# *How to predict breast cancer?*

***By Fatima Vahidnezhad***

***DBDA.X427.(1) Python for Machine Learning and Artificial Intelligence_ Essentials***

***Aug 2020***

## 1. Introduction

Dataset consists of information about 569 Women that they had breast cancer with a malignant or benign tumor. Data has 32 features (columns) which determine the symptoms of the illness. "Diagnosis" column is a categorical column, and it is considered as a target column. It contains two values of a malignant and benign tumor. Other features have numeric values.

## 2. Task:

In this project, I want to predict breast cancer and increase the accuracy of my computation.

## 3. The link address of the dataset:

Dataset is in the csv format and the link of downloading the dataset is:

https://www.kaggle.com/uciml/breast-cancer-wisconsin-data

## 4. List of Variables:

1) ID number
2) Diagnosis: The diagnosis of breast tissues (M = malignant, B = benign)
3) Radius_mean: mean of distances from center to points on the perimeter
4) Texture_mean: standard deviation of gray-scale values
5) Perimeter_mean: mean size of the core tumor
6) Area_mean
7) Smoothness_mean: mean of local variation in radius lengths
8) Compactness_mean: mean of perimeter^2 / area - 1.0
9) Concavity_mean: mean of severity of concave portions of the contour
10) Concave_points_mean: mean for number of concave portions of the contour
11) Symmetry_mean
12) Fractal_dimension_mean: mean for "coastline approximation" - 1
13) Radius_se: standard error for the mean of distances from center to points on the perimeter
14) Texture_se: standard error for standard deviation of gray-scale values
15) Perimeter_se
16) Area_se
17) Smoothness_se: standard error for local variation in radius lengths
18) Compactness_se: standard error for perimeter^2 / area - 1.0
19) Concavity_se: standard error for severity of concave portions of the contour
20) concave points_se: standard error for number of concave portions of the contour
21) symmetry_se
22) fractal dimension_se: standard error for "coastline approximation" - 1
23) radius_worst: "worst" or largest mean value for mean of distances from center to points on the perimeter
24) texture_worst: "worst" or largest mean value for standard deviation of gray-scale values
25) perimeter_worst
26) area_worst

27) smoothness_worst: "worst" or largest mean value for local variation in radius lengths
28) compactness_worst: "worst" or largest mean value for perimeter^2 / area - 1.0
29) concavity_worst: "worst" or largest mean value for severity of concave portions of the contour
30) Fractal_dimension_worst
31) Symmetry_worst
32) Concave_points_worst

## 5. Requirements

We should import these libraries:

In [1]:
```
import numpy as np
import pandas as pd
import os
import seaborn as sb
import matplotlib.pyplot as plt
from scipy import stats # to remove outliers
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score,confusion_matrix
from sklearn.metrics import accuracy_score
```

## 6. Description of the Python program.

This program consists of three steps:

***First***: preparing the dataset:

1. Finding and removing missing values from all columns.
2. Finding outliers and removing them
3. Make a dataset from numeric columns

***Second***: Visualizing features:

1. Plotting a pie chart from the seaborn library to show the per cent of patients with a benign and malignant tumor.
2. Plotting a violin plot from the seaborn library to show the quartile of features.

***Third***: computing and visualizing the output:

1. Finding the correlation between variables and plotting a heatmap matrix from the seaborn library.
2. Plotting the correlation between variables and target column (Diagnosis column).

3. **Select k-best algorithm** :selecting features with a high score and p-value less than 0.05 to make sure that 95 per cent of columns have a relationship with the target column (Diagnosis column) by using the k-best algorithm from the sci-kit-learn library.
4. Split the dataset to two groups of train and test set.
5. **Random forest classifier:** predicting target column from selected variables by using the algorithm of random forest classifier from the sci-kit-learn library.
6. Computing the number of errors and the accuracy of the model after training the dataset.
7. Predicting target column from all variables by using the algorithm of random forest classifier and computing the accuracy of the model.
8. Comparing the accuracy of two models from # number 6 and 7.

## 7. Screenshots of the program output

### Data preparation

Step1: reading data and showing features and type of every column in the dataset:

The mean, standard error and "worst" or largest (mean of the three largest values) of features were computed for each sample, resulting in 33 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. Furthermore, the last column is empty, and we do not have missing values in the other columns. The type of features consists of int, float and object.

In [4]:

```
data = pd.read_csv('breast cancer/data.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
id                         569 non-null int64
diagnosis                  569 non-null object
radius_mean                569 non-null float64
texture_mean               569 non-null float64
perimeter_mean             569 non-null float64
area_mean                  569 non-null float64
smoothness_mean            569 non-null float64
compactness_mean           569 non-null float64
concavity_mean             569 non-null float64
concave points_mean        569 non-null float64
symmetry_mean              569 non-null float64
fractal_dimension_mean     569 non-null float64
radius_se                  569 non-null float64
texture_se                 569 non-null float64
perimeter_se               569 non-null float64
area_se                    569 non-null float64
smoothness_se              569 non-null float64
compactness_se             569 non-null float64
concavity_se               569 non-null float64
concave points_se          569 non-null float64
symmetry_se                569 non-null float64
fractal_dimension_se       569 non-null float64
radius_worst               569 non-null float64
texture_worst              569 non-null float64
perimeter_worst            569 non-null float64
area_worst                 569 non-null float64
smoothness_worst           569 non-null float64
compactness_worst          569 non-null float64
concavity_worst            569 non-null float64
concave points_worst       569 non-null float64
symmetry_worst             569 non-null float64
fractal_dimension_worst    569 non-null float64
Unnamed: 32                0 non-null float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

Step2: finding min, max, std and the quartile of each column. For example, 75 percent of data in ‘area_mean’ column are less than 782.7 but maximum value in this column is 2501. It means than it has outliers.

```
In [14]: new_data = data.drop("id",axis = 'columns')
         new_data.describe()
```

Out[14]:

| | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean | concavity_mean | concave points_mean | symmetry_mean |
|---|---|---|---|---|---|---|---|---|---|
| count | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 |
| mean | 14.127292 | 19.289649 | 91.969033 | 654.889104 | 0.096360 | 0.104341 | 0.088799 | 0.048919 | 0.181162 |
| std | 3.524049 | 4.301036 | 24.298981 | 351.914129 | 0.014064 | 0.052813 | 0.079720 | 0.038803 | 0.027414 |
| min | 6.981000 | 9.710000 | 43.790000 | 143.500000 | 0.052630 | 0.019380 | 0.000000 | 0.000000 | 0.106000 |
| 25% | 11.700000 | 16.170000 | 75.170000 | 420.300000 | 0.086370 | 0.064920 | 0.029560 | 0.020310 | 0.161900 |
| 50% | 13.370000 | 18.840000 | 86.240000 | 551.100000 | 0.095870 | 0.092630 | 0.061540 | 0.033500 | 0.179200 |
| 75% | 15.780000 | 21.800000 | 104.100000 | 782.700000 | 0.105300 | 0.130400 | 0.130700 | 0.074000 | 0.195700 |
| max | 28.110000 | 39.280000 | 188.500000 | 2501.000000 | 0.163400 | 0.345400 | 0.426800 | 0.201200 | 0.304000 |

8 rows × 31 columns

it seems we have outliers

```
In [14]: new_data = data.drop("id",axis = 'columns')
         new_data.describe()
```

Out[14]:

missing values

| perimeter_worst | area_worst | smoothness_worst | compactness_worst | concavity_worst | concave points_worst | symmetry_worst | fractal_dimension_worst | Unnamed: 32 |
|---|---|---|---|---|---|---|---|---|
| 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 569.000000 | 0.0 |
| 107.261213 | 880.583128 | 0.132369 | 0.254265 | 0.272188 | 0.114606 | 0.290076 | 0.083946 | NaN |
| 33.602542 | 569.356993 | 0.022832 | 0.157336 | 0.208624 | 0.065732 | 0.061867 | 0.018061 | NaN |
| 50.410000 | 185.200000 | 0.071170 | 0.027290 | 0.000000 | 0.000000 | 0.156500 | 0.055040 | NaN |
| 84.110000 | 515.300000 | 0.116600 | 0.147200 | 0.114500 | 0.064930 | 0.250400 | 0.071460 | NaN |
| 97.660000 | 686.500000 | 0.131300 | 0.211900 | 0.226700 | 0.099930 | 0.282200 | 0.080040 | NaN |
| 125.400000 | 1084.000000 | 0.146000 | 0.339100 | 0.382900 | 0.161400 | 0.317900 | 0.092080 | NaN |
| 251.200000 | 4254.000000 | 0.222600 | 1.058000 | 1.252000 | 0.291000 | 0.663800 | 0.207500 | NaN |

Step3: Finding the number of missing values:

According the below code, all columns do not have missing values except “Unnamed: 32” column.

```
In [15]: np.sum(pd.isnull(data))

Out[15]: 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
         compactness_se             0
         concavity_se               0
         concave points_se          0
         symmetry_se                0
         fractal_dimension_se       0
         radius_worst               0
         texture_worst              0
         perimeter_worst            0
         area_worst                 0
         smoothness_worst           0
         compactness_worst          0
         concavity_worst            0
         concave points_worst       0
         symmetry_worst             0
         fractal_dimension_worst    0
         Unnamed: 32              569
         dtype: int64
```

Step 4: removing ‘Unnamed: 32” column and checking the name of columns in the dataset after this change:

```
In [18]: #remove unnamed column:
         data.dropna(axis =1,inplace = True)

In [19]: data.columns

Out[19]: Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
                'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
                'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
                'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
                'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
                'fractal_dimension_se', 'radius_worst', 'texture_worst',
                'perimeter_worst', 'area_worst', 'smoothness_worst',
                'compactness_worst', 'concavity_worst', 'concave points_worst',
                'symmetry_worst', 'fractal_dimension_worst'],
               dtype='object')
```

Step 5: choosing numeric features and removing outliers:

```
In [22]:
#remove outliers
df = df[(np.abs(stats.zscore(df))<3).all(axis = 1)]
df.describe()
```

Out[22]:

| | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean | concavity_mean | concave points_mean | symmetry_mean |
|---|---|---|---|---|---|---|---|---|---|
| count | 495.000000 | 495.000000 | 495.000000 | 495.000000 | 495.000000 | 495.000000 | 495.000000 | 495.000000 | 495.000000 |
| mean | 13.868883 | 18.889778 | 89.931939 | 622.613333 | 0.095075 | 0.094741 | 0.074145 | 0.043306 | 0.177602 |
| std | 3.042868 | 3.966250 | 20.795084 | 286.903247 | 0.012822 | 0.041071 | 0.061694 | 0.032550 | 0.023258 |
| min | 6.981000 | 9.710000 | 43.790000 | 143.500000 | 0.062510 | 0.019380 | 0.000000 | 0.000000 | 0.116700 |
| 25% | 11.745000 | 16.000000 | 75.475000 | 424.800000 | 0.085345 | 0.062310 | 0.027420 | 0.019485 | 0.160800 |
| 50% | 13.210000 | 18.580000 | 85.630000 | 538.900000 | 0.094620 | 0.085490 | 0.053080 | 0.030700 | 0.177100 |
| 75% | 15.290000 | 21.460000 | 100.250000 | 718.050000 | 0.103700 | 0.120150 | 0.107250 | 0.062770 | 0.193000 |
| max | 23.270000 | 30.720000 | 152.100000 | 1686.000000 | 0.137100 | 0.228400 | 0.317400 | 0.156200 | 0.254000 |

8 rows × 30 columns

removed outliers

```
In [46]:
print(data.shape)
print(df.shape)
num_outliers = data.shape[0]-df.shape[0]
print("%d rows were outliers and they were removed from dataset." %num_outliers)
```

```
(569, 32)
(495, 30)
74 rows were outliers and they were removed from dataset.
```

Step 6: concatenating numeric columns with binary variable (Diagnosis column). The new dataset was named “new_data” and it will be used for visualization, because it does not have outliers and missing values.

```
In [63]:
print(df.index)
new_data = pd.concat([data['diagnosis'] , df],axis = 'columns')
new_data.dropna(how = 'any',axis = 'rows',inplace = True)
new_data
```

```
Int64Index([  1,   2,   4,   5,   6,   7,   8,  10,  11,  13,
             ...
             553, 554, 555, 556, 558, 560, 563, 564, 565, 566],
            dtype='int64', length=495)
```

Out[63]:

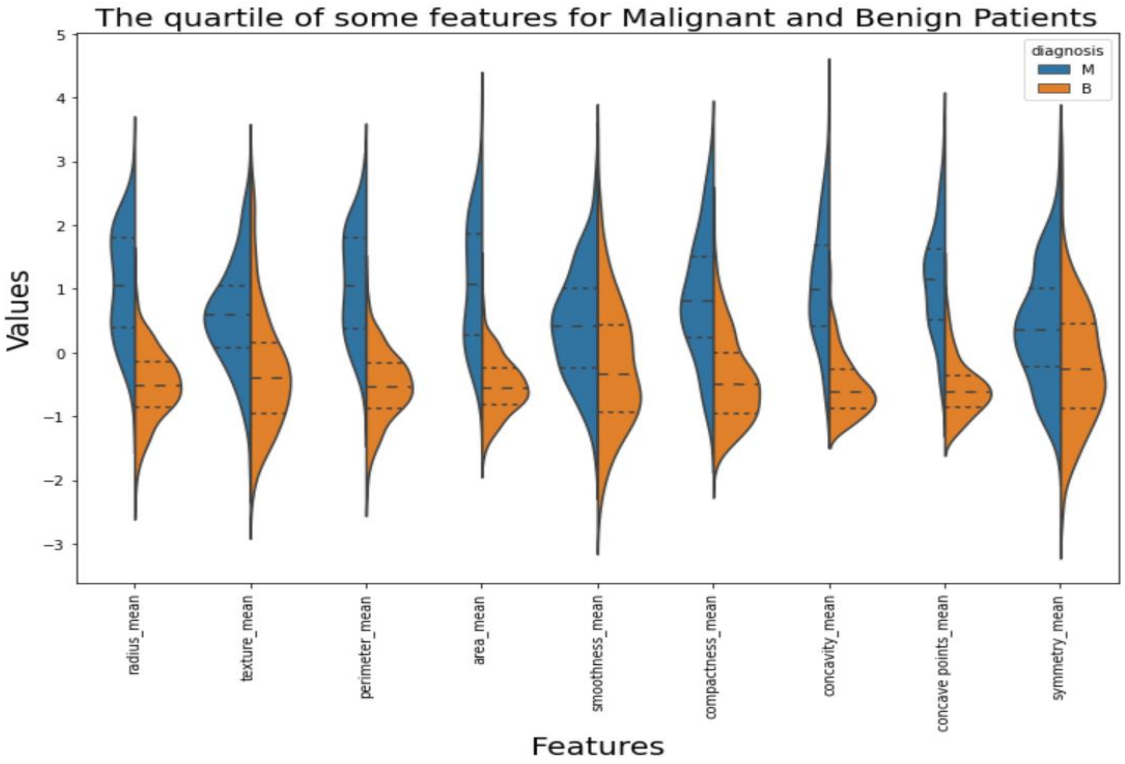
| | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mean | compactness_mean | concavity_mean | concave points_mean | symmetry |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | M | 20.57 | 17.77 | 132.90 | 1326.0 | 0.08474 | 0.07864 | 0.08690 | 0.07017 | |
| 2 | M | 19.69 | 21.25 | 130.00 | 1203.0 | 0.10960 | 0.15990 | 0.19740 | 0.12790 | |
| 4 | M | 20.29 | 14.34 | 135.10 | 1297.0 | 0.10030 | 0.13280 | 0.19800 | 0.10430 | |
| 5 | M | 12.45 | 15.70 | 82.57 | 477.1 | 0.12780 | 0.17000 | 0.15780 | 0.08089 | |
| 6 | M | 18.25 | 19.98 | 119.60 | 1040.0 | 0.09463 | 0.10900 | 0.11270 | 0.07400 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 560 | B | 14.05 | 27.15 | 91.38 | 600.4 | 0.09929 | 0.11260 | 0.04462 | 0.04304 | |
| 563 | M | 20.92 | 25.09 | 143.00 | 1347.0 | 0.10990 | 0.22360 | 0.31740 | 0.14740 | |
| 564 | M | 21.56 | 22.39 | 142.00 | 1479.0 | 0.11100 | 0.11590 | 0.24390 | 0.13890 | |
| 565 | M | 20.13 | 28.25 | 131.20 | 1261.0 | 0.09780 | 0.10340 | 0.14400 | 0.09791 | |
| 566 | M | 16.60 | 28.08 | 108.30 | 858.1 | 0.08455 | 0.10230 | 0.09251 | 0.05302 | |

495 rows × 31 columns

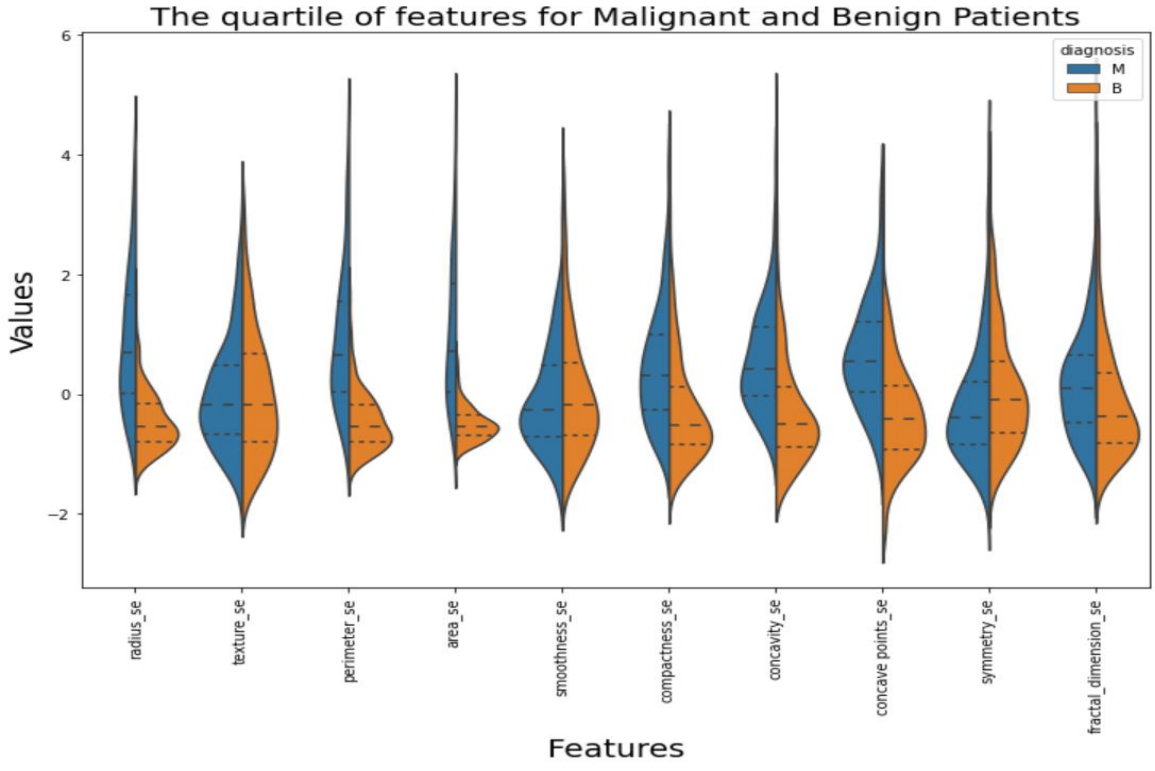
in new_data outliers and NA rows removed

## Data visualization

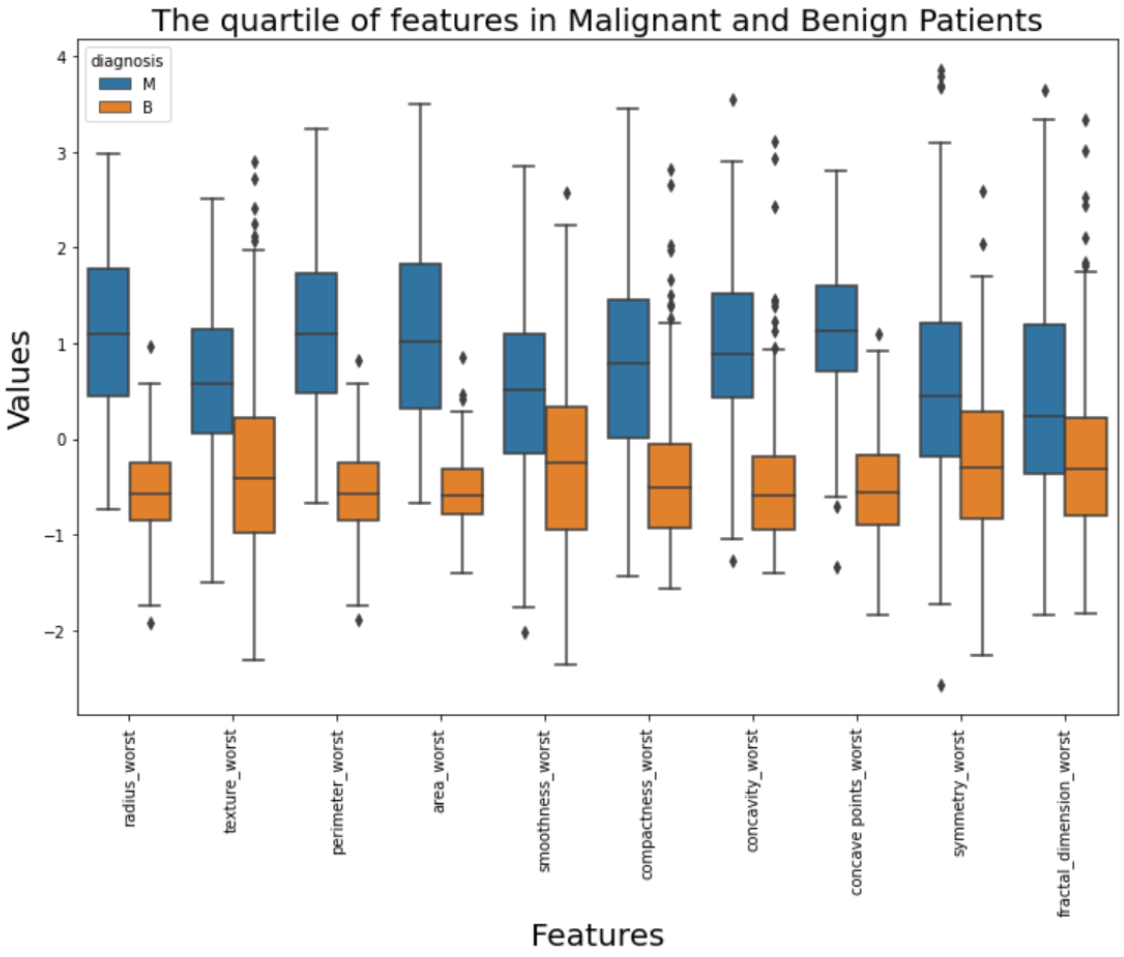
Step 1: showing 10 first features in one plot: approximately, some features with benign tumor have normal distribution.

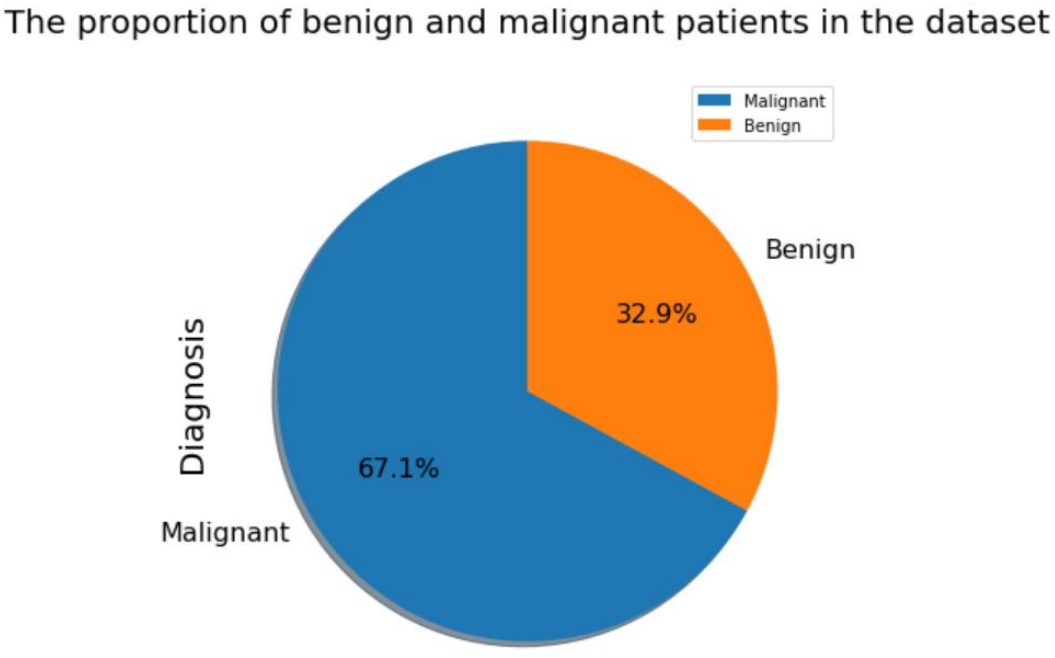



Showing the next 10 features in a plot:



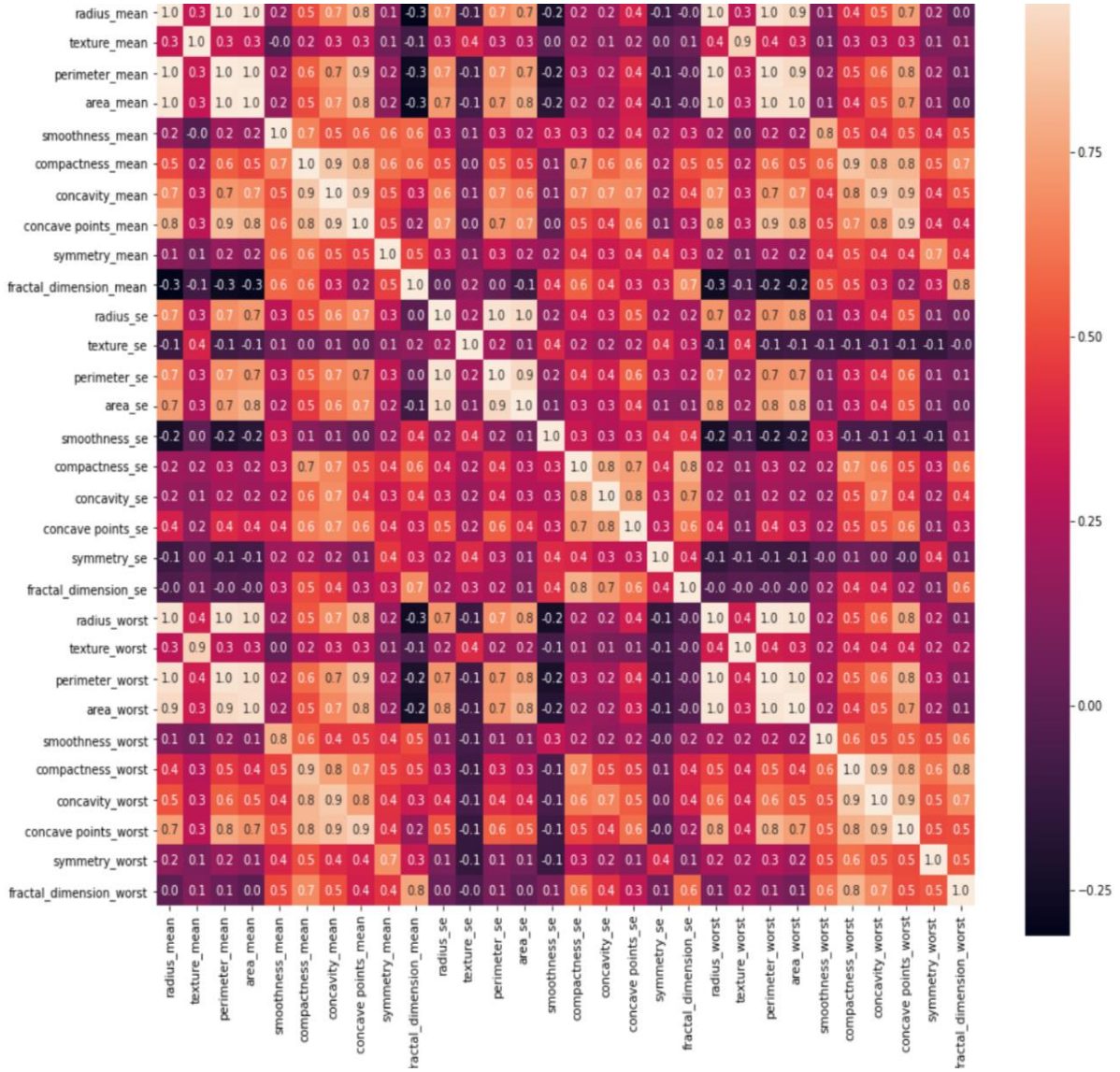
Showing the next 10 features in a plot:



## Computation and Visualizing

Showing the correlation between all features:



### Select k-best algorithm:[1]

I should determine which columns have a high correlation with the Diagnosis column (malignant or benign) and which one is independent. I will use the K-Best algorithm to find dependent and independent features.

For using the K-Best algorithm, numeric features are considered as " x " and "Diagnosis" column is called "y", and it is a target variable.

This algorithm computes score and p_value for all numeric features. Columns with a high score and p_value less than 0.05 will be chosen.

- The null hypothesis is about features contain no information about the target variable.
- The alternative hypothesis is about features that have a relationship with the Diagnosis column.

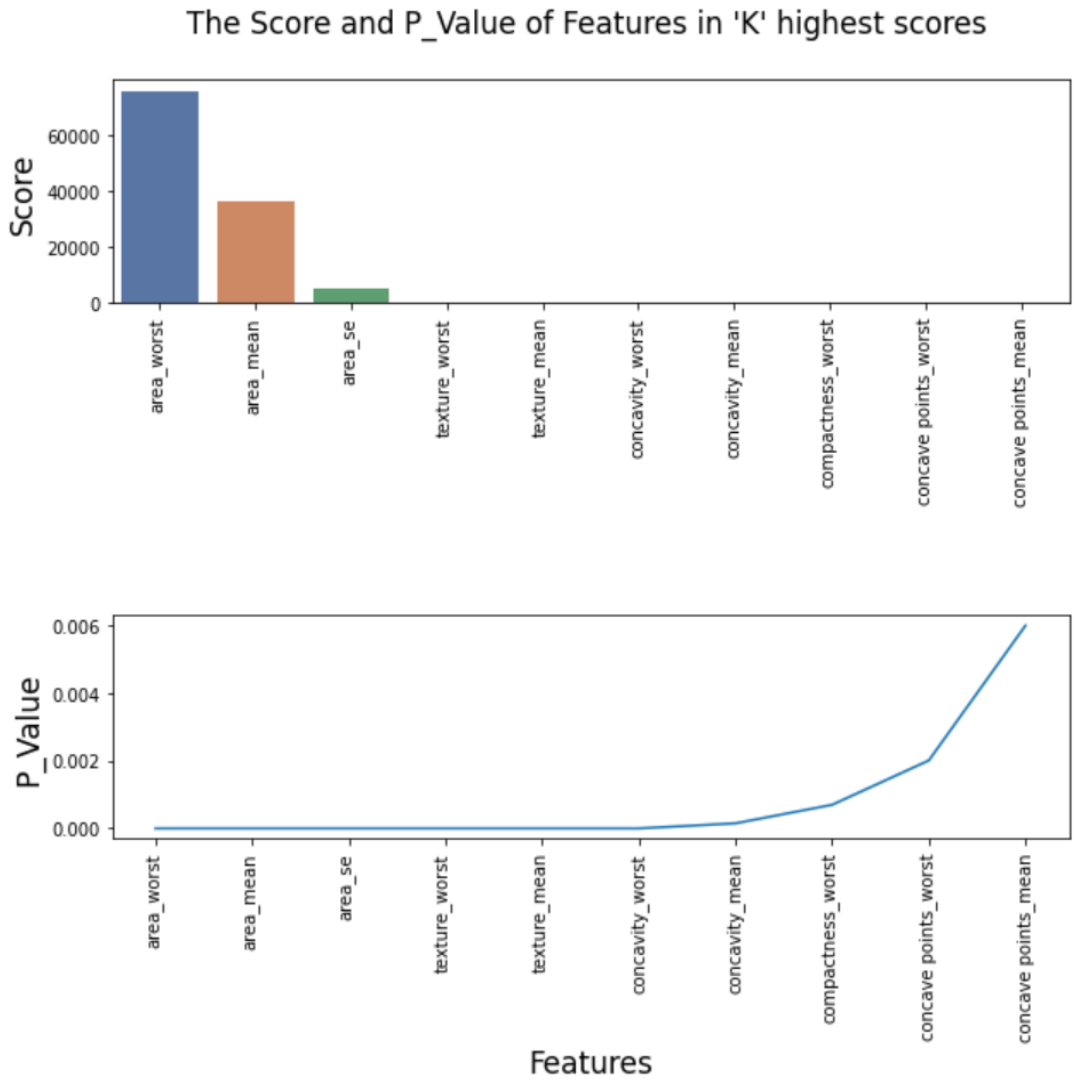
Therefore, with 95 per cent insurance, we can reject the Null hypothesis for features with p_value < 0.05.

Output: Bellow features can impact on the prediction of the type of tumor among people with breast cancer. Because the score of these features is more than zero and P-value is less than 0.05:

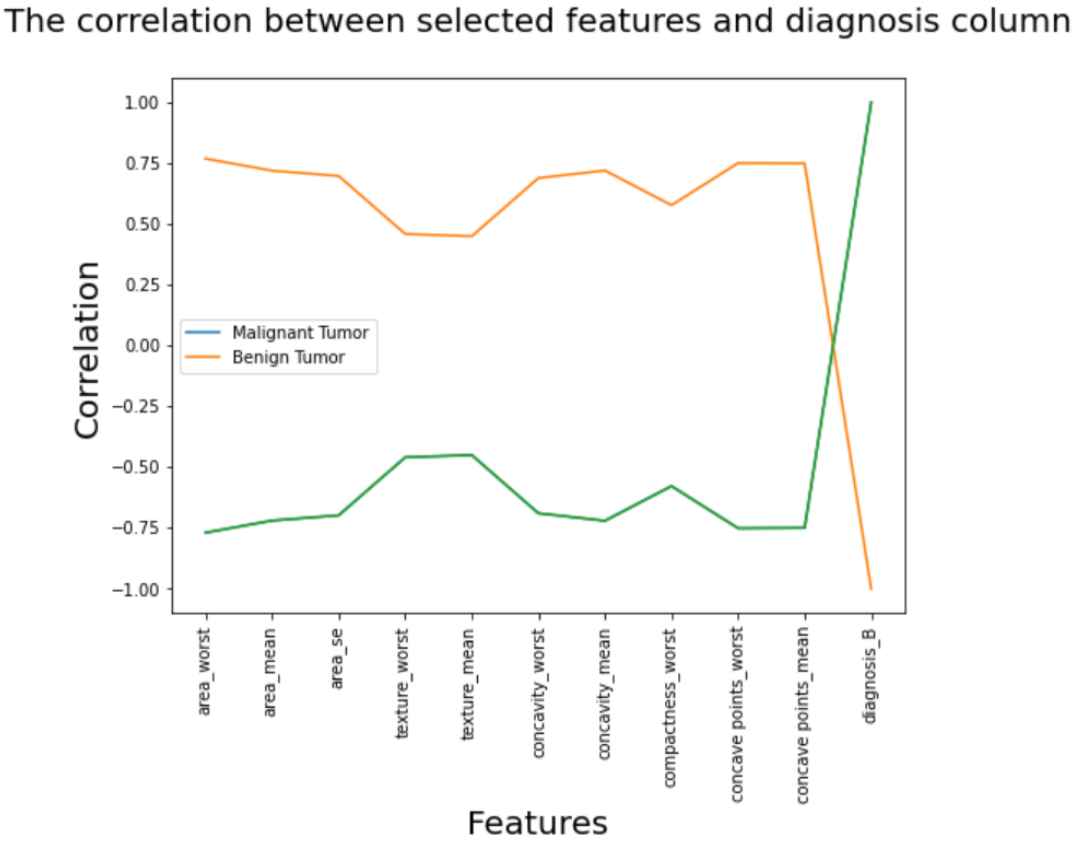
| | features | scores | p_value |
|---|---|---|---|
| 17 | area_worst | 76151 | 0 |
| 1 | area_mean | 36313 | 0 |
| 9 | area_se | 5308 | 0 |
| 16 | texture_worst | 123 | 8.68309e-29 |
| 0 | texture_mean | 68 | 1.48896e-16 |
| 20 | concavity_worst | 28 | 9.84006e-08 |
| 4 | concavity_mean | 14 | 0.000149965 |
| 19 | compactness_worst | 11 | 0.000696395 |
| 21 | concave points_worst | 9 | 0.00200227 |
| 5 | concave points_mean | 7 | 0.0059768 |

[1] reference: https://scikit-learn.org/stable/modules/feature_selection.html

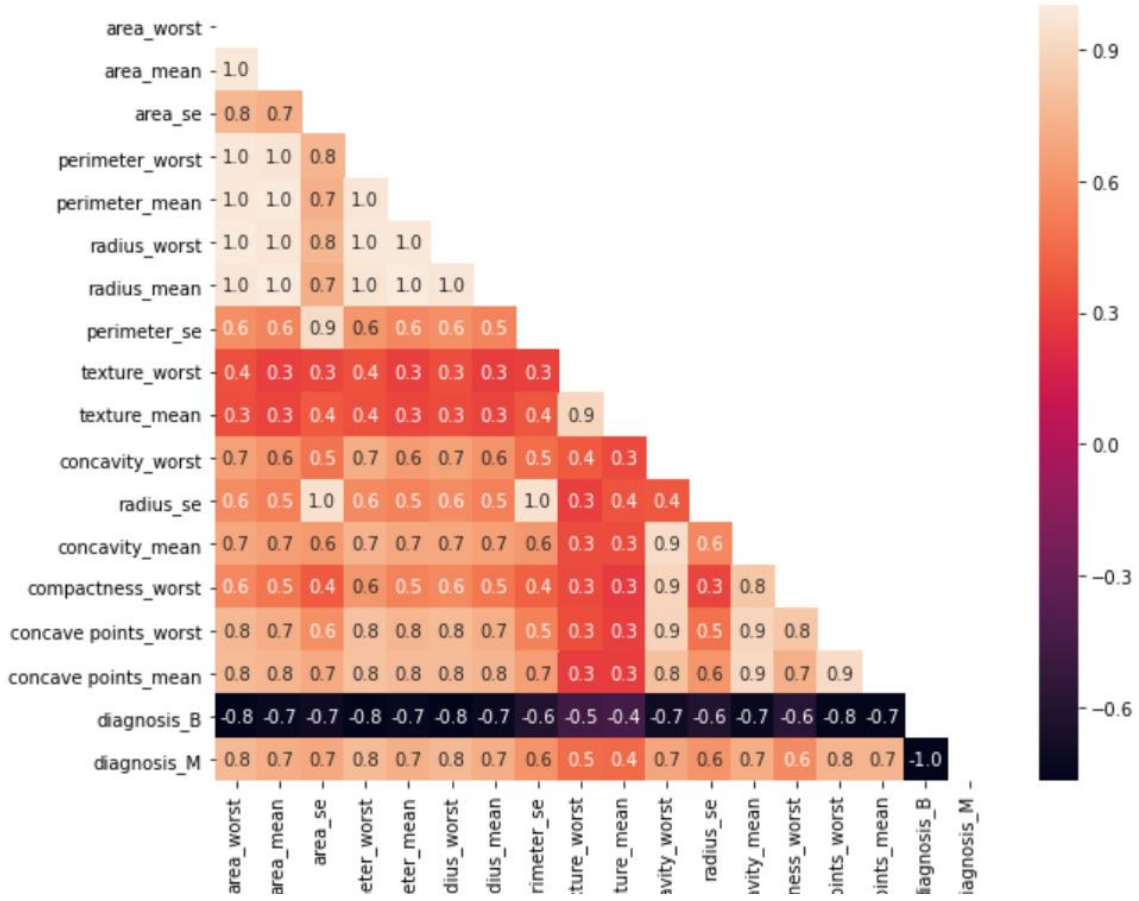
Visualizing the scores and p-value of features:



Extracting a part of correlation matrix for showing the correlation between selected features and Diagnosis column in a line plot:



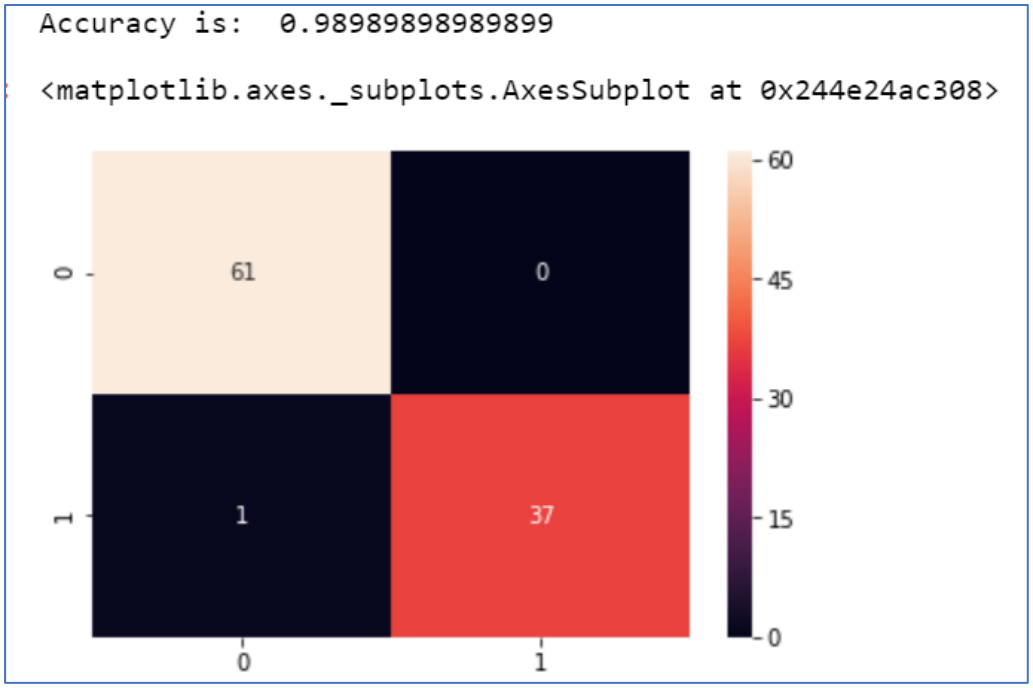
Drawing a heatmap among selected features:



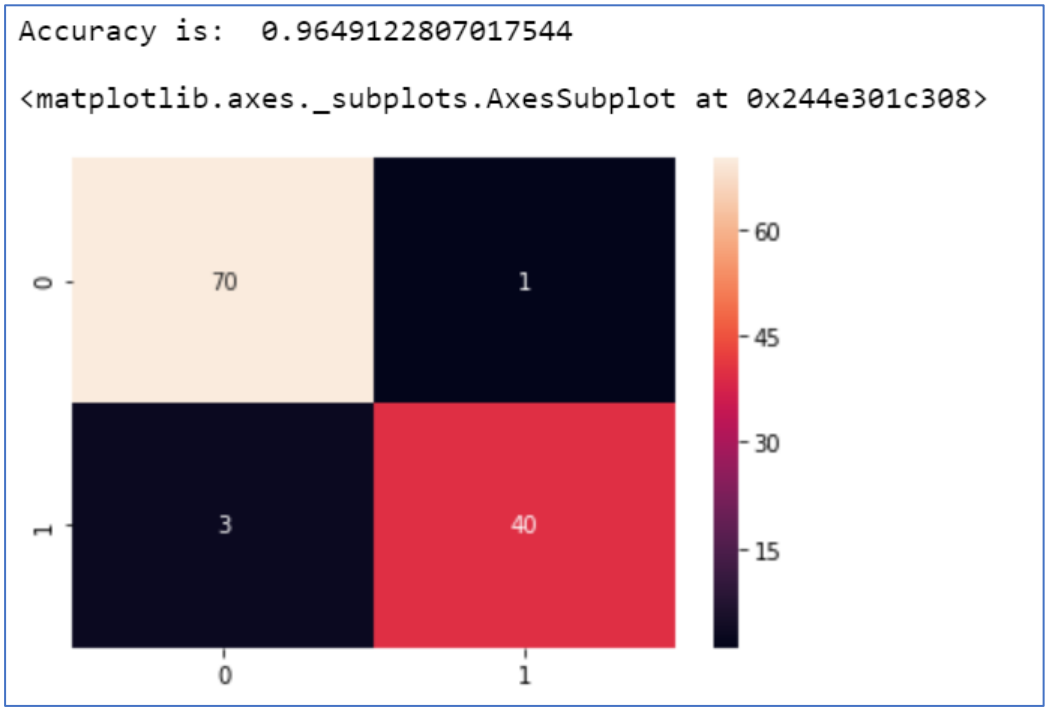
## Random Forest Classifier Algorithm:

Using random forest classifier to predict the target column (Diagnosis column). The accuracy of this algorithm on selected features from Select K-best algorithm is about 99 percent.

According to the below heatmap, the algorithm has only one error.




Using random forest classifier for all features without preparing the dataset: The prediction has 3 error, and the accuracy is about 96.5 percent.



## 8. Conclusion

Regarding the importance of an accurate prediction in the realm of medical, I used some algorithms to increase the accuracy of predicting breast cancer.

The dataset has 31 columns about the symptom of breast cancer among approximately 550 patients. In the Diagnosis column, the type of cancer is available, and I desired to find which columns impact on the type of tumor in the breast cancer.

First, I used the Select K-best algorithm. This feature is a technique where we choose those variables in our data that contribute most to the target variable (Diagnosis column). In other words, I wanted to select the best predictors for the target variable.

Second, I used the Random Forest algorithm to predict the type of cancer, and I computed the accuracy of the model to make sure that selected features work correctly.

Finally, I showed that the accuracy of the algorithm without preparing the dataset is lower than when I used selected features and prepared data before training the model:

- ✓ The accuracy of the random forest algorithm after preparation among selected features is: 99 per cent
- ✓ The accuracy of the random forest algorithm before preparing the dataset is: 96.5 per cent