

# INTRODUCTION TO DATA SCIENCE

## **BREAST CANCER** **CLASSIFICATION**

### **PROJECT REPORT**

**Submitted by : (GROUP NUMBER 11)**

**Manan Bihani (21UCS122)**

**Manan Gupta (21UCS123)**

**Manan Jain (21UCS124)**

**Manas Vashisht (21UCS125)**

#### **Course Instructors**

Dr. Alope Datta

Dr. Subrat Dash

Dr. Lal Upendra

Pratap Singh

Department of Computer Science Engineering  
The LNM Institute of Information Technology

# TABLE OF CONTENTS

## 1. Problem statement

### 1.1 Introduction

### 1.2 Aim of the Project

### 1.3 Dataset

### 1.4 Performance Metrics

### 1.5 Imported libraries for the project

## 2. Data Preprocessing and Preliminary Analysis

### 2.1 Data Preprocessing

Handling Missing Data

Handling Duplicates

Data Transformation

Data Reduction

### 2.2 Preliminary Analysis

Descriptive Statistics

Data Profiling

Data Visualization

Correlation Analysis

## 3. ML Classification Algorithms

3.1 Gather and prepare Data

3.2 Data splitting

3.3 Feature Engineering

3.4 Model Classification

**Logistic Regression**

**K Nearest Neighbour**

**Random Forest**

**Support Vector Machine**

4. Inferences

5. Conclusion

6. References

# 1.0 Problem Statement

In this project we aim to analyze and study breast cancer dataset and use various Machine Algorithms to check whether the patient is diagnosed with benign or malignant depending on various attributes like radius, texture, perimeter of the cell-nucleus etc.

Following classifiers have been chosen for comparison in this project:

- Logistic reasoning
- K-Nearest Neighbour
- Random forest
- Support Vector Machine

## 1.1 Introduction

A neoplasm is an abnormal tissue mass that grows faster and differently than normal tissue. Cancer can start anywhere in the body, spreading through the blood or lymph system. Unlike harmless tumors, cancerous ones invade nearby tissues. Breast cancer is common and causes many deaths yearly. Mammograms help detect it early. A biopsy test like Fine Needle Aspirates checks suspicious masses. This project prepares and examines a breast cancer dataset upfront. It cleans and looks at the data to understand it better before deeper analysis and making models.

## 1.2 Aim of the project

The goal of this report is to use machine learning to predict if breast cancer cells are benign or malignant. Data will be transformed and simplified to uncover patterns, aiding a more thorough analysis. We aim for a model that accurately identifies cancer types from cell images. The focus is on achieving high accuracy and sensitivity, minimizing false negatives, to ensure reliable classification of benign and malignant cells.

## 1.3 Dataset

This report explores the Breast Cancer Wisconsin (Diagnostic) DataSet, curated by Dr. William H. Wolberg from the University of Wisconsin Hospital. The dataset, sourced from ics, consists of biopsy results from 569 patients at Wisconsin Hospital, collected in 1993.

The data has been taken from UCI Machine Learning Repository

<https://archive.ics.uci.edu/ml/datasets.php>.

We have two dataset one consisting of features, namely X and other with the corresponding classes, namely y . Thus , we combined both of them to form a new dataset "data" to analyze the data better. Here is the code provided below.

```
[8] data=pd.merge(X,y,on='index')
```

The features in the breast cancer dataset provide measurements and characteristics of cell nuclei from breast cancer biopsies. These features are used in the context of diagnosing whether a tumor is malignant or benign. Here's a more specific explanation of each variable in the context of breast cancer diagnosis:

1. radius1: The average distance from the center to points on the perimeter of the cell nucleus. Larger values may be associated with larger tumor sizes.
2. texture1: The standard deviation of gray-scale values in the cell nucleus. It reflects the variation in color intensity, potentially indicating irregularities in cell structure.
3. perimeter1: The perimeter of the cell nucleus. This can be an indicator of the overall size and shape of the cell.
4. area1: The area of the cell nucleus. Similar to perimeter, it provides information about the size of the cell.
5. smoothness1: The local variation in radius lengths. Higher values may indicate a smoother cell surface.

6. compactness1: A measure of how compact the cell nucleus is. It is calculated using the formula  $(\text{perimeter}^2 / \text{area} - 1.0)$ .

7. concavity1: The severity of concave portions of the cell nucleus contour. Higher values may indicate more irregularities.

8. concave\_points1: The number of concave portions in the cell nucleus contour. It provides information about the shape of the cell.

9. symmetry1: A measure of symmetry in the cell nucleus. Deviations from symmetry might be indicative of abnormal cell growth.

10. fractal\_dimension1: A measure related to the complexity of the cell nucleus contour. It's a way of quantifying irregularities and complexities in the cell shape.

The same set of features is repeated with the suffixes 2(standard error) and 3(worst - mean of the three largest values), providing additional information and context about the variability of these features.

For example:

- radius2: Standard error of the mean of distances from the center to points on the perimeter.
- radius3: Worst or largest mean of distances from the center to points on the perimeter.

## 1.4 Performance Metrics

- Precision

Precision is the number of correct positive results divided by the number of all positive results returned by the classifier.

- Recall

Recall is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

- F1-score

The F1-score is a measure of a model's accuracy on a dataset. It considers both the precision  $p$  and the recall  $r$  of the test to compute the score. It is the harmonic average of the precision and recall.

- Confusion Matrix

Confusion Matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.

- Accuracy

Accuracy is the number of correctly predicted data points out of all the data points. More formally, it is the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives.

## 1.5 IMPORTED LIBRARIES FOR THE PROJECT

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

*The Imported Libraries are used for the following purposes:-*

1. Data Handling:
  - numpy for numerical computing.
  - pandas for data manipulation.
2. Visualization:
  - matplotlib.pyplot for static plotting.
  - seaborn for statistical visualizations.
  - plotly.express for interactive plotting.
3. Machine Learning:
  - scikit-learn for machine learning tasks.
    - train\_test\_split for data splitting.
    - StandardScaler for feature scaling.
    - PCA for dimensionality reduction.
    - LogisticRegression for classification.
    - RandomForestClassifier for ensemble classification.
    - KNeighborsClassifier for k-nearest neighbors classification.
    - SVC for support vector classification.
4. Evaluation:
  - Metrics from scikit-learn for model evaluation.
    - confusion\_matrix for confusion matrices.
    - accuracy\_score for accuracy.
    - classification\_report for detailed classification metrics.



## 2.Data Preprocessing and Preliminary Analysis

By observing our dataset, we found that it contains 569 observations with 32 variables.

```
[128] data.shape  
  
(569, 32)
```

### 2.1 Data preprocessing

- Handling Missing Data

We have checked whether our dataset contains any missing value and we found out that there is no missing value in the dataset. The code is provided below for the same.

```
X.isnull().sum()  
  
radius1      0  
texture1     0  
perimeter1   0  
area1        0  
smoothness1  0  
compactness1 0  
concavity1   0  
concave_points1 0  
symmetry1    0  
fractal_dimension1 0  
radius2      0  
texture2     0  
perimeter2   0  
area2        0  
smoothness2  0  
compactness2 0  
concavity2   0  
concave_points2 0  
symmetry2    0  
fractal_dimension2 0  
radius3      0  
texture3     0  
perimeter3   0  
area3        0  
smoothness3  0  
compactness3 0  
concavity3   0  
concave_points3 0  
symmetry3    0  
fractal_dimension3 0  
dtype: int64
```

## ● Handling Duplicates

We have checked for the duplicate records and found out that there is no duplicate record present in the dataset. By using `data.duplicated()` and followed by a for loop that counts the number of duplicate rows, which comes out zero. The code is provided below.

```
[142] duplicateRows=data.duplicated()
      count=0
      for duplicate in duplicateRows:
          if duplicate == True:
              count=count+1

      print('Number of duplicate Rows', count)

Number of duplicate Rows 0
```

## ● Data Transformation

Since, dealing in the Numerical number is easy instead of dealing in the characters. Thus, we have converted Malignant ("M") into "1" and Benign ("B") into "0".

```
[189] data['Diagnosis']=pd.Categorical(data.Diagnosis).codes
      data['Diagnosis'].unique()
      data.head()
```

	radius1	texture1	perimeter1	area1	smoothness1	compactness1	concavity1	concave_points1	symmetry1	fractal_dimension1	...	texture3	perimeter3	area3	smoothness3	c
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374	0

5 rows x 31 columns

Using standard scalar on the dataset, we standardize features by removing the mean and scaling to unit variance.

```
[30] from sklearn.preprocessing import StandardScaler
      scaler = StandardScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
```

## • Data Reduction

Applied data dimensionality reduction by dropping Index since it is not significant use in our analysis.

```
[15] data.drop('index',axis=1,inplace=True)
      X.drop('index',axis=1,inplace=True)
      y.drop('index',axis=1,inplace=True)
```

## 2.2 preliminary analysis

### • Descriptive Statistics

Calculated basic statistical measures like mean, median, standard deviation, minimum, maximum etc., to summarize the main characteristics of the dataset.

data.describe().T

	count	mean	std	min	25%	50%	75%	max
radius1	569.0	14.127292	3.524049	6.981000	11.700000	13.370000	15.780000	28.11000
texture1	569.0	19.289649	4.301036	9.710000	16.170000	18.840000	21.800000	39.28000
perimeter1	569.0	91.969033	24.298981	43.790000	75.170000	86.240000	104.100000	188.50000
area1	569.0	654.889104	351.914129	143.500000	420.300000	551.100000	782.700000	2501.00000
smoothness1	569.0	0.096360	0.014064	0.052630	0.086370	0.095870	0.105300	0.16340
compactness1	569.0	0.104341	0.052813	0.019380	0.064920	0.092630	0.130400	0.34540
concavity1	569.0	0.088799	0.079720	0.000000	0.029560	0.061540	0.130700	0.42680
concave_points1	569.0	0.048919	0.038803	0.000000	0.020310	0.033500	0.074000	0.20120
symmetry1	569.0	0.181162	0.027414	0.106000	0.161900	0.179200	0.195700	0.30400
fractal_dimension1	569.0	0.062798	0.007060	0.049960	0.057700	0.061540	0.066120	0.09744
radius2	569.0	0.405172	0.277313	0.111500	0.232400	0.324200	0.478900	2.87300
texture2	569.0	1.216853	0.551648	0.360200	0.833900	1.108000	1.474000	4.88500
perimeter2	569.0	2.866059	2.021855	0.757000	1.606000	2.287000	3.357000	21.98000
area2	569.0	40.337079	45.491006	6.802000	17.850000	24.530000	45.190000	542.20000
smoothness2	569.0	0.007041	0.003003	0.001713	0.005169	0.006380	0.008146	0.03113
compactness2	569.0	0.025478	0.017908	0.002252	0.013080	0.020450	0.032450	0.13540
concavity2	569.0	0.031894	0.030186	0.000000	0.015090	0.025890	0.042050	0.39600

concave_points2	569.0	0.011796	0.006170	0.000000	0.007638	0.010930	0.014710	0.05279
symmetry2	569.0	0.020542	0.008266	0.007882	0.015160	0.018730	0.023480	0.07895
fractal_dimension2	569.0	0.003795	0.002646	0.000895	0.002248	0.003187	0.004558	0.02984
radius3	569.0	16.269190	4.833242	7.930000	13.010000	14.970000	18.790000	36.04000
texture3	569.0	25.677223	6.146258	12.020000	21.080000	25.410000	29.720000	49.54000
perimeter3	569.0	107.261213	33.602542	50.410000	84.110000	97.660000	125.400000	251.20000
area3	569.0	880.583128	569.356993	185.200000	515.300000	686.500000	1084.000000	4254.00000
smoothness3	569.0	0.132369	0.022832	0.071170	0.116600	0.131300	0.146000	0.22260
compactness3	569.0	0.254265	0.157336	0.027290	0.147200	0.211900	0.339100	1.05800
concavity3	569.0	0.272188	0.208624	0.000000	0.114500	0.226700	0.382900	1.25200
concave_points3	569.0	0.114606	0.065732	0.000000	0.064930	0.099930	0.161400	0.29100
symmetry3	569.0	0.290076	0.061867	0.156500	0.250400	0.282200	0.317900	0.66380
fractal_dimension3	569.0	0.083946	0.018061	0.055040	0.071460	0.080040	0.092080	0.20750
Diagnosis	569.0	0.372583	0.483918	0.000000	0.000000	0.000000	1.000000	1.00000

## • Data Profiling

Data types - we have concluded the following data type are used for the variable used in the dataset.

```
data.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 569 entries, 0 to 568
Data columns (total 31 columns):
#   Column                Non-Null Count  Dtype
---  -
0   radius1                569 non-null    float64
1   texture1               569 non-null    float64
2   perimeter1             569 non-null    float64
3   area1                  569 non-null    float64
4   smoothness1            569 non-null    float64
5   compactness1           569 non-null    float64
6   concavity1             569 non-null    float64
7   concave_points1        569 non-null    float64
8   symmetry1              569 non-null    float64
9   fractal_dimension1     569 non-null    float64
10  radius2                569 non-null    float64
11  texture2               569 non-null    float64
12  perimeter2             569 non-null    float64
13  area2                  569 non-null    float64
14  smoothness2            569 non-null    float64
15  compactness2           569 non-null    float64
16  concavity2             569 non-null    float64
17  concave_points2        569 non-null    float64
18  symmetry2              569 non-null    float64
19  fractal_dimension2     569 non-null    float64
20  radius3                569 non-null    float64
21  texture3               569 non-null    float64
22  perimeter3             569 non-null    float64
23  area3                  569 non-null    float64
24  smoothness3            569 non-null    float64
25  compactness3           569 non-null    float64
26  concavity3             569 non-null    float64
27  concave_points3        569 non-null    float64
28  symmetry3              569 non-null    float64
29  fractal_dimension3     569 non-null    float64
30  Diagnosis              569 non-null    int8
dtypes: float64(30), int8(1)
memory usage: 138.4 KB
```

## Range

The term "range" in the context of data refers to the difference between the maximum and minimum values in a dataset. It is a measure of the spread or variability of the data. Mathematically, the range (R) is calculated as:

$$R = \text{Maximum Value} - \text{Minimum Value}$$

```

▶ describedData['range']

↳ radius1          21.129000
   texture1        29.570000
   perimeter1      144.710000
   area1           2357.500000
   smoothness1     0.110770
   compactness1    0.326020
   concavity1      0.426800
   concave_points1 0.201200
   symmetry1       0.198000
   fractal_dimension1 0.047480
   radius2          2.761500
   texture2         4.524800
   perimeter2       21.223000
   area2           535.398000
   smoothness2     0.029417
   compactness2    0.133148
   concavity2      0.396000
   concave_points2 0.052790
   symmetry2       0.071068
   fractal_dimension2 0.028945
   radius3          28.110000
   texture3        37.520000
   perimeter3      200.790000
   area3           4068.800000
   smoothness3     0.151430
   compactness3    1.030710
   concavity3      1.252000
   concave_points3 0.291000
   symmetry3       0.507300
   fractal_dimension3 0.152460
Name: range, dtype: float64

```

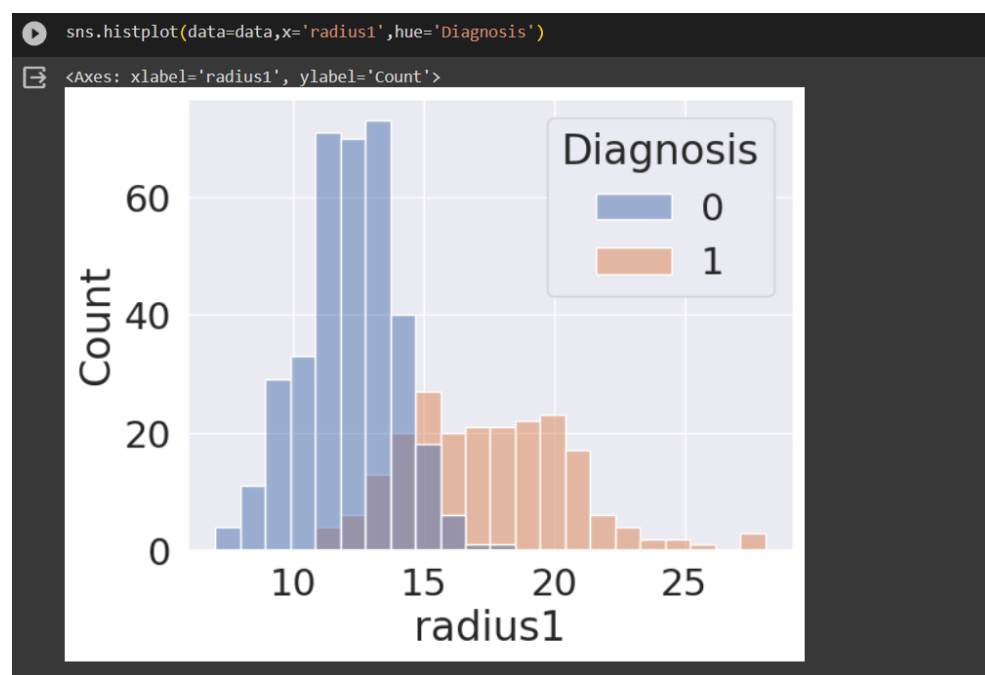
## ● Data Visualization

Created different visual representations such as :-

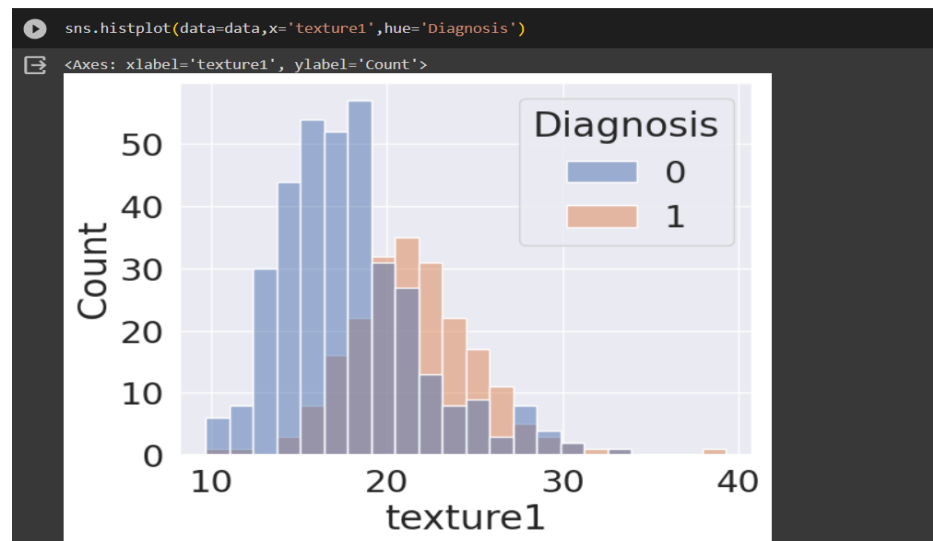
### 1. Histogram analysis

Histogram analysis is a useful technique in the preliminary analysis phase to understand the distribution of numerical variables in a dataset.

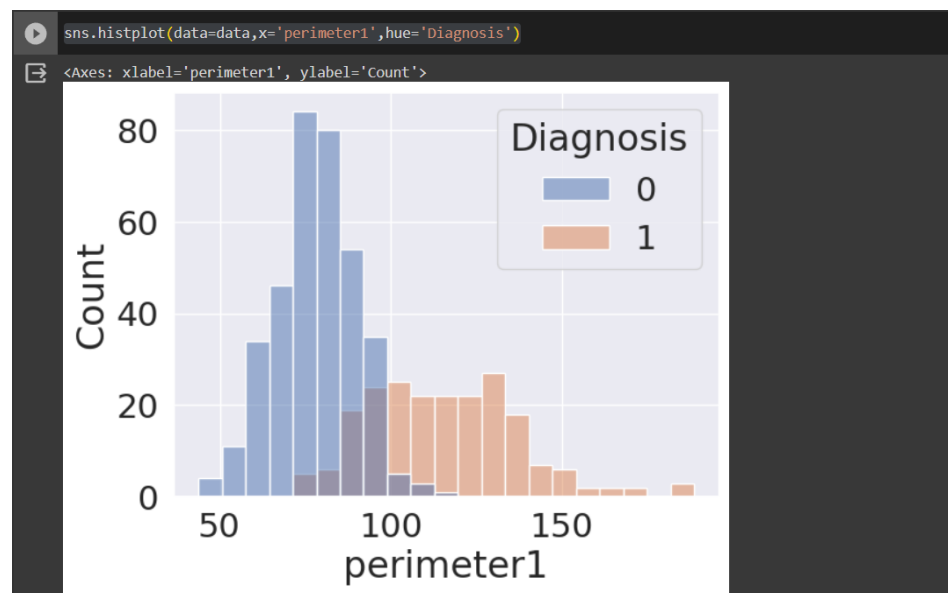
We have represented a histogram for each of the attributes.



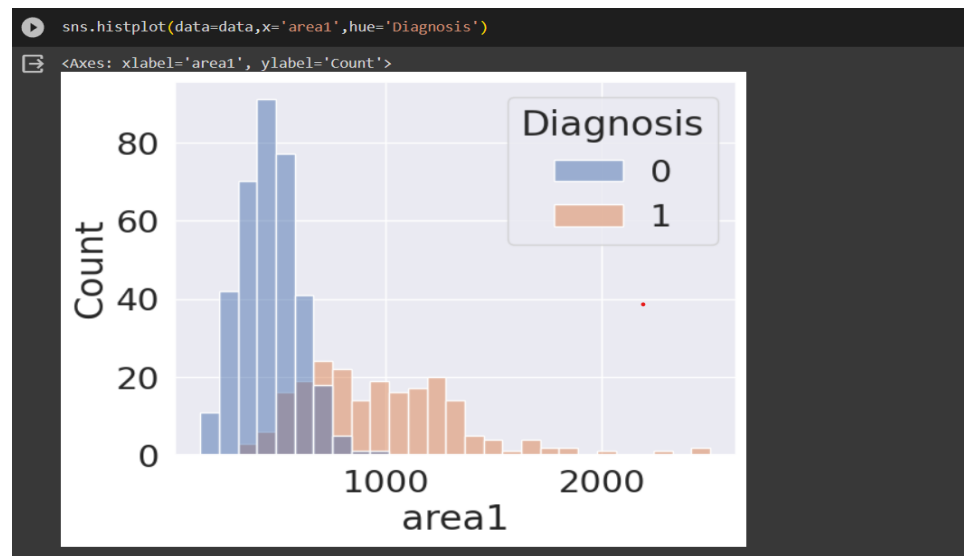
**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis are having more radius1 (mean\_radius) compared to the patient with Benign diagnosis.



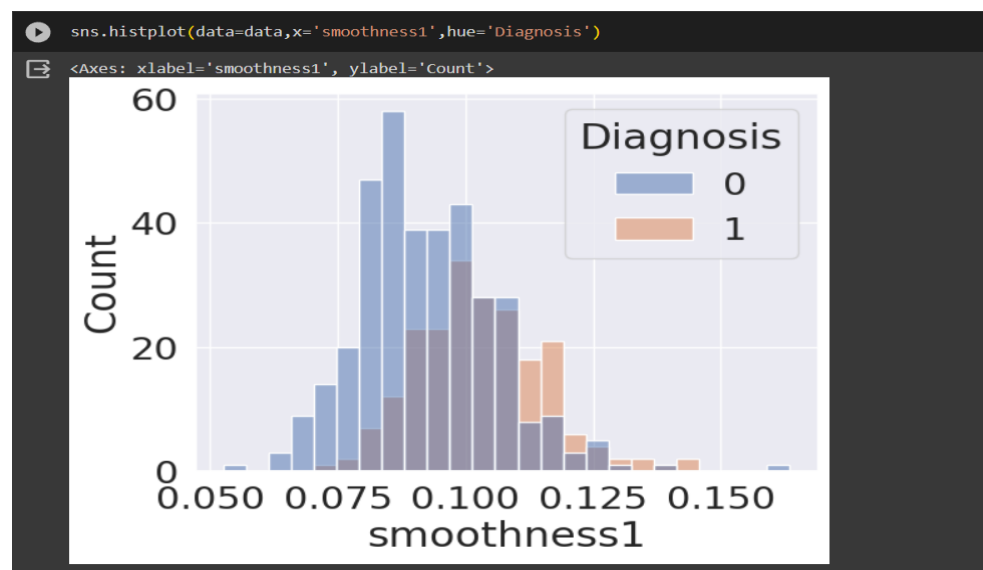
**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis are having slightly more texture1(mean\_texture) compared to the patient with Benign diagnosis.



**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis are having slightly more perimeter1(mean\_perimeter) compared to the patient with Benign diagnosis.

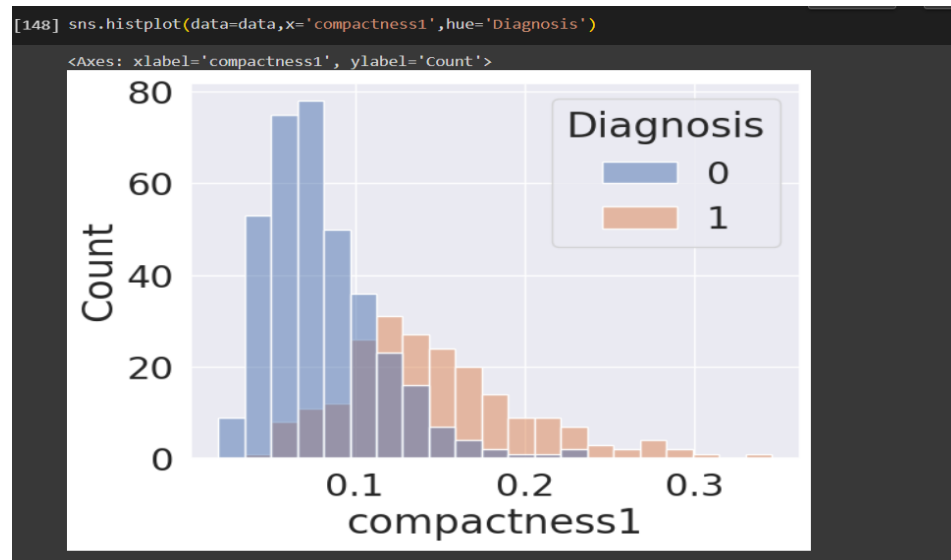


**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis are having more area1 (mean\_area) compared to the patient with Benign diagnosis.

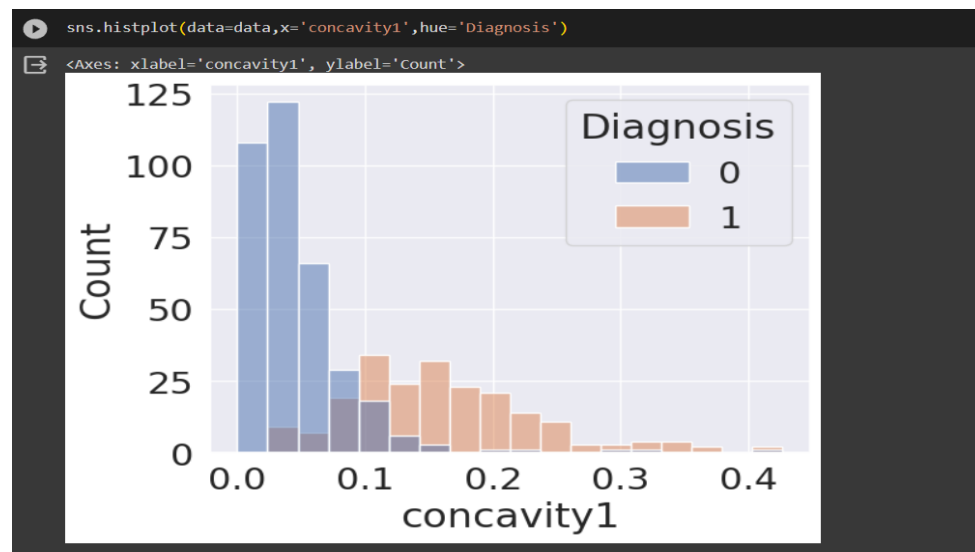


**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis are having slightly more but comparable smoothness1 (mean\_smoothness) compared to the patient with Benign diagnosis.

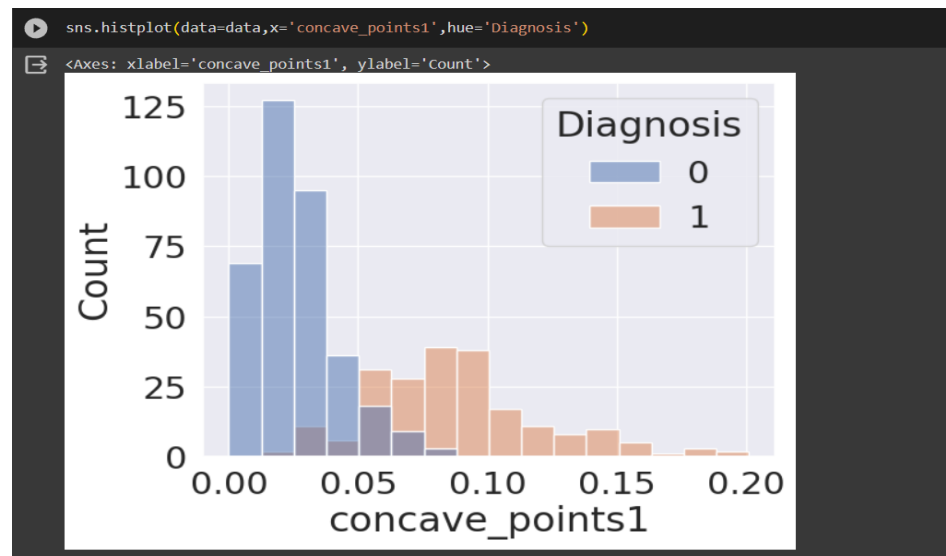




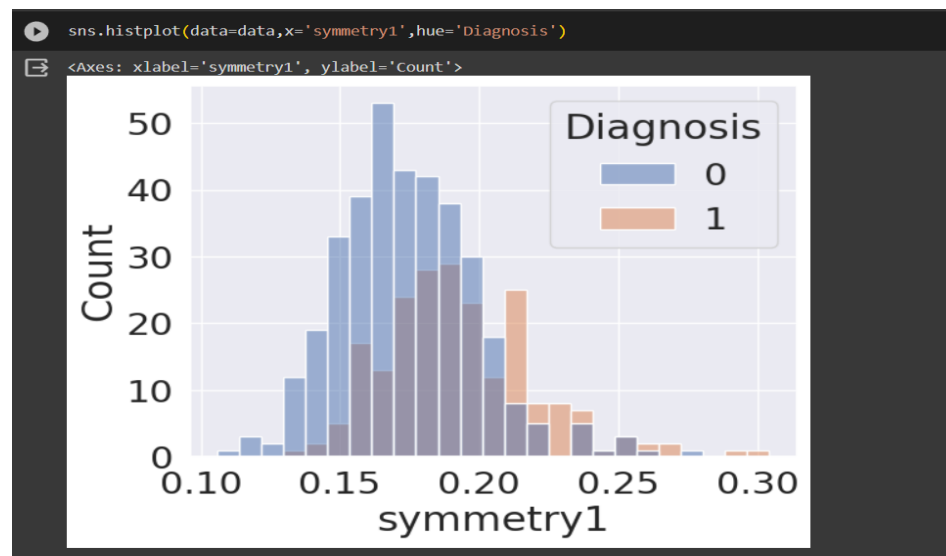
**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis are having slightly more compactness1 (mean\_compactness) compared to the patient with Benign diagnosis.



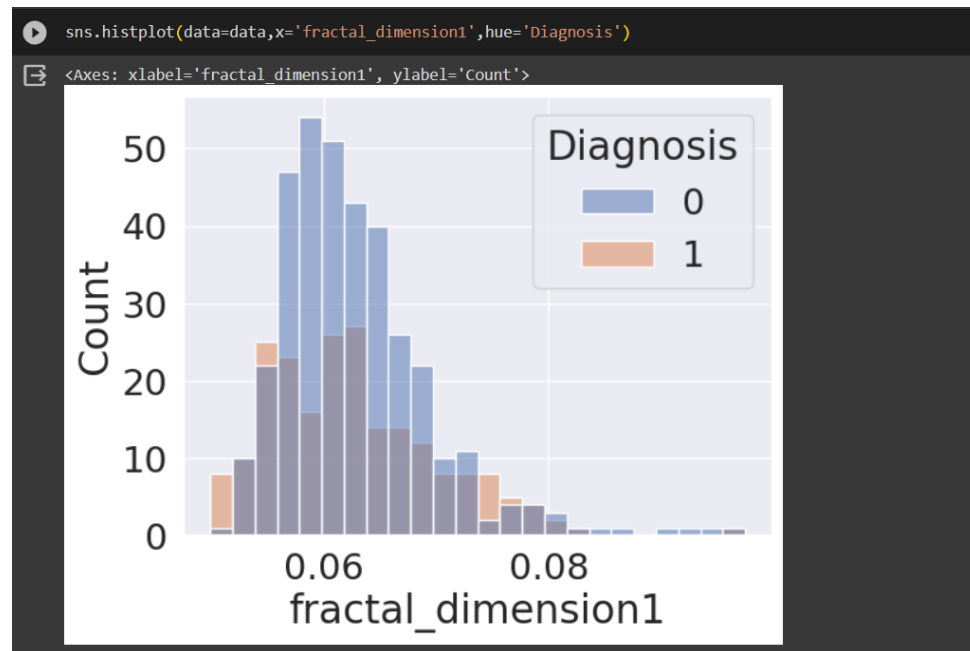
**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis are having more concavity1 (mean\_concavity) compared to the patient with Benign diagnosis.



**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis are having more concave\_points1 (mean\_concave\_points) compared to the patient with Benign diagnosis.



**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis are having comparable symmetry1 (mean\_symmetry1) to the patient with Benign diagnosis.

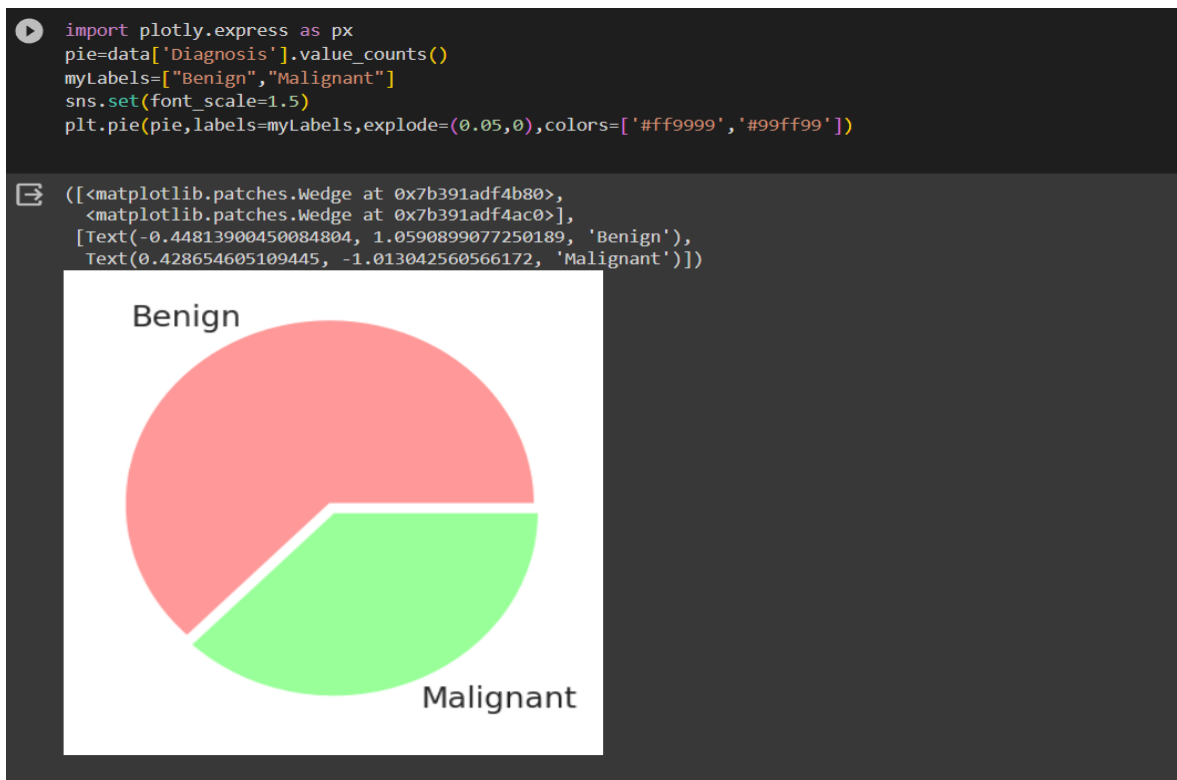


**Inference:** From the above Histogram plot for both diagnosis. We can infer that the patients with Malignant diagnosis have similar fractal\_dimension1 (mean\_fractal\_dimension1) to the patient with Benign diagnosis.

## 2. Pie Chart

The shape resembles a pie divided into slices. slice angle follows proportional representation. most suited for representing a limited number of categories.

We have only two categories for the pie chart representation Malignant and Benign.



**Inference** :From the above Pie chart, we can easily analyze that Benign Diagnosis is more in number as compared to Malignant Diagnosis.

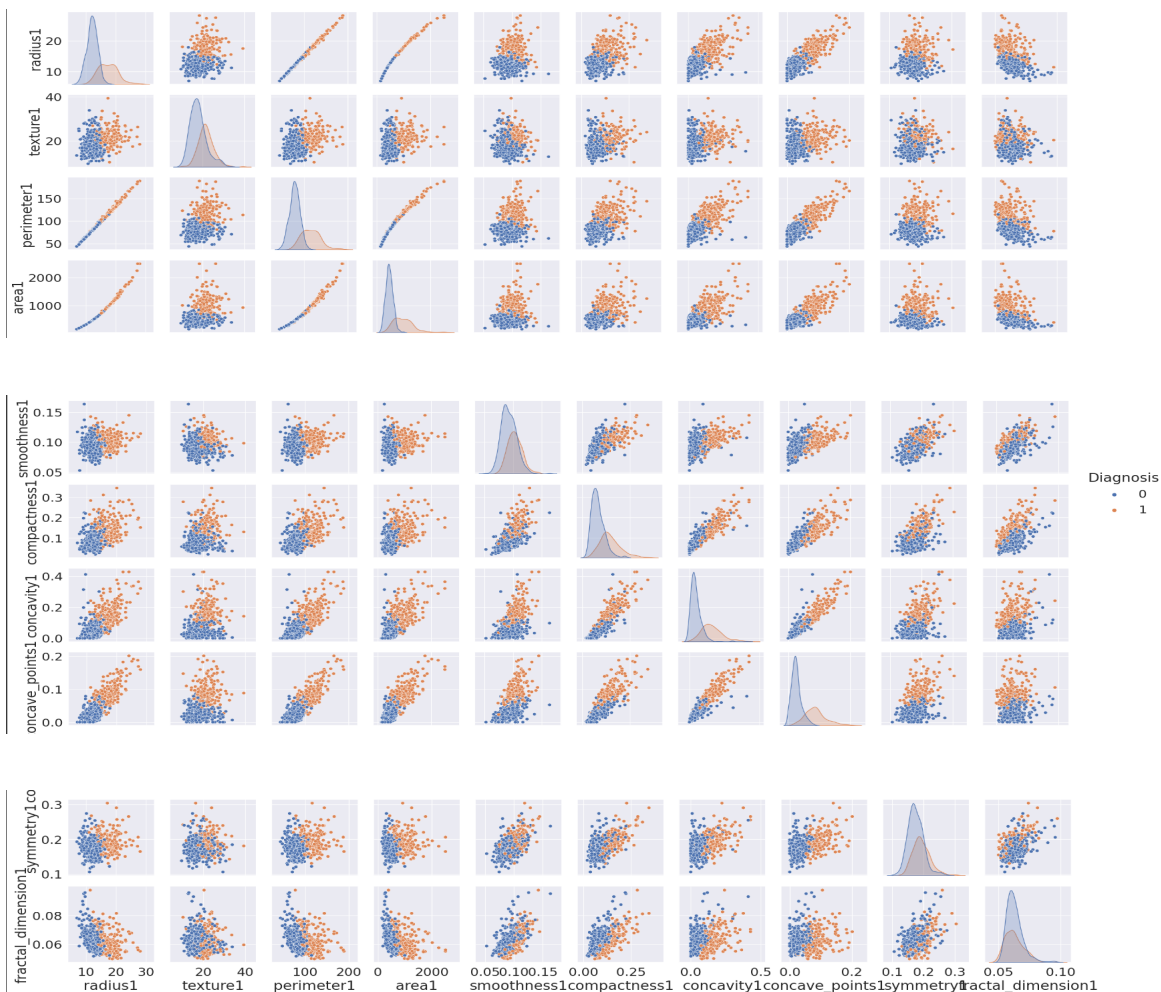
## 3 Scatter Plot

A scatter plot shows points on a graph to reveal relationships between two variables. Useful for exploring connections between features in the breast cancer dataset.

**Inference:** In the scatter plot representation ,we have only considered features which represent mean value like radius1,perimeter1 etc .As we can see from scatter plot there are some features which are highly correlated with each other likewise scatter plot

between perimeter and radius1, radius1 and area1, area and perimeter1 and many more which are given in the correlated analysis ( which are discussed below).

```
[23] cols=['Diagnosis',
        'radius1',
        'texture1',
        'perimeter1',
        'area1',
        'smoothness1',
        'compactness1',
        'concavity1',
        'concave_points1',
        'symmetry1',
        'fractal_dimension1']
sns.pairplot(data=data[cols],hue='Diagnosis')
```

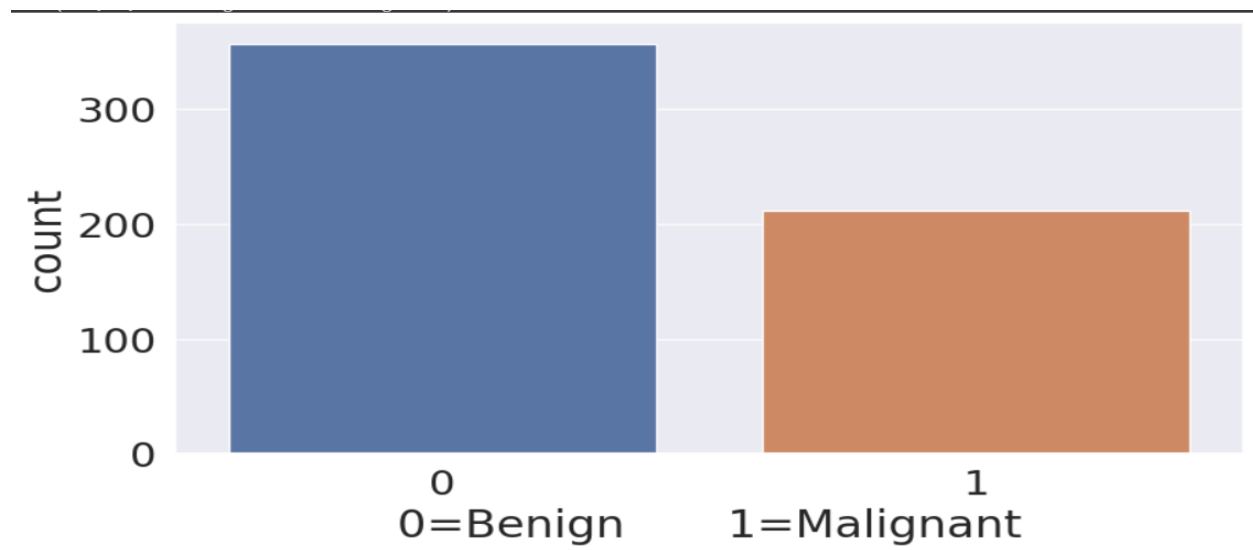


## 4. Count Plot

A count plot shows the frequency of different categories in a categorical variable. Useful for understanding the distribution of, for example, malignant and benign diagnoses in the breast cancer dataset.

```
[ ] data['Diagnosis'].value_counts()

0    357
1    212
Name: Diagnosis, dtype: int64
```

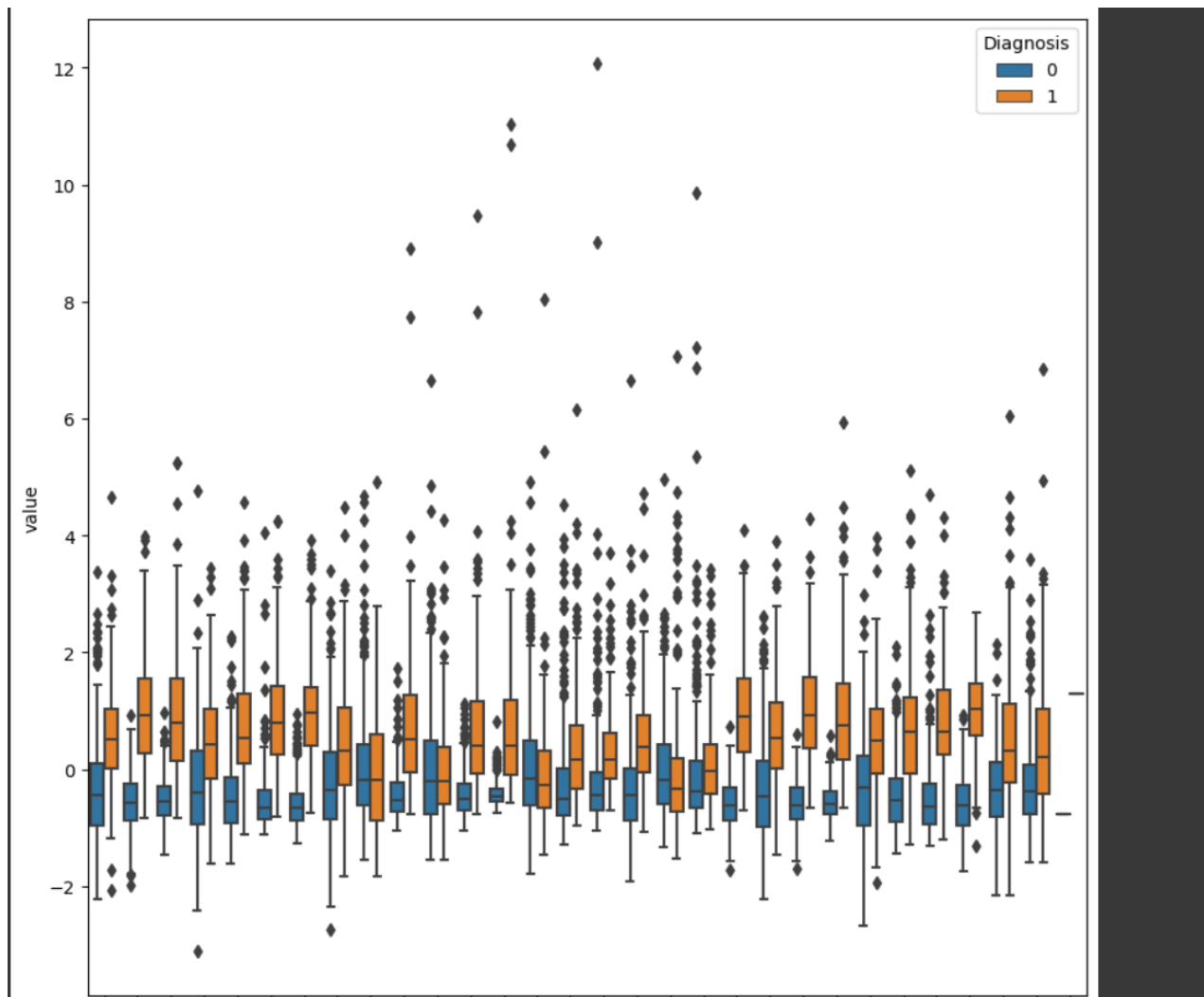


**Inference :** From the above count plot, we can easily analyze that Benign Diagnosis is more in number as compared to Malignant Diagnosis.

## 5. Box Plot

A Box Plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum.

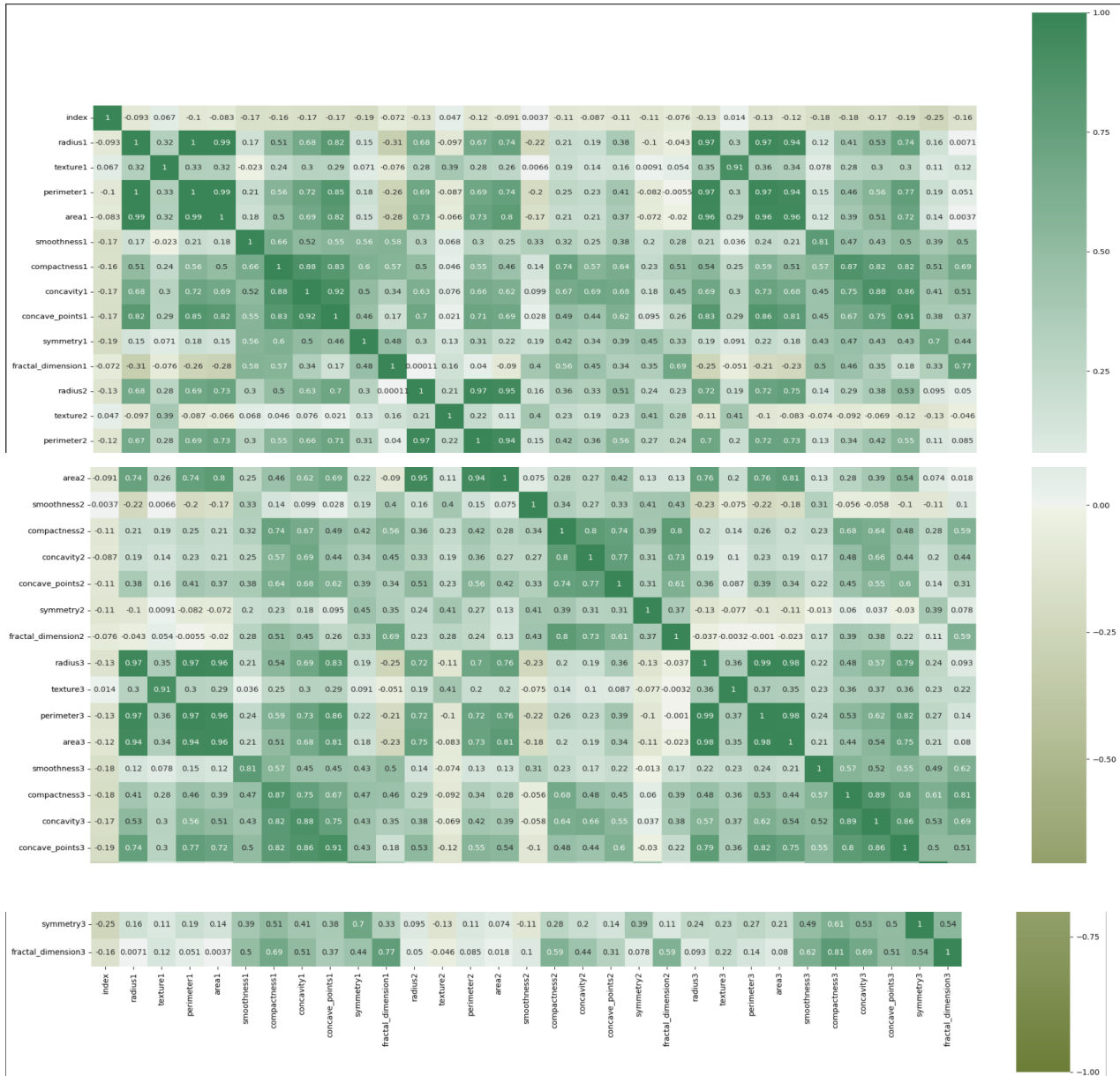
**Inference :** From given box plots we can analyze that some attributes have very similar median for both benign and malignant. Thus these features might not be a good choice for differentiation between two classes.



## ● Correlation Analysis

Correlation analysis assesses how strongly and in what direction two numerical variables are related. The correlation coefficient ranges from -1 to 1, with 1 indicating a strong positive correlation, -1 a strong negative correlation, and 0 no linear correlation. In the breast cancer dataset, it helps identify relationships between different features

```
ax=plt.figure(figsize=(25,25))
sns.heatmap(corr,annot=True, vmin=-1, vmax=1, center=0,cmap=sns.diverging_palette(100, 500, n=500),square=True)
```





## 3. ML Classification algorithms

### 3.1 Gather and Prepare Data

The data is already gathered from UC Irvine Machine Learning Repository and all the preprocessing is already done above

### 3.2 Data Splitting

The Data is splitted into Training and Validation(testing) set to train and find tune the selected module.

We have considered training data around 80% and testing data around 20%.


```
[34] X_train, X_test, Y_train, Y_test=train_test_split(X,y,test_size=0.2,random_state=2)

[35] print(X_train.shape,X_test.shape)

(455, 31) (114, 31)
```

### 3.3 Feature Engineering

We have done dimension reduction of the features which are highly correlated with each other using principal component analysis (PCA).



```
pca=PCA(n_components=10)
X_train=pca.fit_transform(X_train)
X_test=pca.transform(X_test)
```

## 3.4 MODEL CLASSIFICATION :

We have used four classification algorithms. These algorithms are as follows :

- **Logistic Regression**

Logistic regression is a statistical model that predicts the probability of an event occurring based on one or more independent variables. It is a popular choice for binary classification tasks, where the outcome can only be two values, such as "yes" or "no", "pass" or "fail", or "spam" or "not spam". In our case it will be "Benign" and "Malignant".

Unlike linear regression, which predicts a continuous numerical value, logistic regression outputs a probability between 0 and 1. This probability represents the likelihood that the event will occur.

Overall, logistic regression is a powerful and versatile statistical model that is widely used for classification tasks. It is a good choice for problems where the outcome is binary and the relationship between the independent variables and the outcome is not too complex.

```
▶ classifier1=LogisticRegression(random_state=0)
  classifier1.fit(X_train, Y_train)
```

```
➡ /usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143:
   y = column_or_1d(y, warn=True)
```

▼ LogisticRegression

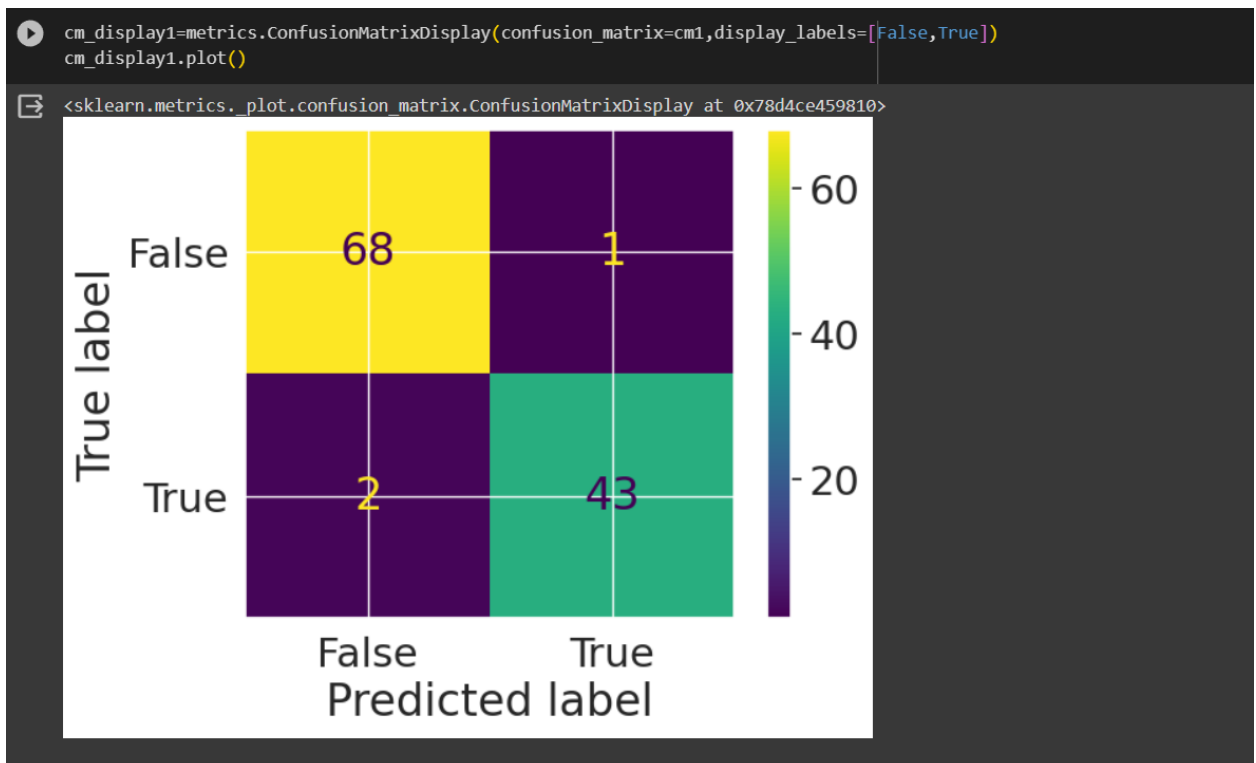
LogisticRegression(random\_state=0)

```
[ ] X_test_prediction_LR=classifier1.predict(X_test)
    testing_data_accuracy_LR=accuracy_score(Y_test,X_test_prediction_LR)

[ ] print('Accuracy on testing data =', testing_data_accuracy_LR)

Accuracy on testing data = 0.9736842105263158
```

Confusion Matrix:



## Classification Report:

```
[ ] print(classification_report(Y_test,X_test_prediction_LR))
```

	precision	recall	f1-score	support
0	0.97	0.99	0.98	69
1	0.98	0.96	0.97	45
accuracy			0.97	114
macro avg	0.97	0.97	0.97	114
weighted avg	0.97	0.97	0.97	114

## • K Nearest Neighbour

K-Nearest Neighbors (KNN) is a simple machine learning algorithm used for classification and regression. It classifies a data point based on the majority class of its k-nearest neighbors or predicts a value based on their average. Key steps include choosing k, calculating distances, identifying neighbors, and making predictions. Considerations include selecting an appropriate distance metric, scaling features, and addressing the curse of dimensionality. KNN is versatile and used in applications like image recognition, regression, and anomaly detection.

```
[ ] classifier3=KNeighborsClassifier()
    classifier3.fit(X_train,Y_train)

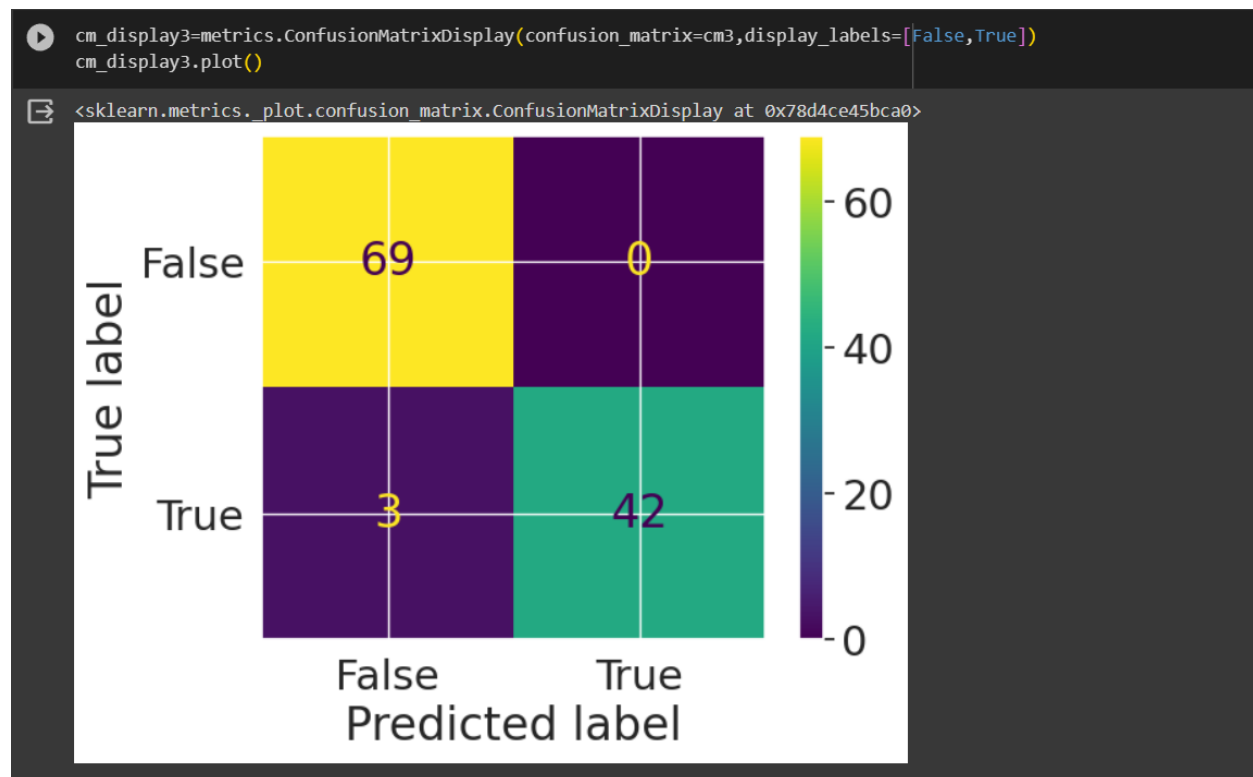
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_classification.py:215:
    return self._fit(X, y)
    ▾ KNeighborsClassifier
    KNeighborsClassifier()
```

```
[ ] X_test_prediction_KNN=classifier3.predict(X_test)
    testing_data_accuracy_KNN=accuracy_score(Y_test,X_test_prediction_KNN)
```

```
[ ] print('Accuracy on testing data =', testing_data_accuracy_KNN)
```

```
Accuracy on testing data = 0.9736842105263158
```

## Confusion Matrix:



## Classification Report:

```
print(classification_report(Y_test,X_test_prediction_KNN))
```

	precision	recall	f1-score	support
0	0.96	1.00	0.98	69
1	1.00	0.93	0.97	45
accuracy			0.97	114
macro avg	0.98	0.97	0.97	114
weighted avg	0.97	0.97	0.97	114

- **Random forest**

Random Forest is an ensemble learning algorithm that builds multiple decision trees during training and merges their predictions for more robust and accurate results. It operates by constructing a "forest" of trees, each trained on a random subset of the data and features. The final prediction is often the average (for regression) or majority vote (for classification) of the individual tree predictions. Random Forest is known for its high accuracy, resilience to overfitting, and applicability to various tasks, including classification and regression.

```
▶ classifier2=RandomForestClassifier()  
classifier2.fit(X_train,Y_train)
```

↳ <ipython-input-170-71c571a74566>:2: DataConversionWarning:  
classifier2.fit(X\_train,Y\_train)

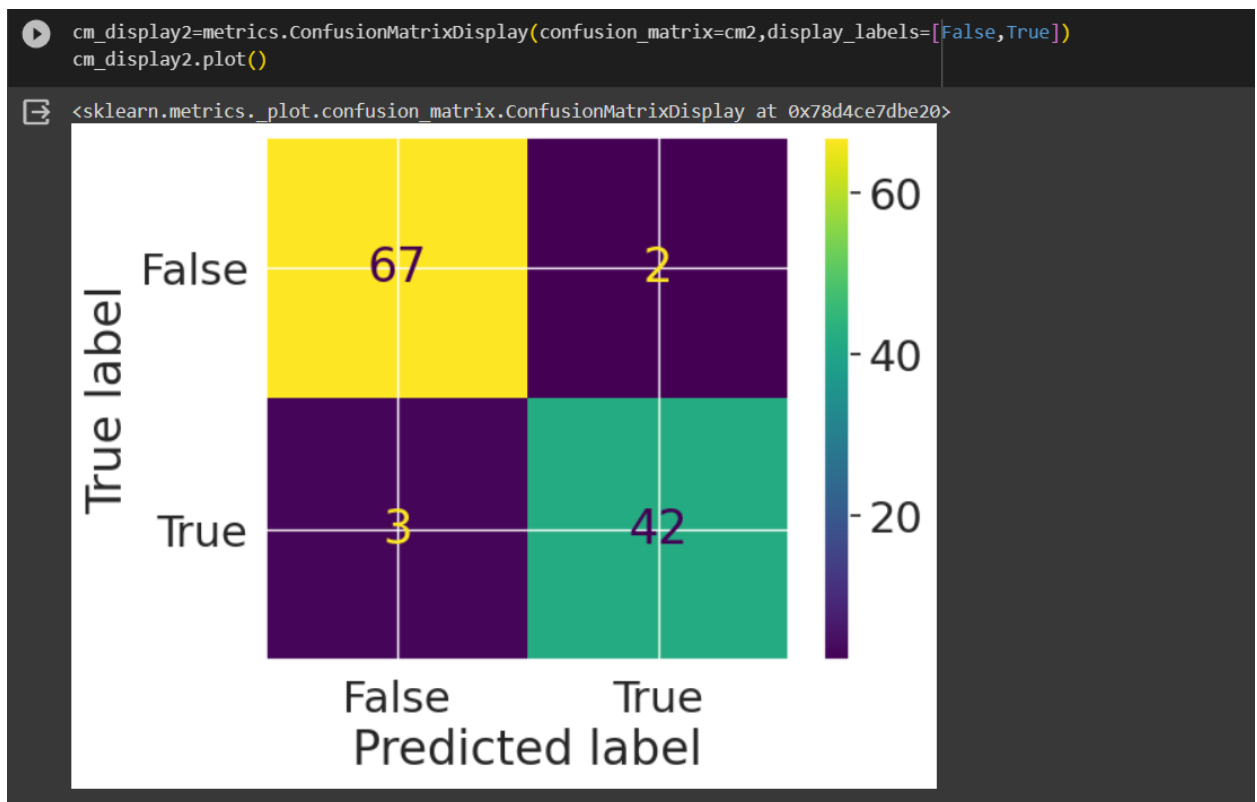
- ▼ RandomForestClassifier
- RandomForestClassifier()

```
[ ] X_test_prediction_RF=classifier2.predict(X_test)  
testing_data_accuracy_RF=accuracy_score(Y_test,X_test_prediction_RF)
```

```
[ ] print('Accuracy on testing data =', testing_data_accuracy_RF)
```

Accuracy on testing data = 0.9649122807017544

## Confusion Matrix:



## Classification Report:

```
[ ] print(classification_report(Y_test,X_test_prediction_RF))
```

	precision	recall	f1-score	support
0	0.96	0.99	0.97	69
1	0.98	0.93	0.95	45
accuracy			0.96	114
macro avg	0.97	0.96	0.96	114
weighted avg	0.97	0.96	0.96	114

- **Support Vector Machine**

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding a hyperplane that best separates data points into different classes. The "support vectors" are the data points closest to the hyperplane. SVM aims to maximize the margin between classes, promoting better generalization to unseen data. It can handle high-dimensional data and is effective in scenarios with complex decision boundaries. SVM is widely used in image classification, text classification, and other pattern recognition tasks.

```
[ ] classifier4=SVC(kernel='linear')
    classifier4.fit(X_train,Y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1143:
  y = column_or_1d(y, warn=True)

▼      SVC
SVC(kernel='linear')
```

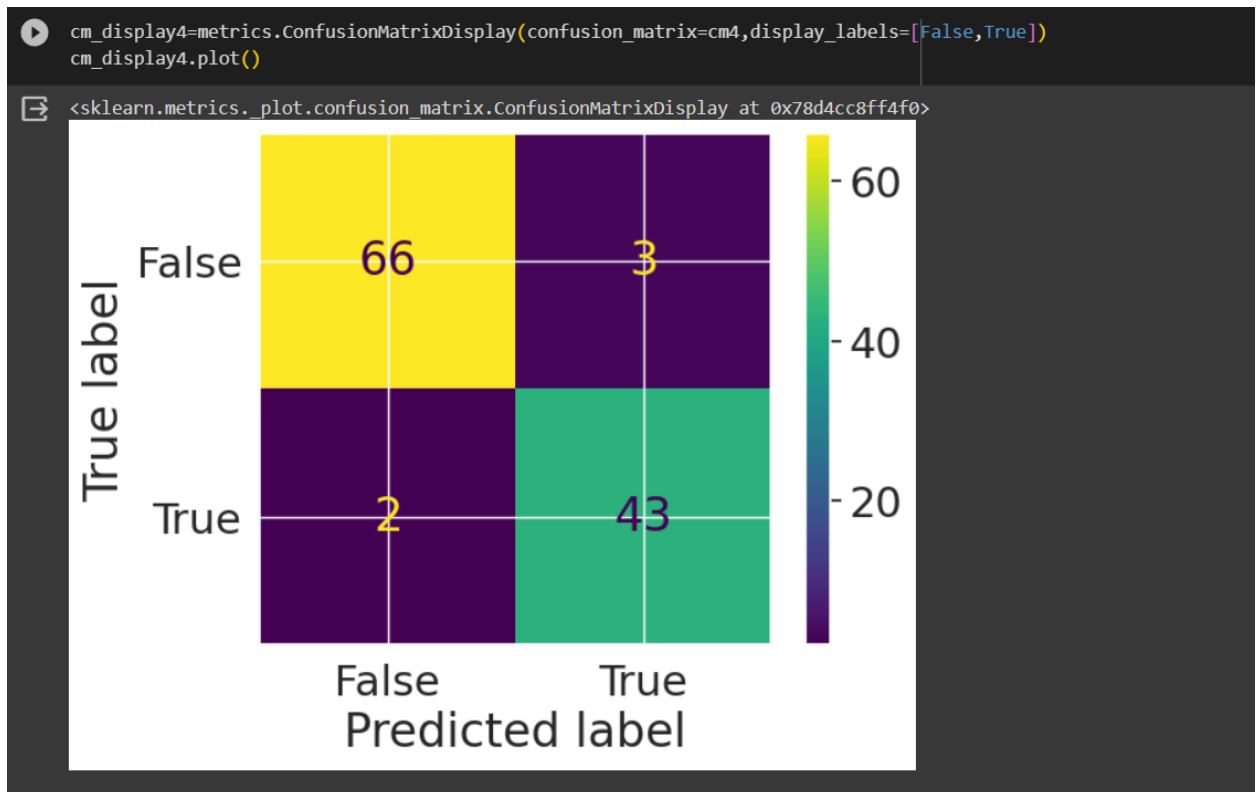
```
[ ] X_test_prediction_SVC=classifier4.predict(X_test)
    testing_data_accuracy_SVC=accuracy_score(Y_test,X_test_prediction_SVC)

[ ] print('Accuracy on testing data =', testing_data_accuracy_SVC)

Accuracy on testing data = 0.956140350877193
```



## Confusion Matrix:



## Classification Report:

```
[ ] print(classification_report(Y_test,X_test_prediction_SVC))
```

	precision	recall	f1-score	support
0	0.97	0.96	0.96	69
1	0.93	0.96	0.95	45
accuracy			0.96	114
macro avg	0.95	0.96	0.95	114
weighted avg	0.96	0.96	0.96	114

## 4. INFERENCES

The project inferences include understanding important features for distinguishing cancer types, evaluating model performance, identifying patterns, and suggesting potential next steps for further investigation.

We have inferred accuracy of different ML algorithms used in our project for classification.

	Classification Algorithm	Accuracy
1.	Logistic Regression	97.36%
2.	K Nearest Neighbour	97.36%
3.	Random Forest	96.49%
4.	Support Vector Machine	95.61%

Further inferred about the prison, Recall and f1 score of the different ML algorithms.

	prison	Recall	F1 score
Logistic Regression	0.97	0.97	0.97
K Nearest Neighbour	0.98	0.97	0.97
Random Forest	0.97	0.96	0.96
Support Vector Machine	0.95	0.96	0.95

## 5. CONCLUSION

In this project , we tried to analysis various features of our dataset and successfully build classifier for our dataset using various Machine Learning Algorithms like K Nearest Neighbour, Logistic Regression, Random Forest, Support vector machine. We find that the best ML algorithm for Classification of dataset is K Nearest Neighbour with accuracy on testing data is 97.36% but same accuracy given by Logistic regression. Thus, we compute various performance metrics like precision, recall and F1- score and find out that K nearest neighbor gives the better precision as compared to logistic regression. Therefore, we concluded that K nearest neighbor is the best ML algorithm for classification of this dataset for our dataset.

The code for the project is uploaded on the github. Here's the link for the github.

## REFERENCES

1. Class notes and Lecture presentations.
2. <https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic>
3. <https://python-data-science.readthedocs.io/en/latest/>
4. <https://scikit-learn.org/stable>
5. <https://github.com/VMvashisht/Breast-Cancer-Classification-using-MI-Algorithms-and-data-analysis>