Cognorise_Infotech_Project_Task1

KHADEEJA THASNEEM

BREAST CANCER CLASSIFICATION

A MACHINE LEARNING PROJECT FOR CLASSIFICATION OF BREAST CANCER INTO MALIGNANT AND BENIGN TUMOR

MALIGNANT : CANCEROUS , FAST GROWING , SPREAD TO OTHER PARTS OF THE BODY.

BENIGN : NON-CANCEROUS , SLOW GROWING , DO NOT SPREAD TO OTHER PARTS OF THE BODY.

```
#Importing the sufficient python libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
#Importing the dataset
df=pd.read_csv('/content/breast_cancer.csv')
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390

3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280
...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362

569 rows × 33 columns

```
#To print the first 5 rows of the dataframe
df.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	co
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	

5 rows × 33 columns

```
#To print the last 5 rows of the dataframe
df.tail()
df.tail()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	cc
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	

5 rows × 33 columns

```
#To print the diamension of the dataframe in (rows, columns) format
df.shape
```

```
(569, 33)
```

```
#To print column names of the dataframe
df.columns
```

```
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', 'Unnamed: 32'],
      dtype='object')
```

```
#To print the datatype of each column
df.dtypes
df.dtypes
```

```
id                           int64
diagnosis                   object
radius_mean                float64
texture_mean               float64
perimeter_mean             float64
area_mean                  float64
smoothness_mean            float64
compactness_mean           float64
concavity_mean             float64
concave points_mean        float64
symmetry_mean              float64
fractal_dimension_mean     float64
radius_se                  float64
texture_se                 float64
perimeter_se               float64
area_se                    float64
smoothness_se              float64
compactness_se             float64
concavity_se               float64
concave points_se          float64
symmetry_se                float64
fractal_dimension_se       float64
radius_worst               float64
texture_worst              float64
perimeter_worst            float64
area_worst                 float64
smoothness_worst           float64
compactness_worst          float64
concavity_worst            float64
concave points_worst       float64
symmetry_worst             float64
fractal_dimension_worst    float64
Unnamed: 32                float64
dtype: object
```

```
#To check whether there is any missing values in the dataframe
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
    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
```

```
#To get a concise summary of the dataframe
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
```

```
#To get statistical measures about the dataset
df.describe()
```

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	co
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	

8 rows × 32 columns

Count: The number of non-null values for each numerical column.

Mean: The mean (average) value.

Std: The standard deviation, a measure of the amount of variation or dispersion.

Min: The minimum value.

25% (Q1): The first quartile, which represents the 25th percentile.

50% (median): The median or 50th percentile.

75% (Q3): The third quartile, which represents the 75th percentile.

Max: The maximum value.

```
#Since ID is not required for prediction we can drop that column
#Also all the values in the column 'Unnamed: 32' are missing, so we can drop that column to
df.drop(['id','Unnamed: 32'],axis=1,inplace=True)
df
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_
0	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3
1	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1
3	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1
...	...	...	...	...	...	...	...	
564	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.2
565	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.1
566	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.0
567	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.3
568	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.0

569 rows × 31 columns

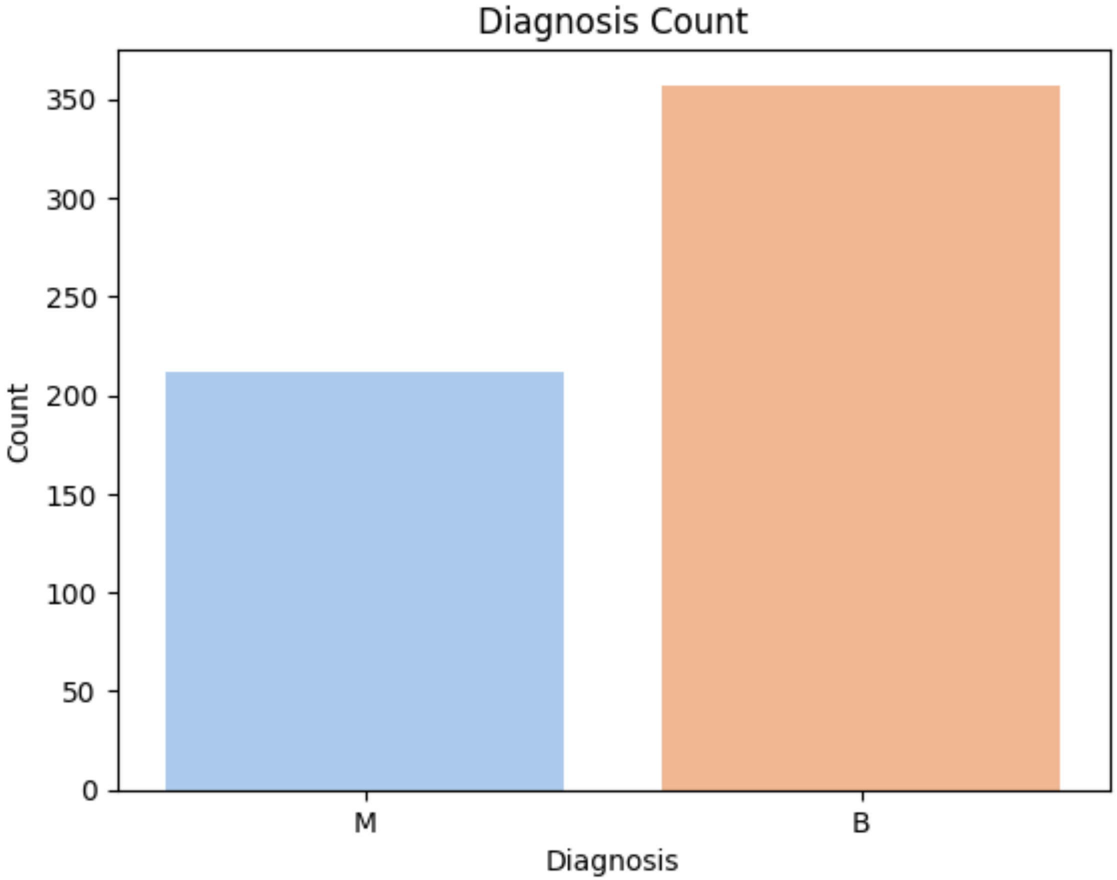
```
#Checking the distribution of the target variable
df['diagnosis'].value_counts()
```

```
B    357
M    212
```

```
Name: diagnosis, dtype: int64
```

```
sns.countplot(x='diagnosis',data=df,hue='diagnosis',palette='pastel')
plt.title('Diagnosis Count')
plt.xlabel('Diagnosis')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



```
df['diagnosis']=df['diagnosis'].map({'M':1,'B':0})
df
```

```
#1 represents Malignant (riskier)
#0 represents Benign
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1
...	...	...	...	...	...	...	...	
564	1	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.2
565	1	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.1
566	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.0
567	1	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.3
568	0	7.76	24.54	47.92	181.0	0.05263	0.04362	0.0

569 rows × 31 columns

```
#To check how much riskier is Malignant than Benign
df.groupby('diagnosis').mean()
```

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
diagnosis							
0	12.146524	17.914762	78.075406	462.790196	0.092478	0.080085	0.046058

1	17.462830	21.604906	115.365377	978.376415	0.102898	0.145188	0.16077

2 rows × 30 columns

```
#Since this dataset has no object input features, no need to perform encoding, also there a
```

OUTLIER DETECTION AND REMOVAL

OUTLIER : A datapoint which is significally far from other data points

IQR : Inter Quartile Range

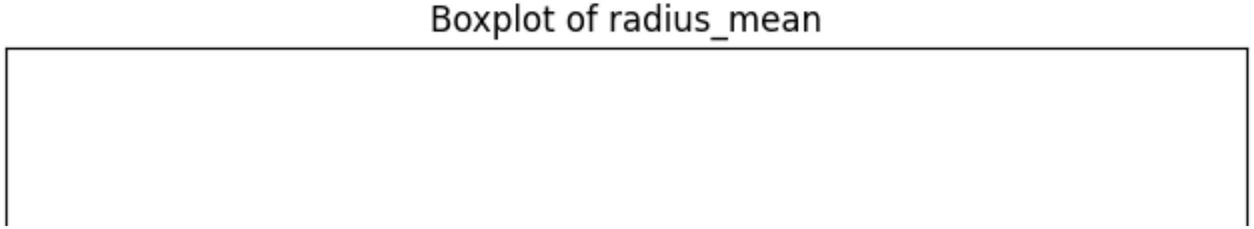
IQR = Q3-Q1

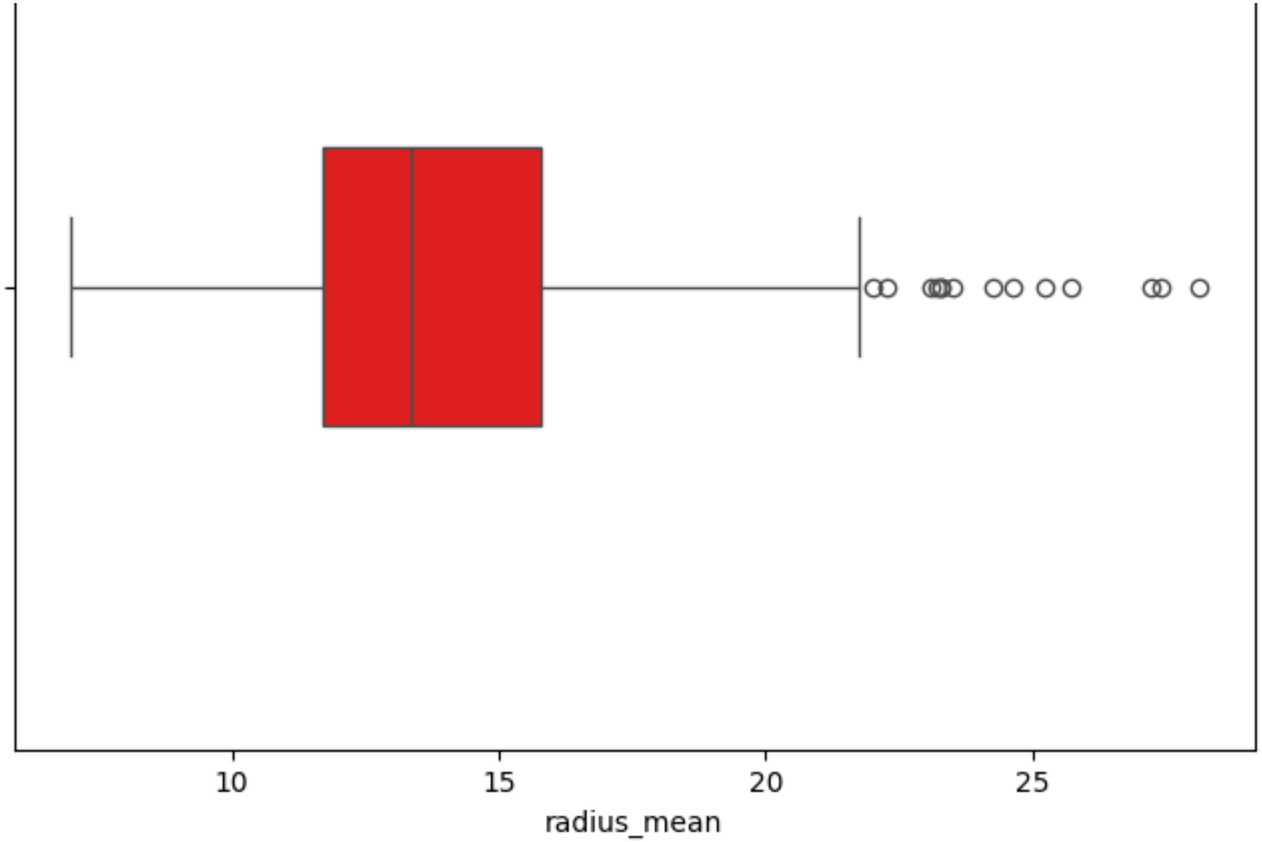
Top = Q3+1.5*IQR

Bottom = Q1-1.5*IQR

Boxplot

```
plt.figure(figsize=(8,6))
sns.boxplot(x='radius_mean',data=df,orient='h',width=0.3,color='red')
plt.title('Boxplot of radius_mean')
plt.show()
```





```
Q1=df.quantile(0.25)
Q3=df.quantile(0.75)
IQR=Q3-Q1
threshold=1.5
df=df[~((df<(Q1-threshold*IQR)) | (df>(Q3+threshold*IQR))).any(axis=1)]
df
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_
6	1	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.1
7	1	13.71	20.83	90.20	577.9	0.11890	0.16450	0.0

7	1	13.71	20.83	90.20	577.9	0.11890	0.16450	0.0
10	1	16.02	23.24	102.70	797.8	0.08206	0.06669	0.0
11	1	15.78	17.89	103.60	781.0	0.09710	0.12920	0.0
13	1	15.85	23.95	103.70	782.7	0.08401	0.10020	0.0
...	...	...	...	...	...	...	...	
554	0	12.88	28.92	82.50	514.3	0.08123	0.05824	0.0
555	0	10.29	27.61	65.67	321.4	0.09030	0.07658	0.0
558	0	14.59	22.68	96.39	657.1	0.08473	0.13300	0.1
560	0	14.05	27.15	91.38	600.4	0.09929	0.11260	0.0
566	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.0

398 rows × 31 columns

FEATURE SELECTION

```
#To print the correlation matrix
#Checking correlations of different features with the target variable.
df.corr()
```

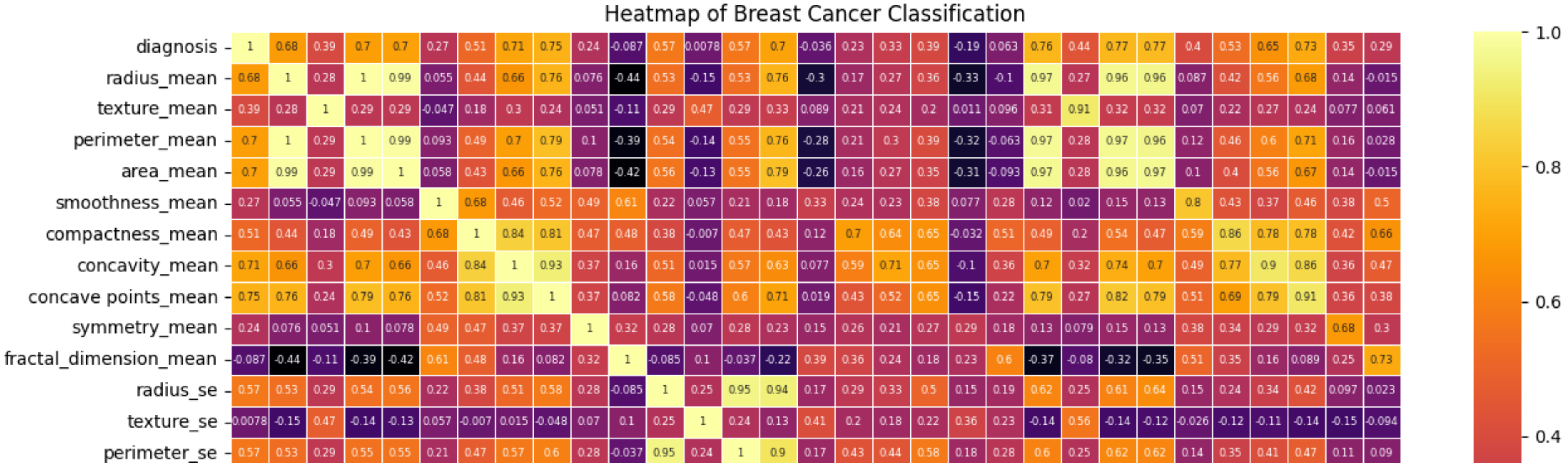
	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactn
diagnosis	1.000000	0.680705	0.388596	0.695835	0.698005	0.271570	
radius_mean	0.680705	1.000000	0.280796	0.998113	0.992047	0.055311	
texture_mean	0.388596	0.280796	1.000000	0.285880	0.288688	-0.046879	
perimeter_mean	0.695835	0.998113	0.285880	1.000000	0.990235	0.093106	

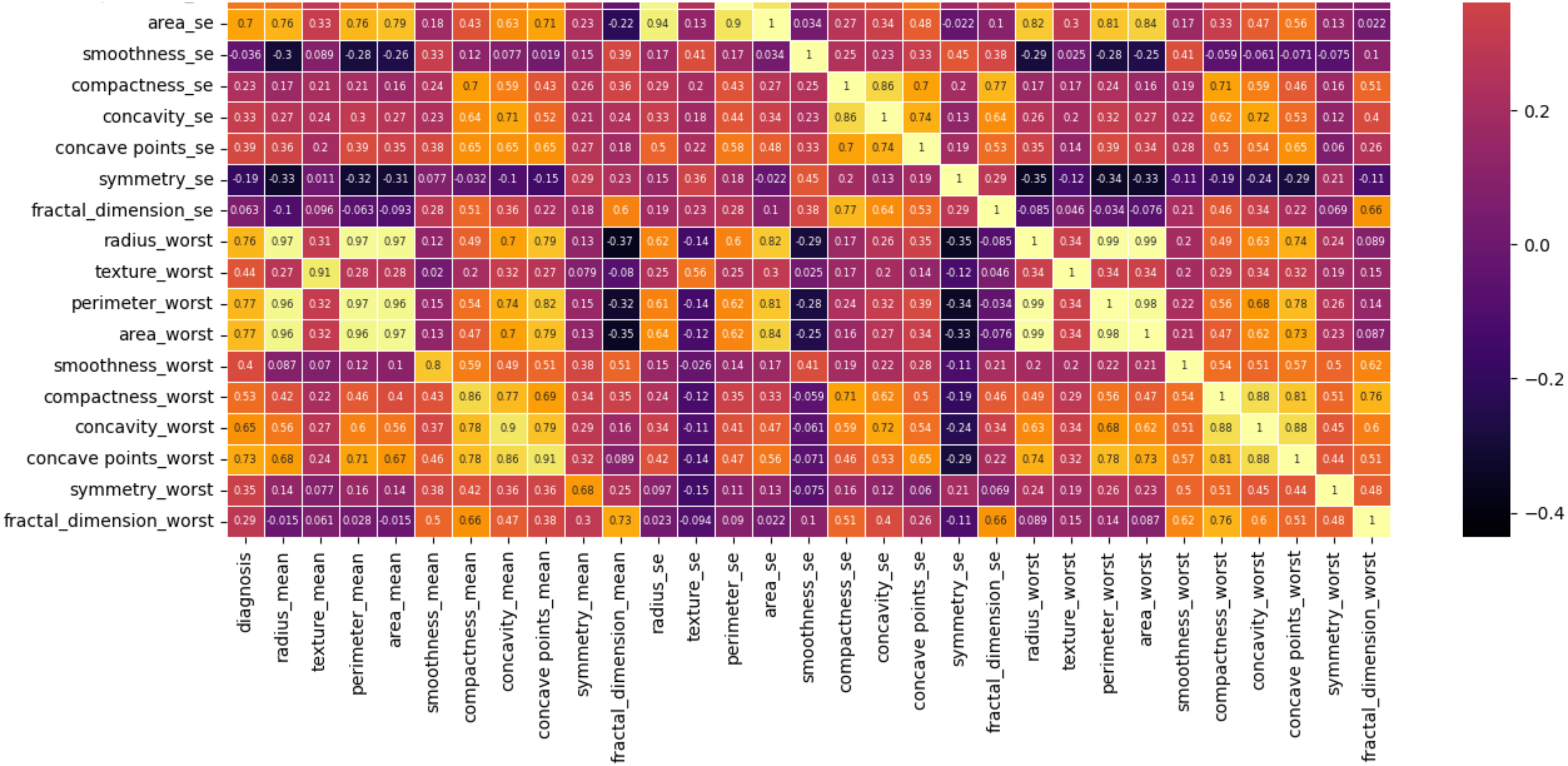
perimeter_mean	0.695835	0.998113	0.285880	1.000000	0.990235	0.093196	
area_mean	0.698005	0.992047	0.288688	0.990235	1.000000	0.058280	
smoothness_mean	0.271570	0.055311	-0.046879	0.093196	0.058280	1.000000	
compactness_mean	0.514346	0.442360	0.181078	0.492915	0.428703	0.675183	
concavity_mean	0.710567	0.658031	0.295111	0.696240	0.664847	0.463982	
concave points_mean	0.752190	0.755607	0.239147	0.785954	0.758145	0.522678	
symmetry_mean	0.243588	0.075653	0.051037	0.102371	0.077918	0.494062	
fractal_dimension_mean	-0.086631	-0.436397	-0.110214	-0.393736	-0.423006	0.614693	
radius_se	0.571842	0.534801	0.290195	0.542837	0.561732	0.224720	
texture_se	0.007755	-0.149935	0.474188	-0.144895	-0.132103	0.056968	-
perimeter_se	0.565909	0.530347	0.294101	0.546622	0.554631	0.213312	
area_se	0.703715	0.758127	0.325637	0.763671	0.786897	0.181729	
smoothness_se	-0.036030	-0.301325	0.089100	-0.283129	-0.260649	0.330804	
compactness_se	0.229104	0.165214	0.212798	0.207785	0.161122	0.240507	
concavity_se	0.325974	0.265486	0.242464	0.302232	0.268137	0.230219	
concave points_se	0.387972	0.363046	0.198599	0.394612	0.354173	0.378191	
symmetry_se	-0.194890	-0.327179	0.011039	-0.318263	-0.308046	0.077346	-
fractal_dimension_se	0.062799	-0.100183	0.095912	-0.063267	-0.093091	0.283411	
radius_worst	0.761081	0.971286	0.311152	0.972912	0.969353	0.122928	
texture_worst	0.437909	0.273847	0.914721	0.280552	0.280953	0.020158	
perimeter_worst	0.768194	0.964851	0.316562	0.971558	0.962721	0.146367	
area_worst	0.773626	0.957772	0.317755	0.959710	0.971053	0.125280	
smoothness_worst	0.396847	0.087209	0.069886	0.121237	0.102255	0.803945	

_						
compactness_worst	0.533996	0.416464	0.216026	0.460901	0.400703	0.432375
concavity_worst	0.653300	0.563240	0.274070	0.599672	0.559618	0.372220
concave points_worst	0.726639	0.682235	0.238186	0.712906	0.671695	0.463910
symmetry_worst	0.347406	0.137655	0.076977	0.160178	0.136700	0.375969
fractal_dimension_worst	0.290821	-0.014583	0.060915	0.027877	-0.014754	0.504310

31 rows × 31 columns

```
#To visualize the correlation matrix graphically
#Heatmap
plt.figure(figsize=(15,10))
corr=df.corr()
sns.heatmap(corr,cmap='inferno',linewidths=.5,annot=True,annot_kws={"fontsize":6})
plt.title("Heatmap of Breast Cancer Classification")
plt.show()
```





	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
area_se	0.7	0.76	0.33	0.76	0.79	0.18	0.43	0.63	0.71	0.23	-0.22	0.94	0.13	0.9	1	0.034	0.27	0.34	0.48	-0.022	0.1	0.82	0.3	0.81	0.84	0.17	0.33	0.47	0.56	0.13	0.022
smoothness_se	-0.036	-0.3	0.089	-0.28	-0.26	0.33	0.12	0.077	0.019	0.15	0.39	0.17	0.41	0.17	0.034	1	0.25	0.23	0.33	0.45	0.38	-0.29	0.025	-0.28	-0.25	0.41	-0.059	-0.061	-0.071	-0.075	0.1
compactness_se	0.23	0.17	0.21	0.21	0.16	0.24	0.7	0.59	0.43	0.26	0.36	0.29	0.2	0.43	0.27	0.25	1	0.86	0.7	0.2	0.77	0.17	0.17	0.24	0.16	0.19	0.71	0.59	0.46	0.16	0.51
concavity_se	0.33	0.27	0.24	0.3	0.27	0.23	0.64	0.71	0.52	0.21	0.24	0.33	0.18	0.44	0.34	0.23	0.86	1	0.74	0.13	0.64	0.26	0.2	0.32	0.27	0.22	0.62	0.72	0.53	0.12	0.4
concave points_se	0.39	0.36	0.2	0.39	0.35	0.38	0.65	0.65	0.65	0.27	0.18	0.5	0.22	0.58	0.48	0.33	0.7	0.74	1	0.19	0.53	0.35	0.14	0.39	0.34	0.28	0.5	0.54	0.65	0.06	0.26
symmetry_se	-0.19	-0.33	0.011	-0.32	-0.31	0.077	-0.032	-0.1	-0.15	0.29	0.23	0.15	0.36	0.18	-0.022	0.45	0.2	0.13	0.19	1	0.29	-0.35	-0.12	-0.34	-0.33	-0.11	-0.19	-0.24	-0.29	0.21	-0.11
fractal_dimension_se	0.063	-0.1	0.096	-0.063	-0.093	0.28	0.51	0.36	0.22	0.18	0.6	0.19	0.23	0.28	0.1	0.38	0.77	0.64	0.53	0.29	1	-0.085	0.046	-0.034	-0.076	0.21	0.46	0.34	0.22	0.069	0.66
radius_worst	0.76	0.97	0.31	0.97	0.97	0.12	0.49	0.7	0.79	0.13	-0.37	0.62	-0.14	0.6	0.82	-0.29	0.17	0.26	0.35	-0.35	-0.085	1	0.34	0.99	0.99	0.2	0.49	0.63	0.74	0.24	0.089
texture_worst	0.44	0.27	0.91	0.28	0.28	0.02	0.2	0.32	0.27	0.079	-0.08	0.25	0.56	0.25	0.3	0.025	0.17	0.2	0.14	-0.12	0.046	0.34	1	0.34	0.34	0.2	0.29	0.34	0.32	0.19	0.15
perimeter_worst	0.77	0.96	0.32	0.97	0.96	0.15	0.54	0.74	0.82	0.15	-0.32	0.61	-0.14	0.62	0.81	-0.28	0.24	0.32	0.39	-0.34	-0.034	0.99	0.34	1	0.98	0.22	0.56	0.68	0.78	0.26	0.14
area_worst	0.77	0.96	0.32	0.96	0.97	0.13	0.47	0.7	0.79	0.13	-0.35	0.64	-0.12	0.62	0.84	-0.25	0.16	0.27	0.34	-0.33	-0.076	0.99	0.34	0.98	1	0.21	0.47	0.62	0.73	0.23	0.087
smoothness_worst	0.4	0.087	0.07	0.12	0.1	0.8	0.59	0.49	0.51	0.38	0.51	0.15	-0.026	0.14	0.17	0.41	0.19	0.22	0.28	-0.11	0.21	0.2	0.2	0.22	0.21	1	0.54	0.51	0.57	0.5	0.62
compactness_worst	0.53	0.42	0.22	0.46	0.4	0.43	0.86	0.77	0.69	0.34	0.35	0.24	-0.12	0.35	0.33	-0.059	0.71	0.62	0.5	-0.19	0.46	0.49	0.29	0.56	0.47	0.54	1	0.88	0.81	0.51	0.76
concavity_worst	0.65	0.56	0.27	0.6	0.56	0.37	0.78	0.9	0.79	0.29	0.16	0.34	-0.11	0.41	0.47	-0.061	0.59	0.72	0.54	-0.24	0.34	0.63	0.34	0.68	0.62	0.51	0.88	1	0.88	0.45	0.6
concave points_worst	0.73	0.68	0.24	0.71	0.67	0.46	0.78	0.86	0.91	0.32	0.089	0.42	-0.14	0.47	0.56	-0.071	0.46	0.53	0.65	-0.29	0.22	0.74	0.32	0.78	0.73	0.57	0.81	0.88	1	0.44	0.51
symmetry_worst	0.35	0.14	0.077	0.16	0.14	0.38	0.42	0.36	0.36	0.68	0.25	0.097	-0.15	0.11	0.13	-0.075	0.16	0.12	0.06	0.21	0.069	0.24	0.19	0.26	0.23	0.5	0.51	0.45	0.44	1	0.48
fractal_dimension_worst	0.29	-0.015	0.061	0.028	-0.015	0.5	0.66	0.47	0.38	0.3	0.73	0.023	-0.094	0.09	0.022	0.1	0.51	0.4	0.26	-0.11	0.66	0.089	0.15	0.14	0.087	0.62	0.76	0.6	0.51	0.48	1

Applying **ANNOVA TEST** for Feature Selection

```
df_copy=df.copy()
x_copy=df_copy.drop(['diagnosis'],axis=1)
y_copy=df['diagnosis']
from sklearn.feature_selection import f_classif
score2=f_classif(x_copy,y_copy)
score2
```

```
(array([3.41923458e+02, 7.04350576e+01, 3.71719551e+02, 3.76248046e+02,
        3.15303773e+01, 1.42446996e+02, 4.03848026e+02, 5.15999640e+02,
        2.49788576e+01, 2.99440304e+00, 1.92412626e+02, 2.38191066e-02,
        1.86569790e+02, 3.88492502e+02, 5.14731747e-01, 2.19369677e+01,
        4.70812572e+01, 7.01688565e+01, 1.56347914e+01, 1.56789246e+00,
        5.45163293e+02, 9.39559734e+01, 5.70142066e+02, 5.90294605e+02,
        7.40225711e+01, 1.57963547e+02, 2.94858852e+02, 4.42989671e+02,
        5.43535467e+01, 3.65869335e+01]),
 array([1.76202025e-55, 8.51880326e-16, 6.80281885e-59, 2.11648603e-59,
        3.70131299e-08, 2.92362441e-28, 1.98753990e-62, 9.78048268e-74,
        8.71811279e-07, 8.43317446e-02, 6.16508492e-36, 8.77424701e-01,
        4.49299726e-35, 9.31925157e-61, 4.73520649e-01, 3.87829076e-06,
        2.63532505e-11, 9.55311414e-16, 9.10031278e-05, 2.11252375e-01,
        1.89923667e-76, 4.46169928e-20, 1.05225985e-78, 1.75342370e-80,
        1.83097645e-16, 1.01615082e-30, 8.52732130e-50, 1.51484069e-66,
        9.87189587e-13, 3.38223856e-09]))
```

```
f_value2=pd.Series(score2[0],index=x_copy.columns)
f_value2.sort_values(ascending=False)
```

```
area_worst                 590.294605
perimeter_worst            570.142066
radius_worst               545.163293
concave points_mean        515.999640
concave points_mean        515.999640
concave points_worst       442.989671
concavity_mean             403.848026
area_se                    388.492502
area_mean                  376.248046
perimeter_mean             371.719551
radius_mean                341.923458
concavity_worst            294.858852
radius_se                  192.412626
perimeter_se               186.569790
compactness_worst          157.963547
compactness_mean           142.446996
texture_worst               93.955973
smoothness_worst            74.022571
texture_mean                70.435058
concave points_se           70.168856
symmetry_worst              54.353547
concavity_se                47.081257
fractal_dimension_worst     36.586934
smoothness_mean             31.530377
symmetry_mean               24.978858
compactness_se              21.936968
symmetry_se                 15.634791
fractal_dimension_mean       2.994403
fractal_dimension_se         1.567892
smoothness_se                0.514732
texture_se                   0.023819
dtype: float64
```

```
p_value2=pd.Series(score2[1],index=x_copy.columns)
p_value2.sort_values(ascending=False)
```

```
texture_se                 8.774247e-01
smoothness_se              4.735206e-01
fractal_dimension_se       2.112524e-01
fractal_dimension_mean     8.433174e-02
symmetry_se                9.100313e-05
compactness_se             3.878291e-06
symmetry_mean              8.718113e-07
smoothness_mean            3.701313e-08
```

```
fractal_dimension_worst    3.382239e-09
concavity_se               2.635325e-11
symmetry_worst             9.871896e-13
concave points_se          9.553114e-16
texture_mean               8.518803e-16
smoothness_worst           1.830976e-16
texture_worst              4.461699e-20
compactness_mean           2.923624e-28
compactness_worst          1.016151e-30
perimeter_se               4.492997e-35
radius_se                  6.165085e-36
concavity_worst            8.527321e-50
radius_mean                1.762020e-55
perimeter_mean             6.802819e-59
area_mean                  2.116486e-59
area_se                    9.319252e-61
concavity_mean             1.987540e-62
concave points_worst       1.514841e-66
concave points_mean        9.780483e-74
radius_worst               1.899237e-76
perimeter_worst            1.052260e-78
area_worst                 1.753424e-80
dtype: float64
```

```
#Annova test : we can drop the columns 'fractal_dimension_se','smoothness_se','fractal_dime
df.drop(['fractal_dimension_se','smoothness_se','fractal_dimension_mean','texture_se'],axis
df
```

```
<ipython-input-24-c4208603c380>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retur
  df.drop(['fractal_dimension_se','smoothness_se','fractal_dimension_mean','texture_se'],axis=1,inplace=True)
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_
6	1	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.1
7	1	13.71	20.83	90.20	577.9	0.11890	0.16450	0.0

10	1	16.02	23.24	102.70	797.8	0.08206	0.06669	0.0
11	1	15.78	17.89	103.60	781.0	0.09710	0.12920	0.0
13	1	15.85	23.95	103.70	782.7	0.08401	0.10020	0.0
...	...	...	...	...	...	...	...	
554	0	12.88	28.92	82.50	514.3	0.08123	0.05824	0.0
555	0	10.29	27.61	65.67	321.4	0.09030	0.07658	0.0
558	0	14.59	22.68	96.39	657.1	0.08473	0.13300	0.1
560	0	14.05	27.15	91.38	600.4	0.09929	0.11260	0.0
566	1	16.60	28.08	108.30	858.1	0.08455	0.10230	0.0

398 rows × 27 columns

SEPARATING THE INPUT AND OUTPUT FEATURES AS X AND Y

```
x=df.drop(['diagnosis'],axis=1)
y=df['diagnosis']
```

BALANCING THE DATASET

```
from imblearn.over_sampling import SMOTE
smote=SMOTE()
x_resampled,y_resampled=smote.fit_resample(x,y)
```

```
from collections import Counter
print("Before Smote :",Counter(y))
print("After Smote :",Counter(y_resampled))
```

```
    Before Smote : Counter({0: 300, 1: 98})
    After Smote : Counter({1: 300, 0: 300})
```

SPLITTING INTO TRAINING AND TESTING DATA

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_resampled,y_resampled,test_size=0.30,rando
y_train
```

```
    108    1
    272    0
    599    1
    479    1
    436    1
          ..
    71     0
    106    0
    270    0
    435    1
    102    0
    Name: diagnosis, Length: 420, dtype: int64
```

NORMALIZATION USING STANDARDSCALER

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x train)
```

```
nx_train=scaler.transform(x_train)
nx_test=scaler.transform(x_test)
```

MODEL CREATION USING K-NEAREST NEIGHBORS CLASSIFIER

GRIDSEARCHCV

```
from sklearn.neighbors import KNeighborsClassifier
knn1=KNeighborsClassifier()
param={'n_neighbors':[3,5,7,9],'weights':['uniform','distance']}

from sklearn.model_selection import GridSearchCV
clf=GridSearchCV(knn1,param,cv=10,scoring='accuracy')
clf.fit(nx_train,y_train)
print(clf.best_params_)
```

```
{'n_neighbors': 5, 'weights': 'distance'}
```

```
#Model creation
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=3,weights='distance')
knn.fit(nx_train,y_train)
y_prediction=knn.predict(nx_test)
y_prediction
```

```
array([0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 0])
```

PERFORMANCE EVALUATION

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,Confusion
#confusion matrix
matr=confusion_matrix(y_test,y_prediction)
print(matr)
```

```
[[75  3]
 [ 6 96]]
```

```
#accuracy score
score=accuracy_score(y_test,y_prediction)
score
```

```
0.95
```

```
#classification report
report=classification_report(y_test,y_prediction)
print(report)
```

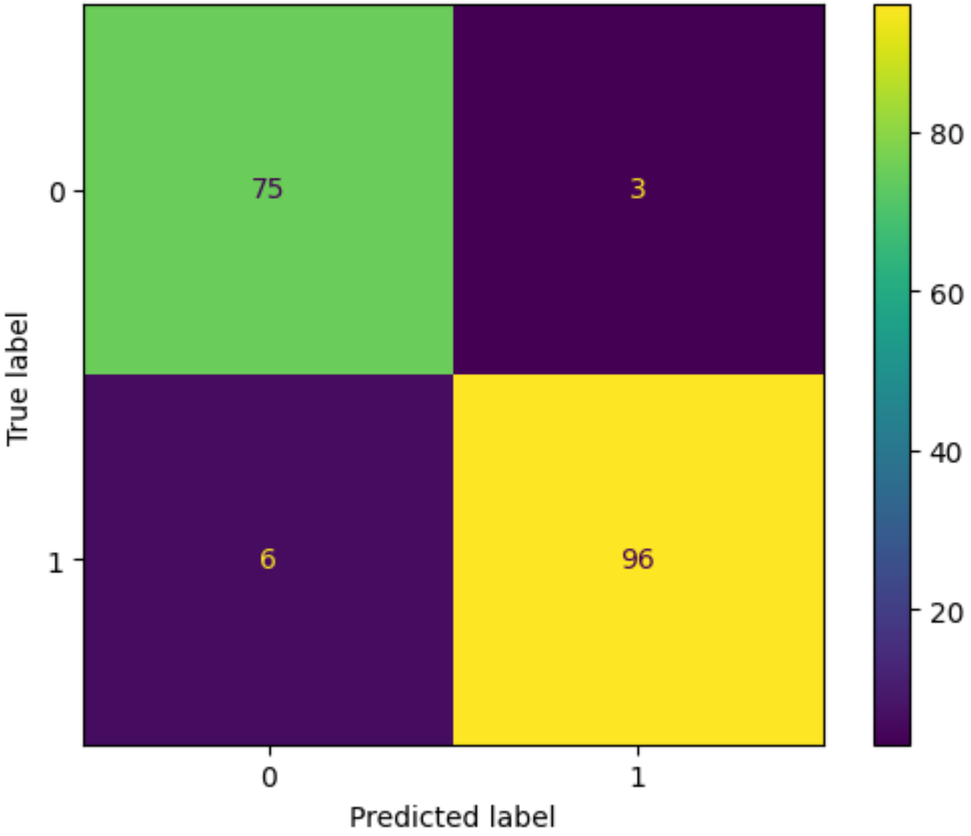
```
              precision    recall  f1-score   support

           0       0.93      0.96      0.94        78
           1       0.97      0.94      0.96       102

    accuracy                           0.95       180
   macro avg       0.95      0.95      0.95       180
weighted avg       0.95      0.95      0.95       180
```

```
#confusion matrix display
labels=[0,1]
cmd=ConfusionMatrixDisplay(matr,display_labels=labels)
cmd.plot()
#0 : Benign
#1 : Malignant(riskier)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7b3ed84c5d20>
```



BUILDING A PREDICTIVE SYSTEM

BUILDING A PREDICTIVE SYSTEM

```
input_data=[[13,21.82,87.5,519.8,0.1273,0.1932,0.1859,0.09353,0.235,0.3063,2.406,24.32,0.03
prediction=knn.predict(input_data)
if (prediction[0]==1):
  print("The Breast Cancer is Malignant")
else:
  print("The Breast Cancer is Benign")
```

```
The Breast Cancer is Malignant
```

MODEL CREATION USING DECISION TREE CLASSIFIER, RANDOMFOREST CLASSIFIER, LOGISTIC REGRESSION AND THEN PERFORMANCE EVALUATION

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

dtc=DecisionTreeClassifier(criterion='entropy')
rfc=RandomForestClassifier(n_estimators=100,criterion='entropy')
lr=LogisticRegression(solver='saga',max_iter=100,class_weight='balanced')

lst=[dtc,rfc,lr]
for i in lst:
  print(i)
  i.fit(x_train,y_train)
  y_prediction=i.predict(x_test)
  print(confusion_matrix(y_test,y_prediction))
  print(accuracy_score(y_test,y_prediction))
  print(accuracy_score(y_test,y_prediction))
  print(classification_report(y_test,y_prediction))
```

```
DecisionTreeClassifier(criterion='entropy')
[[76  2]
 [ 7 95]]
0.95
              precision    recall  f1-score   support

           0       0.92      0.97      0.94        78
           1       0.98      0.93      0.95       102

    accuracy                           0.95       180
   macro avg       0.95      0.95      0.95       180
weighted avg       0.95      0.95      0.95       180

RandomForestClassifier(criterion='entropy')
[[74  4]
 [ 4 98]]
0.9555555555555556
              precision    recall  f1-score   support

           0       0.95      0.95      0.95        78
           1       0.96      0.96      0.96       102

    accuracy                           0.96       180
   macro avg       0.95      0.95      0.95       180
weighted avg       0.96      0.96      0.96       180

LogisticRegression(class_weight='balanced', solver='saga')
[[71  7]
 [11 91]]
0.9
              precision    recall  f1-score   support

           0       0.87      0.91      0.89        78
           1       0.93      0.89      0.91       102

    accuracy                           0.90       180
   macro avg       0.90      0.90      0.90       180
weighted avg       0.90      0.90      0.90       180
weighted avg       0.90      0.90      0.90       180

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_sag.py:350: ConvergenceWarning: The max_iter was rea
  warnings.warn(
```