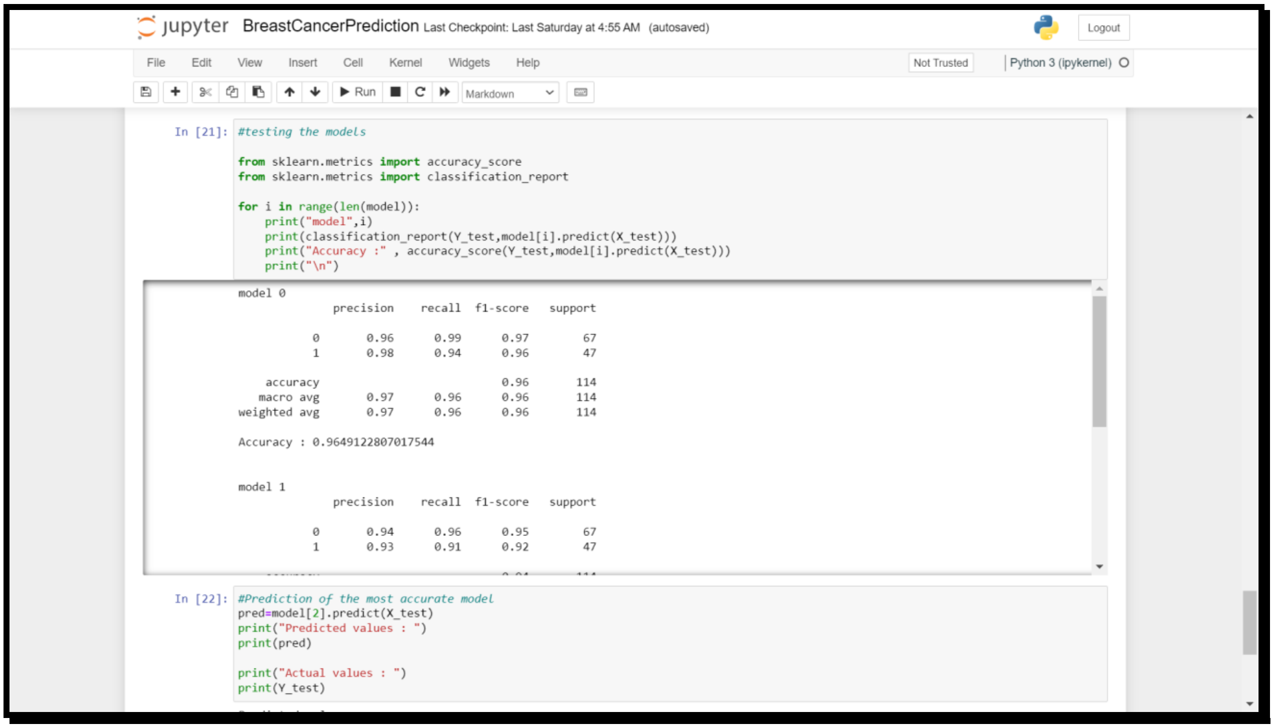
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	moothness_mean	mpactness_mean	concavity_mean	cave points_mean
diagnosis	100%	73%	42%	74%	71%	36%	60%	70%	78%
radius_mean	73%	100%	32%	100%	99%	17%	51%	68%	82%
texture_mean	42%	32%	100%	33%	32%	-2%	24%	30%	29%
perimeter_mean	74%	100%	33%	100%	99%	21%	56%	72%	85%
area_mean	71%	99%	32%	99%	100%	18%	50%	69%	82%
smoothness_mean	36%	17%	-2%	21%	18%	100%	66%	52%	55%
compactness_mean	60%	51%	24%	56%	50%	66%	100%	88%	83%
concavity_mean	70%	68%	30%	72%	69%	52%	88%	100%	92%
concave points_mean	78%	82%	29%	85%	82%	55%	83%	92%	100%

Splitting the data for training and testing purposes



```
In [16]: #Splitting the dataset into dependent and independent values
         X=df.iloc[:,2:31].values
         Y=df.iloc[:,1].values

In [17]: #Splitting the data into training and testing 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 [18]: #Feaature Scaling
         from sklearn.preprocessing import StandardScaler
         X_train=StandardScaler().fit_transform(X_train)
         X_test=StandardScaler().fit_transform(X_test)

In [19]: #defining training model function
         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

In [20]: 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

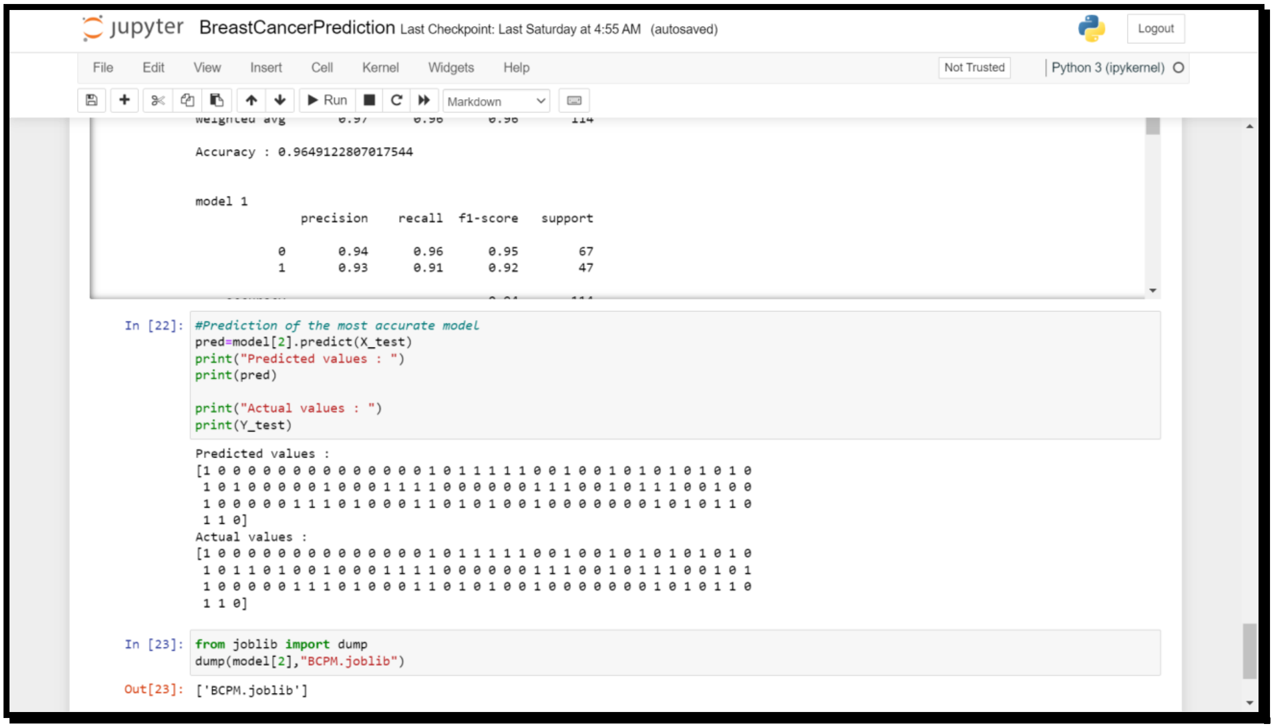

jupyter BreastCancerPrediction Last Checkpoint: Last Saturday at 4:55 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
In [21]: #testing the models

         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)))
             print("\n")
```

```
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
```

```
In [22]: #Prediction of the most accurate model
         pred=model[2].predict(X_test)
         print("Predicted values : ")
         print(pred)

         print("Actual values : ")
         print(Y_test)
```

Prediction using the test cases


jupyter BreastCancerPrediction Last Checkpoint: Last Saturday at 4:55 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Markdown

```
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
```

```
In [22]: #Prediction of the most accurate model
         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 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 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 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 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]
```

```
In [23]: from joblib import dump
         dump(model[2],"BCPM.joblib")

Out[23]: ['BCPM.joblib']
```

ALGORITHMS USED

Logistic regression:

It is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Random Forest Classifier:

The Random Forest classifier creates a set of decision trees from a randomly selected subset of the training set. It is basically a set of decision trees (DT) from a randomly selected subset of the training set and then It collects the votes from different decision trees to decide the final prediction.

Decision tree classifier:

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

CONCLUSION

Combining multiple risk factors in modelling for breast cancer prediction could help the early diagnosis of the disease with necessary care plans. Collection, storage, and management of different data and intelligent systems based on multiple factors for predicting breast cancer are effective in disease management.

This particular model considers value from a dataset representing various values and attributes of potential patients. Upon being trained under three different algorithms , the models proves itself to be pretty effective and accurate . The accuracy obtained in the results are as follows

Logistic regression – 96.49%

Decision tree – 9.86%

Random forest classifier – 97.34%

The training model using random forest classifier training model turns out to be the best algorithm for this particular case , coming up with 97.34% accuracy rate.

BIBLIOGRAPHY

- https://www.javatpoint.com/machine-learning
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9175124
- https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data
- https://www.youtube.com/watch?v=HXnDyrraRb0&t=789s&ab_channel=CodeForLife

PROJECT GITHUB LINK

https://github.com/amanpandey-03/BreastCancerPrediction