```python
import pandas as pd
import numpy as np
import  matplotlib.pyplot as plt
import  seaborn as sns
import missingno as msno


import warnings
warnings.filterwarnings('ignore')


sns.set()
plt.style.use('ggplot')
```

```
pip install missingno
```

```
Collecting missingno
  Downloading missingno-0.5.2-py3-none-any.whl.metadata (639 bytes)
Requirement already satisfied: numpy in c:\users\preet\python1\envs\
notebook-7.0.8\lib\site-packages (from missingno) (1.26.4)
Requirement already satisfied: matplotlib in c:\users\preet\python1\
envs\notebook-7.0.8\lib\site-packages (from missingno) (3.9.0)
Requirement already satisfied: scipy in c:\users\preet\python1\envs\
notebook-7.0.8\lib\site-packages (from missingno) (1.13.1)
Requirement already satisfied: seaborn in c:\users\preet\python1\envs\
notebook-7.0.8\lib\site-packages (from missingno) (0.13.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\preet\
python1\envs\notebook-7.0.8\lib\site-packages (from matplotlib-
>missingno) (1.2.1)
Requirement already satisfied: cycler>=0.10 in c:\users\preet\python1\
envs\notebook-7.0.8\lib\site-packages (from matplotlib->missingno)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\preet\
python1\envs\notebook-7.0.8\lib\site-packages (from matplotlib-
>missingno) (4.53.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\preet\
python1\envs\notebook-7.0.8\lib\site-packages (from matplotlib-
>missingno) (1.4.5)
Requirement already satisfied: packaging>=20.0 in c:\users\preet\
python1\envs\notebook-7.0.8\lib\site-packages (from matplotlib-
>missingno) (23.2)
Requirement already satisfied: pillow>=8 in c:\users\preet\python1\
envs\notebook-7.0.8\lib\site-packages (from matplotlib->missingno)
(10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\preet\
python1\envs\notebook-7.0.8\lib\site-packages (from matplotlib-
>missingno) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\preet\
python1\envs\notebook-7.0.8\lib\site-packages (from matplotlib-
>missingno) (2.9.0.post0)
```

```
Requirement already satisfied: pandas>=1.2 in c:\users\preet\python1\
envs\notebook-7.0.8\lib\site-packages (from seaborn->missingno)
(2.2.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\preet\python1\
envs\notebook-7.0.8\lib\site-packages (from pandas>=1.2->seaborn-
>missingno) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\preet\
python1\envs\notebook-7.0.8\lib\site-packages (from pandas>=1.2-
>seaborn->missingno) (2024.1)
Requirement already satisfied: six>=1.5 in c:\users\preet\python1\
envs\notebook-7.0.8\lib\site-packages (from python-dateutil>=2.7-
>matplotlib->missingno) (1.16.0)
Downloading missingno-0.5.2-py3-none-any.whl (8.7 kB)
Installing collected packages: missingno
Successfully installed missingno-0.5.2
Note: you may need to restart the kernel to use updated packages.
```

Data collection and Data cleaning

```
data=pd.read_csv(r"C:\Preet\breast-cancer.csv")

data.head()

         id diagnosis  radius_mean  texture_mean  perimeter_mean
area_mean  \
0    842302         M        17.99         10.38          122.80
1001.0
1    842517         M        20.57         17.77          132.90
1326.0
2  84300903         M        19.69         21.25          130.00
1203.0
3  84348301         M        11.42         20.38           77.58
386.1
4  84358402         M        20.29         14.34          135.10
1297.0

   smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
0.10520
4          0.10030           0.13280          0.1980
0.10430

   ...  radius_worst  texture_worst  perimeter_worst  area_worst  \
```

```
0  ...        25.38         17.33           184.60      2019.0
1  ...        24.99         23.41           158.80      1956.0
2  ...        23.57         25.53           152.50      1709.0
3  ...        14.91         26.50            98.87       567.7
4  ...        22.54         16.67           152.20      1575.0

   smoothness_worst  compactness_worst  concavity_worst  concave
points_worst  \
0            0.1622             0.6656           0.7119
0.2654
1            0.1238             0.1866           0.2416
0.1860
2            0.1444             0.4245           0.4504
0.2430
3            0.2098             0.8663           0.6869
0.2575
4            0.1374             0.2050           0.4000
0.1625

   symmetry_worst  fractal_dimension_worst
0          0.4601                  0.11890
1          0.2750                  0.08902
2          0.3613                  0.08758
3          0.6638                  0.17300
4          0.2364                  0.07678

[5 rows x 32 columns]
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 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
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

```
data.diagnosis.unique()
```

```
array(['M', 'B'], dtype=object)
```

```
#M--> Malignant
#B-->Benign

#Supeversied -->target
#UnSupversied

data.describe()
```

```
                 id  radius_mean  texture_mean  perimeter_mean
area_mean  \
count  5.690000e+02   569.000000    569.000000      569.000000
569.000000
mean   3.037183e+07    14.127292     19.289649       91.969033
654.889104
std    1.250206e+08     3.524049      4.301036       24.298981
351.914129
min    8.670000e+03     6.981000      9.710000       43.790000
143.500000
25%    8.692180e+05    11.700000     16.170000       75.170000
420.300000
50%    9.060240e+05    13.370000     18.840000       86.240000
551.100000
75%    8.813129e+06    15.780000     21.800000      104.100000
782.700000
max    9.113205e+08    28.110000     39.280000      188.500000
2501.000000
```

```
       smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
count       569.000000        569.000000      569.000000
569.000000
mean          0.096360          0.104341        0.088799
0.048919
std           0.014064          0.052813        0.079720
0.038803
min           0.052630          0.019380        0.000000
0.000000
25%           0.086370          0.064920        0.029560
0.020310
50%           0.095870          0.092630        0.061540
0.033500
75%           0.105300          0.130400        0.130700
0.074000
max           0.163400          0.345400        0.426800
0.201200

       symmetry_mean  ...  radius_worst  texture_worst
perimeter_worst  \
count     569.000000  ...    569.000000     569.000000
569.000000
mean        0.181162  ...     16.269190      25.677223
107.261213
std         0.027414  ...      4.833242       6.146258
33.602542
min         0.106000  ...      7.930000      12.020000
50.410000
25%         0.161900  ...     13.010000      21.080000
84.110000
50%         0.179200  ...     14.970000      25.410000
97.660000
75%         0.195700  ...     18.790000      29.720000
125.400000
max         0.304000  ...     36.040000      49.540000
251.200000

        area_worst  smoothness_worst  compactness_worst
concavity_worst  \
count   569.000000        569.000000         569.000000
569.000000
mean    880.583128          0.132369           0.254265
0.272188
std     569.356993          0.022832           0.157336
0.208624
min     185.200000          0.071170           0.027290
0.000000
25%     515.300000          0.116600           0.147200
```

```
0.114500
50%     686.500000          0.131300           0.211900
0.226700
75%    1084.000000          0.146000           0.339100
0.382900
max    4254.000000          0.222600           1.058000
1.252000

       concave points_worst  symmetry_worst  fractal_dimension_worst
count            569.000000      569.000000               569.000000
mean               0.114606        0.290076                 0.083946
std                0.065732        0.061867                 0.018061
min                0.000000        0.156500                 0.055040
25%                0.064930        0.250400                 0.071460
50%                0.099930        0.282200                 0.080040
75%                0.161400        0.317900                 0.092080
max                0.291000        0.663800                 0.207500

[8 rows x 31 columns]
```

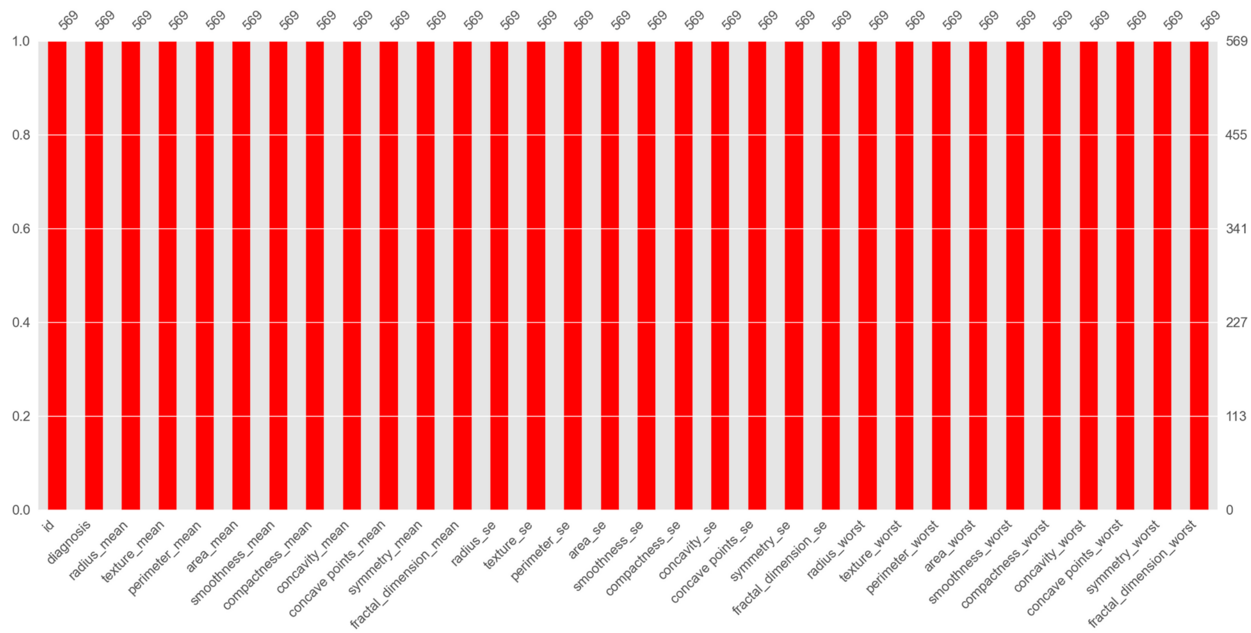
```
#Checking missing values
data.isna().sum()
```

```
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
```

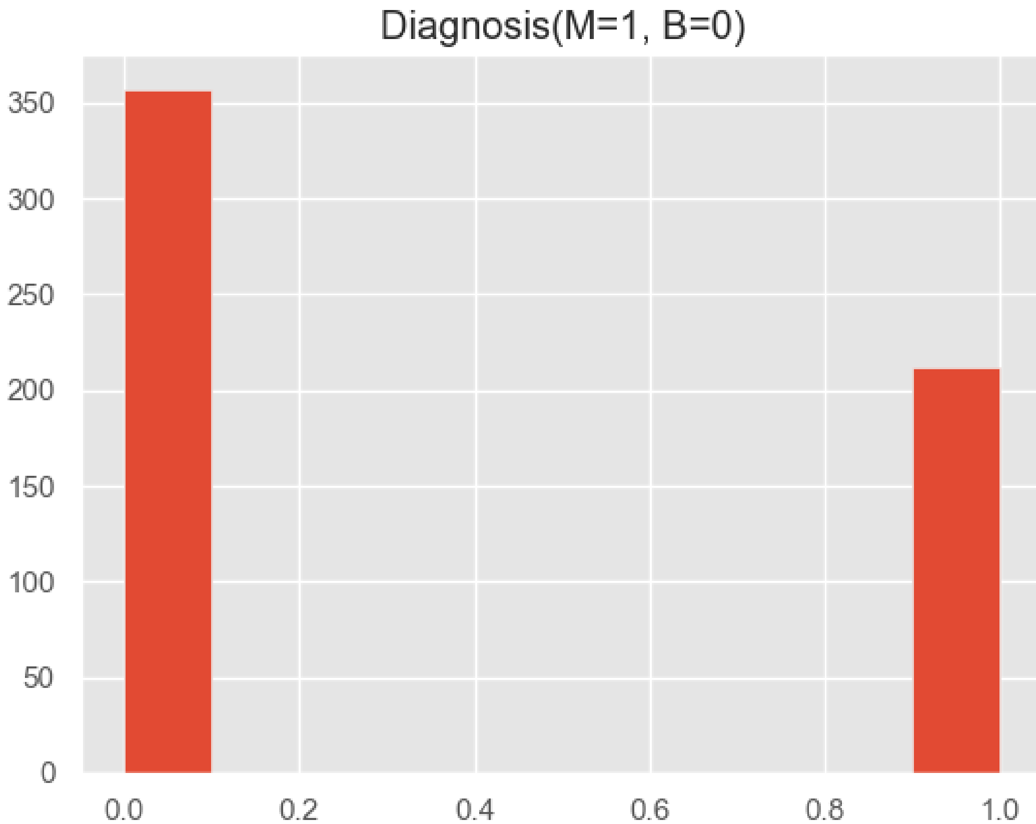
```
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

```
msno.bar(data,color='red')
```

```
<Axes: >
```

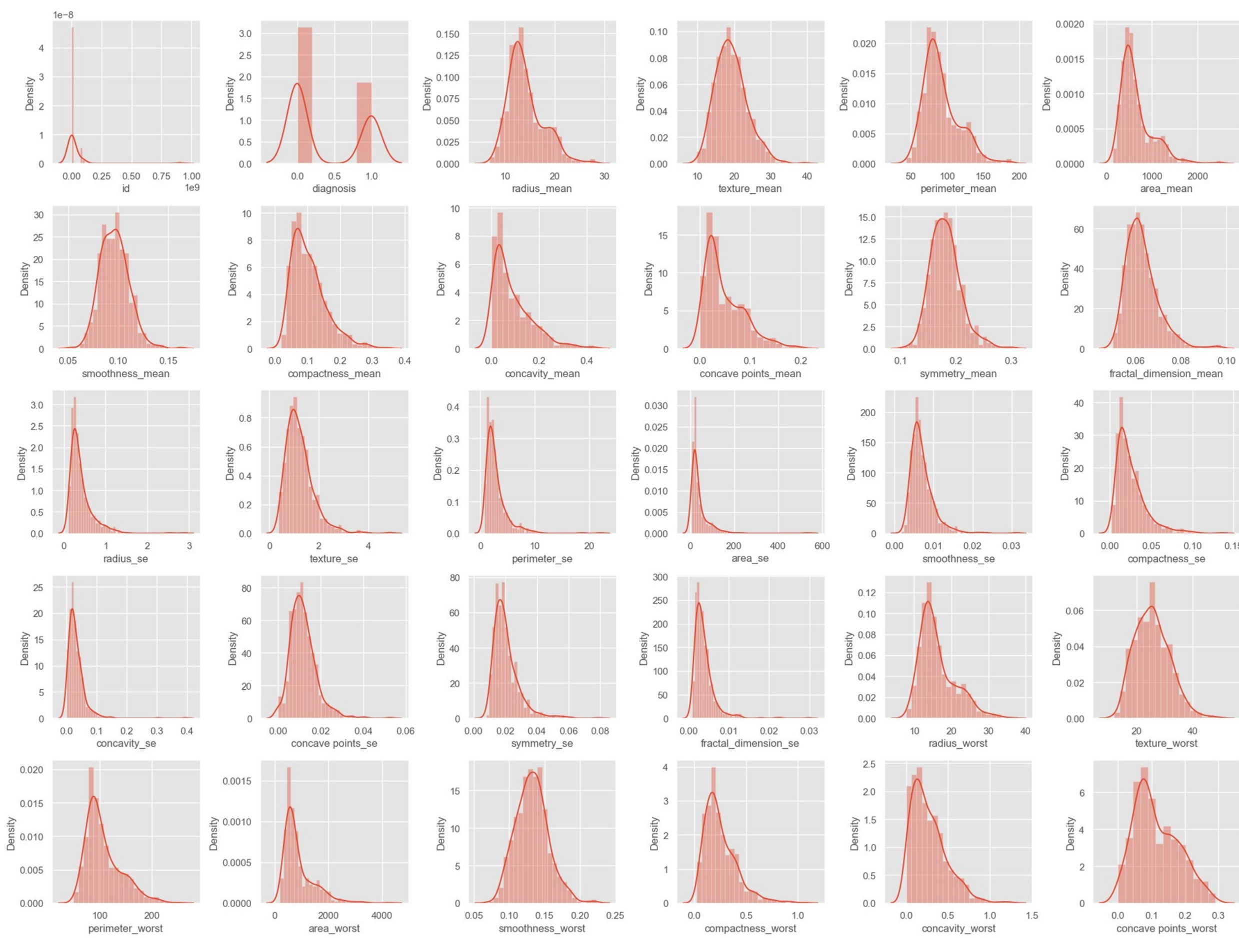


```
#There is no missing values in the dataset


data['diagnosis'] = data['diagnosis'].apply(lambda val:1 if val=='M'
else 0)


plt.hist(data['diagnosis'])
plt.title('Diagnosis(M=1, B=0)')
plt.show()
```



EDA

```
# each 5 row its having 6 columns
# density graph

plt.figure(figsize=(20,15))
plotnumber=1
for column in data:
    if plotnumber<=30:
     ax = plt.subplot(5,6,plotnumber)
     sns.distplot(data[column])
     plt.xlabel(column)
     plotnumber+=1

plt.tight_layout()
plt.show()
```



General Observations Density Peaks:

Each histogram has a peak where most data points are concentrated. The shape and spread of the distribution provide insights into the nature of the data. Skewness:

Many features exhibit skewness (either left or right), indicating that data points are not symmetrically distributed around the mean. Outliers:

Some features show long tails, suggesting the presence of outliers. Specific Features and Insights First Set of Features ID:

Distribution is skewed towards the left with a sharp peak, likely because IDs are unique identifiers rather than informative features. Diagnosis:

This appears to be a binary feature (0 or 1), possibly representing benign (0) and malignant (1) cases. The distribution shows a higher density around 0, suggesting more benign cases in the dataset. Radius Mean:

The distribution is slightly right-skewed with a peak around 12-15. This indicates most tumors have a radius in this range. Texture Mean:

Also right-skewed, with a peak around 15-20. Most tumors have this texture range. Perimeter Mean:

Similar to radius mean, this feature shows a right-skewed distribution with most values around 75-100. Area Mean:

This feature is heavily right-skewed with a peak around 500-750, indicating most tumors fall in this area range. Smoothness Mean:

More normally distributed with a peak around 0.1, indicating a typical range for this feature. Compactness Mean:

Shows a peak around 0.1-0.15, with a right-skewed distribution. Concavity Mean:

Right-skewed with a peak around 0.05-0.1. Concave Points Mean:

Right-skewed with a peak around 0.05-0.1, similar to concavity mean. Symmetry Mean:

Normally distributed with a peak around 0.2, suggesting most tumors have this symmetry range. Fractal Dimension Mean:

Right-skewed with a peak around 0.05-0.06. Second Set of Features Radius SE (Standard Error):

Heavily right-skewed with a sharp peak close to 0, indicating most standard errors are very small. Texture SE:

Similar to radius SE, right-skewed with most values close to 0. Perimeter SE:

Right-skewed with most values close to 0, suggesting small variations in perimeter measurements. Area SE:

Heavily right-skewed with a long tail, indicating some large standard errors but most values close to 0. Smoothness SE:

Similar pattern with most values close to 0. Compactness SE:

Right-skewed with a peak around 0.02. Concavity SE:

Right-skewed with a peak around 0.02. Concave Points SE:

Right-skewed with most values close to 0. Symmetry SE:

Right-skewed with a peak around 0.02-0.03. Fractal Dimension SE:

Right-skewed with a peak around 0.002. Third Set of Features Radius Worst:

Right-skewed with a peak around 20-25, indicating worst-case radius measurements. Texture Worst:

Right-skewed with a peak around 20-30. Perimeter Worst:

Right-skewed with most values around 100-150. Area Worst:

Heavily right-skewed with a peak around 1000-2000. Smoothness Worst:

Normally distributed with a peak around 0.15-0.2. Compactness Worst:

Right-skewed with a peak around 0.2-0.3. Concavity Worst:

Right-skewed with a peak around 0.2-0.5. Concave Points Worst:

Right-skewed with a peak around 0.1-0.2. Symmetry Worst:

Normally distributed with a peak around 0.3. Fractal Dimension Worst:

Right-skewed with a peak around 0.08. Insights and Implications Data Skewness:

Many features are right-skewed, suggesting the need for normalization or transformation before applying certain machine learning algorithms. Feature Importance:

Features like radius_mean, texture_mean, and area_mean show distinct distributions which could be important for diagnostic models. Data Preprocessing:

The presence of outliers and skewness implies that robust preprocessing techniques, such as scaling and outlier treatment, are essential. Model Building:

Understanding the distribution helps in selecting appropriate algorithms and handling features effectively during model training.

```
data.corr()

                               id  diagnosis  radius_mean
texture_mean  \
id                       1.000000   0.039769     0.074626
0.099770
diagnosis                0.039769   1.000000     0.730029
0.415185
radius_mean              0.074626   0.730029     1.000000
0.323782
texture_mean             0.099770   0.415185     0.323782
1.000000
perimeter_mean           0.073159   0.742636     0.997855
0.329533
area_mean                0.096893   0.708984     0.987357
0.321086
smoothness_mean         -0.012968   0.358560     0.170581     -
0.023389
compactness_mean         0.000096   0.596534     0.506124
0.236702
concavity_mean           0.050080   0.696360     0.676764
0.302418
concave points_mean      0.044158   0.776614     0.822529
0.293464
symmetry_mean           -0.022114   0.330499     0.147741
0.071401
fractal_dimension_mean  -0.052511  -0.012838    -0.311631     -
0.076437
radius_se                0.143048   0.567134     0.679090
0.275869
texture_se              -0.007526  -0.008303    -0.097317
0.386358
```

```
perimeter_se              0.137331   0.556141     0.674172
0.281673
area_se                   0.177742   0.548236     0.735864
0.259845
smoothness_se             0.096781  -0.067016    -0.222600
0.006614
compactness_se            0.033961   0.292999     0.206000
0.191975
concavity_se              0.055239   0.253730     0.194204
0.143293
concave points_se         0.078768   0.408042     0.376169
0.163851
symmetry_se              -0.017306  -0.006522    -0.104321
0.009127
fractal_dimension_se      0.025725   0.077972    -0.042641
0.054458
radius_worst              0.082405   0.776454     0.969539
0.352573
texture_worst             0.064720   0.456903     0.297008
0.912045
perimeter_worst           0.079986   0.782914     0.965137
0.358040
area_worst                0.107187   0.733825     0.941082
0.343546
smoothness_worst          0.010338   0.421465     0.119616
0.077503
compactness_worst        -0.002968   0.590998     0.413463
0.277830
concavity_worst           0.023203   0.659610     0.526911
0.301025
concave points_worst      0.035174   0.793566     0.744214
0.295316
symmetry_worst           -0.044224   0.416294     0.163953
0.105008
fractal_dimension_worst -0.029866   0.323872     0.007066
0.119205

                         perimeter_mean  area_mean  smoothness_mean  \
id                             0.073159   0.096893        -0.012968
diagnosis                      0.742636   0.708984         0.358560
radius_mean                    0.997855   0.987357         0.170581
texture_mean                   0.329533   0.321086        -0.023389
perimeter_mean                 1.000000   0.986507         0.207278
area_mean                      0.986507   1.000000         0.177028
smoothness_mean                0.207278   0.177028         1.000000
compactness_mean               0.556936   0.498502         0.659123
concavity_mean                 0.716136   0.685983         0.521984
concave points_mean            0.850977   0.823269         0.553695
symmetry_mean                  0.183027   0.151293         0.557775
```

fractal_dimension_mean	-0.261477	-0.283110	0.584792
radius_se	0.691765	0.732562	0.301467
texture_se	-0.086761	-0.066280	0.068406
perimeter_se	0.693135	0.726628	0.296092
area_se	0.744983	0.800086	0.246552
smoothness_se	-0.202694	-0.166777	0.332375
compactness_se	0.250744	0.212583	0.318943
concavity_se	0.228082	0.207660	0.248396
concave points_se	0.407217	0.372320	0.380676
symmetry_se	-0.081629	-0.072497	0.200774
fractal_dimension_se	-0.005523	-0.019887	0.283607
radius_worst	0.969476	0.962746	0.213120
texture_worst	0.303038	0.287489	0.036072
perimeter_worst	0.970387	0.959120	0.238853
area_worst	0.941550	0.959213	0.206718
smoothness_worst	0.150549	0.123523	0.805324
compactness_worst	0.455774	0.390410	0.472468
concavity_worst	0.563879	0.512606	0.434926
concave points_worst	0.771241	0.722017	0.503053
symmetry_worst	0.189115	0.143570	0.394309
fractal_dimension_worst	0.051019	0.003738	0.499316

	compactness_mean	concavity_mean \
id	0.000096	0.050080
diagnosis	0.596534	0.696360
radius_mean	0.506124	0.676764
texture_mean	0.236702	0.302418
perimeter_mean	0.556936	0.716136
area_mean	0.498502	0.685983
smoothness_mean	0.659123	0.521984
compactness_mean	1.000000	0.883121
concavity_mean	0.883121	1.000000
concave points_mean	0.831135	0.921391
symmetry_mean	0.602641	0.500667
fractal_dimension_mean	0.565369	0.336783
radius_se	0.497473	0.631925
texture_se	0.046205	0.076218
perimeter_se	0.548905	0.660391
area_se	0.455653	0.617427
smoothness_se	0.135299	0.098564
compactness_se	0.738722	0.670279
concavity_se	0.570517	0.691270
concave points_se	0.642262	0.683260
symmetry_se	0.229977	0.178009
fractal_dimension_se	0.507318	0.449301
radius_worst	0.535315	0.688236
texture_worst	0.248133	0.299879
perimeter_worst	0.590210	0.729565
area_worst	0.509604	0.675987

```
smoothness_worst                 0.565541        0.448822
compactness_worst                0.865809        0.754968
concavity_worst                  0.816275        0.884103
concave points_worst             0.815573        0.861323
symmetry_worst                   0.510223        0.409464
fractal_dimension_worst          0.687382        0.514930

                         concave points_mean  ...  radius_worst  \
id                                  0.044158  ...      0.082405
diagnosis                           0.776614  ...      0.776454
radius_mean                         0.822529  ...      0.969539
texture_mean                        0.293464  ...      0.352573
perimeter_mean                      0.850977  ...      0.969476
area_mean                           0.823269  ...      0.962746
smoothness_mean                     0.553695  ...      0.213120
compactness_mean                    0.831135  ...      0.535315
concavity_mean                      0.921391  ...      0.688236
concave points_mean                 1.000000  ...      0.830318
symmetry_mean                       0.462497  ...      0.185728
fractal_dimension_mean              0.166917  ...     -0.253691
radius_se                           0.698050  ...      0.715065
texture_se                          0.021480  ...     -0.111690
perimeter_se                        0.710650  ...      0.697201
area_se                             0.690299  ...      0.757373
smoothness_se                       0.027653  ...     -0.230691
compactness_se                      0.490424  ...      0.204607
concavity_se                        0.439167  ...      0.186904
concave points_se                   0.615634  ...      0.358127
symmetry_se                         0.095351  ...     -0.128121
fractal_dimension_se                0.257584  ...     -0.037488
radius_worst                        0.830318  ...      1.000000
texture_worst                       0.292752  ...      0.359921
perimeter_worst                     0.855923  ...      0.993708
area_worst                          0.809630  ...      0.984015
smoothness_worst                    0.452753  ...      0.216574
compactness_worst                   0.667454  ...      0.475820
concavity_worst                     0.752399  ...      0.573975
concave points_worst                0.910155  ...      0.787424
symmetry_worst                      0.375744  ...      0.243529
fractal_dimension_worst             0.368661  ...      0.093492

                         texture_worst  perimeter_worst  area_worst  \
id                            0.064720         0.079986    0.107187
diagnosis                     0.456903         0.782914    0.733825
radius_mean                   0.297008         0.965137    0.941082
texture_mean                  0.912045         0.358040    0.343546
perimeter_mean                0.303038         0.970387    0.941550
area_mean                     0.287489         0.959120    0.959213
smoothness_mean               0.036072         0.238853    0.206718
```

```
compactness_mean           0.248133        0.590210    0.509604
concavity_mean             0.299879        0.729565    0.675987
concave points_mean        0.292752        0.855923    0.809630
symmetry_mean              0.090651        0.219169    0.177193
fractal_dimension_mean    -0.051269       -0.205151   -0.231854
radius_se                  0.194799        0.719684    0.751548
texture_se                 0.409003       -0.102242   -0.083195
perimeter_se               0.200371        0.721031    0.730713
area_se                    0.196497        0.761213    0.811408
smoothness_se             -0.074743       -0.217304   -0.182195
compactness_se             0.143003        0.260516    0.199371
concavity_se               0.100241        0.226680    0.188353
concave points_se          0.086741        0.394999    0.342271
symmetry_se               -0.077473       -0.103753   -0.110343
fractal_dimension_se      -0.003195       -0.001000   -0.022736
radius_worst               0.359921        0.993708    0.984015
texture_worst              1.000000        0.365098    0.345842
perimeter_worst            0.365098        1.000000    0.977578
area_worst                 0.345842        0.977578    1.000000
smoothness_worst           0.225429        0.236775    0.209145
compactness_worst          0.360832        0.529408    0.438296
concavity_worst            0.368366        0.618344    0.543331
concave points_worst       0.359755        0.816322    0.747419
symmetry_worst             0.233027        0.269493    0.209146
fractal_dimension_worst    0.219122        0.138957    0.079647

                         smoothness_worst  compactness_worst
concavity_worst  \
id                               0.010338          -0.002968
0.023203
diagnosis                        0.421465           0.590998
0.659610
radius_mean                      0.119616           0.413463
0.526911
texture_mean                     0.077503           0.277830
0.301025
perimeter_mean                   0.150549           0.455774
0.563879
area_mean                        0.123523           0.390410
0.512606
smoothness_mean                  0.805324           0.472468
0.434926
compactness_mean                 0.565541           0.865809
0.816275
concavity_mean                   0.448822           0.754968
0.884103
concave points_mean              0.452753           0.667454
0.752399
symmetry_mean                    0.426675           0.473200
```

```
0.433721
fractal_dimension_mean           0.504942          0.458798
0.346234
radius_se                        0.141919          0.287103
0.380585
texture_se                      -0.073658         -0.092439         -
0.068956
perimeter_se                     0.130054          0.341919
0.418899
area_se                          0.125389          0.283257
0.385100
smoothness_se                    0.314457         -0.055558         -
0.058298
compactness_se                   0.227394          0.678780
0.639147
concavity_se                     0.168481          0.484858
0.662564
concave points_se                0.215351          0.452888
0.549592
symmetry_se                     -0.012662          0.060255
0.037119
fractal_dimension_se             0.170568          0.390159
0.379975
radius_worst                     0.216574          0.475820
0.573975
texture_worst                    0.225429          0.360832
0.368366
perimeter_worst                  0.236775          0.529408
0.618344
area_worst                       0.209145          0.438296
0.543331
smoothness_worst                 1.000000          0.568187
0.518523
compactness_worst                0.568187          1.000000
0.892261
concavity_worst                  0.518523          0.892261
1.000000
concave points_worst             0.547691          0.801080
0.855434
symmetry_worst                   0.493838          0.614441
0.532520
fractal_dimension_worst          0.617624          0.810455
0.686511

                         concave points_worst  symmetry_worst  \
id                                   0.035174       -0.044224
diagnosis                            0.793566        0.416294
radius_mean                          0.744214        0.163953
texture_mean                         0.295316        0.105008
```

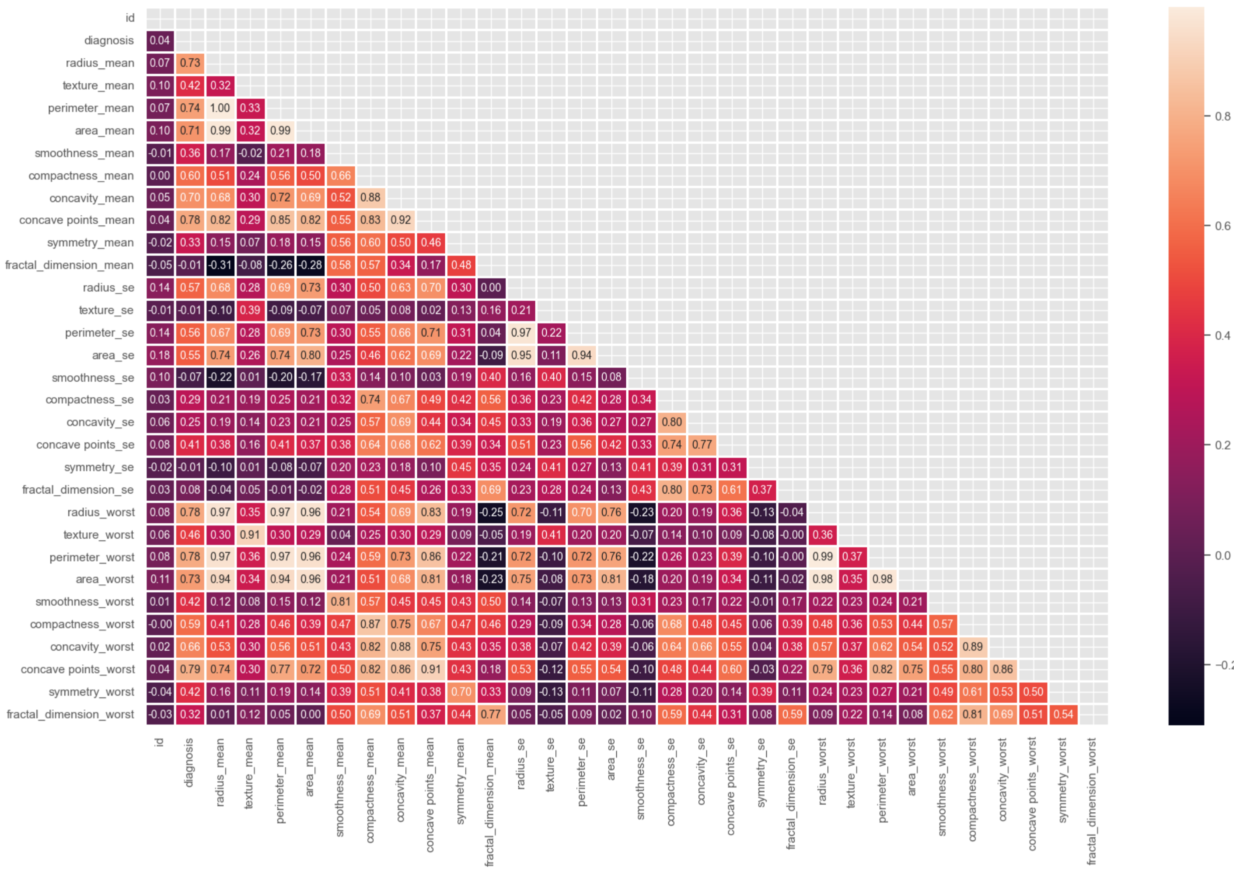
perimeter_mean	0.771241	0.189115
area_mean	0.722017	0.143570
smoothness_mean	0.503053	0.394309
compactness_mean	0.815573	0.510223
concavity_mean	0.861323	0.409464
concave points_mean	0.910155	0.375744
symmetry_mean	0.430297	0.699826
fractal_dimension_mean	0.175325	0.334019
radius_se	0.531062	0.094543
texture_se	-0.119638	-0.128215
perimeter_se	0.554897	0.109930
area_se	0.538166	0.074126
smoothness_se	-0.102007	-0.107342
compactness_se	0.483208	0.277878
concavity_se	0.440472	0.197788
concave points_se	0.602450	0.143116
symmetry_se	-0.030413	0.389402
fractal_dimension_se	0.215204	0.111094
radius_worst	0.787424	0.243529
texture_worst	0.359755	0.233027
perimeter_worst	0.816322	0.269493
area_worst	0.747419	0.209146
smoothness_worst	0.547691	0.493838
compactness_worst	0.801080	0.614441
concavity_worst	0.855434	0.532520
concave points_worst	1.000000	0.502528
symmetry_worst	0.502528	1.000000
fractal_dimension_worst	0.511114	0.537848

	fractal_dimension_worst
id	-0.029866
diagnosis	0.323872
radius_mean	0.007066
texture_mean	0.119205
perimeter_mean	0.051019
area_mean	0.003738
smoothness_mean	0.499316
compactness_mean	0.687382
concavity_mean	0.514930
concave points_mean	0.368661
symmetry_mean	0.438413
fractal_dimension_mean	0.767297
radius_se	0.049559
texture_se	-0.045655
perimeter_se	0.085433
area_se	0.017539
smoothness_se	0.101480
compactness_se	0.590973
concavity_se	0.439329

```
concave points_se                                  0.310655
symmetry_se                                        0.078079
fractal_dimension_se                               0.591328
radius_worst                                       0.093492
texture_worst                                      0.219122
perimeter_worst                                    0.138957
area_worst                                         0.079647
smoothness_worst                                   0.617624
compactness_worst                                  0.810455
concavity_worst                                    0.686511
concave points_worst                               0.511114
symmetry_worst                                     0.537848
fractal_dimension_worst                            1.000000

[32 rows x 32 columns]
```

Key Relationships Strong Positive Correlations Diagnosis and Perimeter Mean (0.742636): If the perimeter mean of a tumor is high, it's likely that the diagnosis is malignant (cancerous). Radius Mean and Perimeter Mean (0.997855): Tumors with larger radii tend to have larger perimeters. Radius Mean and Area Mean (0.987357): Larger radius usually means a larger tumor area. Perimeter Mean and Area Mean (0.986507): Tumors with a larger perimeter often have a larger area. Concavity Mean and Concave Points Mean (0.921391): If the tumor is more concave, it has more concave points. Moderate Positive Correlations Compactness Mean and Concave Points Mean (0.831135): More compact tumors tend to have more concave points. Concavity Mean and Compactness Mean (0.883121): Tumors with higher concavity also tend to be more compact. Diagnosis and Concave Points Mean (0.776614): Tumors with more concave points are likely to be malignant. Radius Worst and Perimeter Worst (0.993708): Tumors with the largest radius in their worst state have the largest perimeter in their worst state. Weak or No Correlation Diagnosis and Texture Mean (0.415185): There's some relationship, but not strong. Texture mean doesn’t strongly predict malignancy. Smoothness Mean and Texture Mean (-0.023389): The smoothness of a tumor doesn't significantly relate to its texture. Fractal Dimension Mean and Perimeter Mean (-0.311631): Little to no relationship between these measures. Diagnosis and Smoothness Worst (0.421465): Slight relationship, meaning smoothness in the worst case has a small predictive value for diagnosis. Negative Correlations Fractal Dimension Mean and Radius Mean (-0.311631): As the fractal dimension increases, the radius mean tends to decrease slightly. Symmetry Worst and Perimeter Mean (0.189115): Minimal relationship; higher symmetry doesn’t necessarily mean a larger perimeter. Understanding Patterns Size-Related Features: Features like radius, perimeter, and area are highly correlated with each other. Larger tumors have larger measures in these aspects. Shape-Related Features: Features like concavity, concave points, and compactness are also highly correlated. Tumors that are more concave and compact tend to have more concave points. Conclusion The correlation matrix helps identify which features of a tumor are most closely related to malignancy and to each other. This understanding can be crucial for developing diagnostic models and understanding the characteristics of cancerous tumors. High positive correlations (close to 1) indicate that as one feature increases, another feature also increases, and this relationship is strong. Negative correlations (close to -1) indicate that as one feature increases, the other decreases, and this relationship is also strong but in the opposite direction.

```python
plt.figure(figsize=(20,12))
corr=data.corr()
mask = np.triu(np.ones_like(corr, dtype=bool))
sns.heatmap(corr, mask=mask, linewidths=1, annot=True, fmt = ".2f")
plt.show()
```



Specific Insights Diagnosis:

The feature diagnosis has high correlations with certain features like radius_mean (0.73), indicating that larger radii might be associated with a particular diagnosis (likely malignancy). Other features such as concave points_mean also show high correlation with diagnosis (0.78). Feature Relationships:

area_mean, perimeter_mean, and radius_mean are highly correlated, which makes sense since larger perimeters and radii usually result in larger areas.

```python
# highly correlated feature
# multicollinearity

data.drop('id', axis=1, inplace=True)

# feature selection
corr_matrix = data.corr().abs()
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
```

```
tri_df = corr_matrix.mask(mask)

to_drop = [x for x in tri_df.columns if any(tri_df[x]>0.92)]

data = data.drop(to_drop, axis=1)

print(data.shape[1])
```

```
23
```

```
data.head()
```

```
   diagnosis  texture_mean  smoothness_mean  compactness_mean  \
0          1         10.38          0.11840           0.27760
1          1         17.77          0.08474           0.07864
2          1         21.25          0.10960           0.15990
3          1         20.38          0.14250           0.28390
4          1         14.34          0.10030           0.13280

   concave points_mean  symmetry_mean  fractal_dimension_mean
texture_se  \
0              0.14710         0.2419                 0.07871
0.9053
1              0.07017         0.1812                 0.05667
0.7339
2              0.12790         0.2069                 0.05999
0.7869
3              0.10520         0.2597                 0.09744
1.1560
4              0.10430         0.1809                 0.05883
0.7813

   area_se  smoothness_se  ...  symmetry_se  fractal_dimension_se  \
0   153.40       0.006399  ...      0.03003              0.006193
1    74.08       0.005225  ...      0.01389              0.003532
2    94.03       0.006150  ...      0.02250              0.004571
3    27.23       0.009110  ...      0.05963              0.009208
4    94.44       0.011490  ...      0.01756              0.005115

   texture_worst  area_worst  smoothness_worst  compactness_worst  \
0          17.33      2019.0            0.1622             0.6656
1          23.41      1956.0            0.1238             0.1866
2          25.53      1709.0            0.1444             0.4245
3          26.50       567.7            0.2098             0.8663
4          16.67      1575.0            0.1374             0.2050

   concavity_worst  concave points_worst  symmetry_worst  \
0           0.7119                0.2654          0.4601
1           0.2416                0.1860          0.2750
2           0.4504                0.2430          0.3613
3           0.6869                0.2575          0.6638
```

```
4          0.4000                0.1625       0.2364

   fractal_dimension_worst
0                  0.11890
1                  0.08902
2                  0.08758
3                  0.17300
4                  0.07678

[5 rows x 23 columns]
```

```
# 32 feature reduce it 23 now

X=data.drop('diagnosis', axis=1)
y=data['diagnosis']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train ,y_test =train_test_split(X,y, test_size=0.2,
random_state=0)

#scalar data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

X_train.shape
```

```
(455, 22)
```

```
# Apply Machine learning Algo

from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(X_train, y_train)
```

```
LogisticRegression()
```

```
y_pred = log_reg.predict(X_test)

y_pred
```

```
array([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, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
```

```
0,
       0, 1, 1, 0], dtype=int64)
```

```python
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print(accuracy_score(y_train, log_reg.predict(X_train)))
log_reg_acc = accuracy_score(y_test, log_reg.predict(X_test))
print(log_reg_acc)
y_pred = log_reg.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.989010989010989
0.9649122807017544
[[66  1]
 [ 3 44]]
              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
```

```python
# KNN
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier()
```

```python
y_pred = knn.predict(X_test)
```

```python
y_pred
```

```
array([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, 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, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
0,
       0, 1, 1, 0], dtype=int64)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print(accuracy_score(y_train, knn.predict(X_train)))
knn_acc = accuracy_score(y_test, knn.predict(X_test))
print(knn_acc)
y_pred = knn.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.967032967032967
0.956140350877193
[[66  1]
 [ 4 43]]
              precision    recall  f1-score   support

           0       0.94      0.99      0.96        67
           1       0.98      0.91      0.95        47

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

```
# SVC
#Hyperparameter tuning
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
svc= SVC(probability=True)

parameters = {
    'gamma': [0.0001, 0.001, 0.01, 0.1],
    'C':[0.01, 0.05, 0.5, 0.1, 1,10, 15,20]
}
grid_search = GridSearchCV(svc, parameters)
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(estimator=SVC(probability=True),
             param_grid={'C': [0.01, 0.05, 0.5, 0.1, 1, 10, 15, 20],
                         'gamma': [0.0001, 0.001, 0.01, 0.1]})
```

```
grid_search.best_params_
```

```
{'C': 15, 'gamma': 0.01}
```

```
grid_search.best_score_
```

```
0.9802197802197803
```

```
svc = SVC(C=15, gamma=0.01, probability=True)
svc.fit(X_train, y_train)
```

```
SVC(C=15, gamma=0.01, probability=True)
```

```
y_pred = svc.predict(X_test)

y_pred

array([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, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,
1,
       0, 1, 1, 0], dtype=int64)

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print(accuracy_score(y_train, svc.predict(X_train)))
svc_acc = accuracy_score(y_test, svc.predict(X_test))
print(svc_acc)
y_pred = svc.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

0.989010989010989
0.9824561403508771
[[67  0]
 [ 2 45]]
              precision    recall  f1-score   support

           0       0.97      1.00      0.99        67
           1       1.00      0.96      0.98        47

    accuracy                           0.98       114
   macro avg       0.99      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114


# DT
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

parameters = {
    'criterion':['gini','entropy'],
    'max_depth':range(2,32,1),
    'min_samples_leaf':range(1,10,1),
    'min_samples_split':range(2,10,1),
    'splitter':['best','random']
}
```

```
grid_search_dt = GridSearchCV(dtc, parameters, cv=5, n_jobs=-1,
verbose=1)
grid_search_dt.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 8640 candidates, totalling 43200 fits
```

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(), n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': range(2, 32),
                         'min_samples_leaf': range(1, 10),
                         'min_samples_split': range(2, 10),
                         'splitter': ['best', 'random']},
             verbose=1)
```

```
grid_search_dt.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 12,
 'min_samples_leaf': 3,
 'min_samples_split': 2,
 'splitter': 'random'}
```

```
grid_search_dt.best_score_
```

```
0.9626373626373628
```

```
dtc = DecisionTreeClassifier(criterion='entropy', max_depth=15,
min_samples_leaf=4, min_samples_split=5, splitter = 'random')

dtc.fit(X_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=15,
min_samples_leaf=4,
                       min_samples_split=5, splitter='random')
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print(accuracy_score(y_train, dtc.predict(X_train)))
dtc_acc = accuracy_score(y_test, dtc.predict(X_test))
print(dtc_acc)
y_pred = dtc.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.967032967032967
0.9473684210526315
[[65  2]
 [ 4 43]]
              precision    recall  f1-score   support

           0       0.94      0.97      0.96        67
```

```
           1       0.96      0.91      0.93        47

    accuracy                           0.95       114
   macro avg       0.95      0.94      0.95       114
weighted avg       0.95      0.95      0.95       114
```

```
from sklearn.ensemble import RandomForestClassifier

rand_clf = RandomForestClassifier(criterion = 'entropy', max_depth =
10, max_features = 0.5, min_samples_leaf = 2, min_samples_split = 3,
n_estimators = 130)
rand_clf.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', max_depth=10,
max_features=0.5,
                       min_samples_leaf=2, min_samples_split=3,
                       n_estimators=130)
```

```
y_pred = rand_clf.predict(X_test)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print(accuracy_score(y_train, rand_clf.predict(X_train)))
rand_clf_acc = accuracy_score(y_test, rand_clf.predict(X_test))
print(rand_clf_acc)
y_pred = rand_clf.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.9978021978021978
0.9824561403508771
[[66  1]
 [ 1 46]]
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        67
           1       0.98      0.98      0.98        47

    accuracy                           0.98       114
   macro avg       0.98      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

```
from sklearn.ensemble import GradientBoostingClassifier

gbc = GradientBoostingClassifier()

parameters = {
    'loss': ['deviance', 'exponential'],
    'learning_rate': [0.001, 0.1],
```

```
    'n_estimators': [100, 150, 180]
}

grid_search_gbc = GridSearchCV(gbc, parameters, cv = 2, n_jobs = -5,
verbose = 1)
grid_search_gbc.fit(X_train, y_train)

Fitting 2 folds for each of 12 candidates, totalling 24 fits

GridSearchCV(cv=2, estimator=GradientBoostingClassifier(), n_jobs=-5,
             param_grid={'learning_rate': [0.001, 0.1],
                         'loss': ['deviance', 'exponential'],
                         'n_estimators': [100, 150, 180]},
             verbose=1)

grid_search_gbc.best_params_

{'learning_rate': 0.1, 'loss': 'exponential', 'n_estimators': 180}

grid_search_gbc.best_score_

0.9582850297550043

gbc = GradientBoostingClassifier(learning_rate = 0.1, loss =
'exponential', n_estimators = 180)
gbc.fit(X_train, y_train)

GradientBoostingClassifier(loss='exponential', n_estimators=180)

from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print(accuracy_score(y_train, gbc.predict(X_train)))
gbc_acc = accuracy_score(y_test, gbc.predict(X_test))
print(gbc_acc)
y_pred = gbc.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

1.0
0.9649122807017544
[[64  3]
 [ 1 46]]
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        67
           1       0.94      0.98      0.96        47

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

```python
from xgboost import XGBClassifier

xgb = XGBClassifier(objective = 'binary:logistic', learning_rate =
0.01, max_depth = 5, n_estimators = 180)

xgb.fit(X_train, y_train)
```

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None,
early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None,
feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.01,
max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=5, max_leaves=None,
              min_child_weight=None, missing=nan,
monotone_constraints=None,
              multi_strategy=None, n_estimators=180, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```python
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
print(accuracy_score(y_train, xgb.predict(X_train)))
xgb_acc = accuracy_score(y_test, xgb.predict(X_test))
print(xgb_acc)
y_pred = xgb.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.9934065934065934
0.956140350877193
[[65  2]
 [ 3 44]]
              precision    recall  f1-score   support

           0       0.96      0.97      0.96        67
           1       0.96      0.94      0.95        47

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

```python
models = pd.DataFrame({
    'Model': ['Logistic Regression', 'KNN', 'SVM', 'Decision Tree
Classifier', 'Random Forest Classifier', 'Gradient Boosting
Classifier', 'XgBoost'],
    'Score': [100*round(log_reg_acc,4), 100*round(knn_acc,4),
```

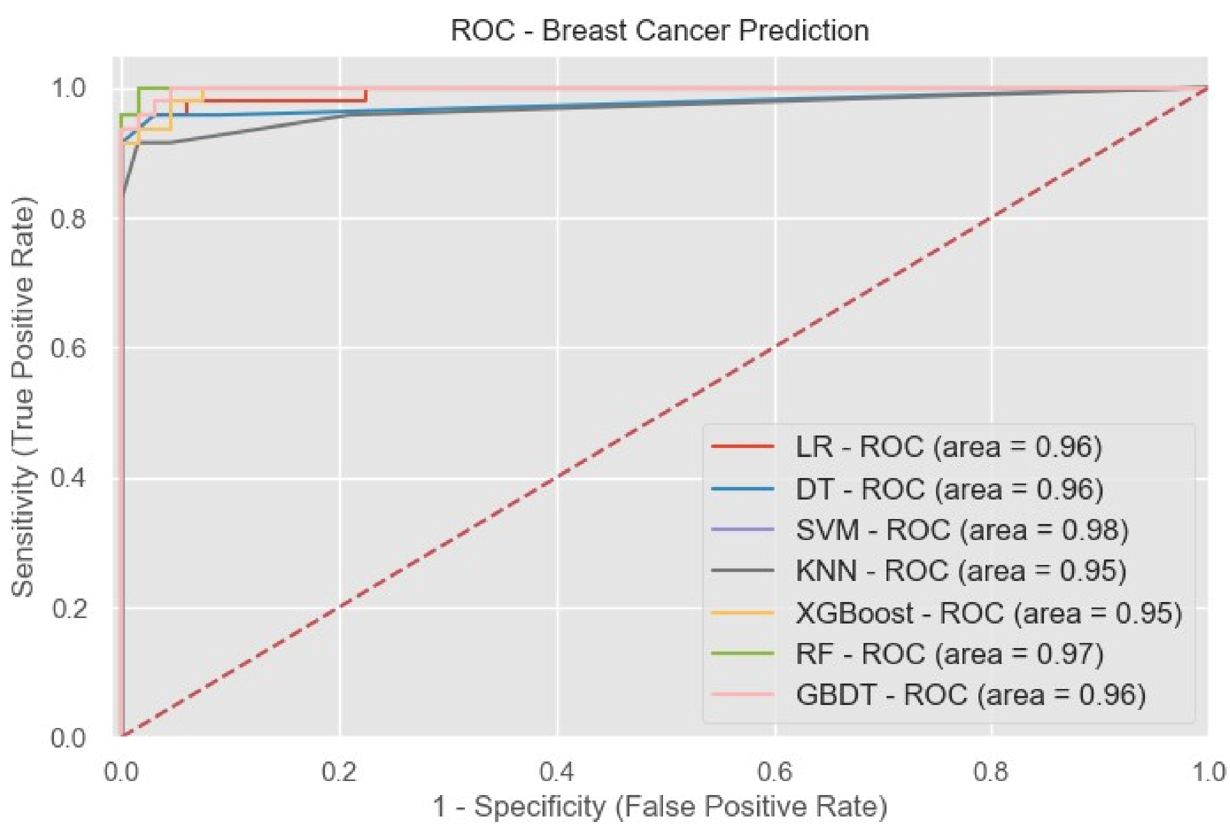
```
100*round(svc_acc,4), 100*round(dtc_acc,4), 100*round(rand_clf_acc,4),

              100*round(gbc_acc,4), 100*round(xgb_acc,4)]
})
models.sort_values(by = 'Score', ascending = False)
```

```
                          Model  Score
2                           SVM  98.25
4      Random Forest Classifier  98.25
0           Logistic Regression  96.49
5  Gradient Boosting Classifier  96.49
1                           KNN  95.61
6                       XgBoost  95.61
3      Decision Tree Classifier  94.74
```

```
import pickle
model = svc
pickle.dump(model, open("brest_cancer.pkl","wb"))
```

```
from sklearn import metrics
plt.figure(figsize=(8,5))
models = [
{
    'label': 'LR',
    'model': log_reg,
},
{
    'label': 'DT',
    'model': dtc,
},
{
    'label': 'SVM',
    'model': svc,
},
{
    'label': 'KNN',
    'model': knn,
},
{
    'label': 'XGBoost',
    'model': xgb,
},
{
    'label': 'RF',
    'model': rand_clf,
},
{
    'label': 'GBDT',
    'model': gbc,
}
```

```
]
for m in models:
    model = m['model']
    model.fit(X_train, y_train)
    y_pred=model.predict(X_test)
    fpr1, tpr1, thresholds = metrics.roc_curve(y_test, 
model.predict_proba(X_test)[:,1])
    auc = metrics.roc_auc_score(y_test,model.predict(X_test))
    plt.plot(fpr1, tpr1, label='%s - ROC (area = %0.2f)' % 
(m['label'], auc))

plt.plot([0, 1], [0, 1],'r--')
plt.xlim([-0.01, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('1 - Specificity (False Positive Rate)', fontsize=12)
plt.ylabel('Sensitivity (True Positive Rate)', fontsize=12)
plt.title('ROC - Breast Cancer Prediction', fontsize=12)
plt.legend(loc="lower right", fontsize=12)
plt.savefig("roc_breast_cancer.jpeg", format='jpeg', dpi=400, 
bbox_inches='tight')
plt.show()
```



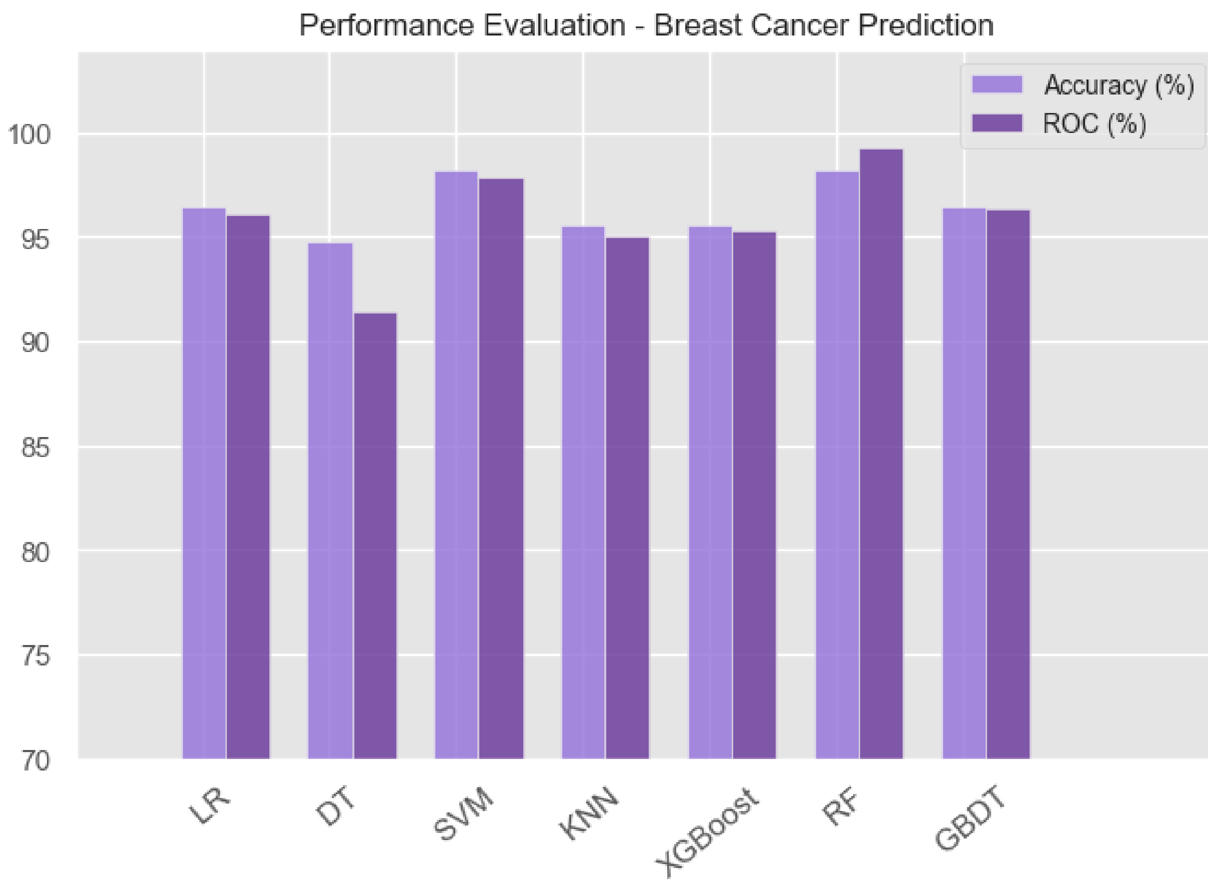
The ROC curve for Breast Cancer Prediction provides valuable insights into the performance of various classifiers:

High AUC Scores: The AUC scores for all models are impressive, with SVM leading at 0.98, indicating excellent model performance. Model Comparison: The ROC curve shows that SVM, RF, and GBDT have slightly better discriminative abilities than other models. Sensitivity and Specificity: The curve illustrates each model's trade-off between sensitivity and specificity, crucial for medical diagnosis. Clinical Decision Making: These insights can guide the selection of the most suitable model for clinical use, balancing the need for both high true positive rates and low false positives.

```
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
models = [
{
    'label': 'LR',
    'model': log_reg,
},
{
    'label': 'DT',
    'model': dtc,
},
{
    'label': 'SVM',
    'model': svc,
},
{
    'label': 'KNN',
    'model': knn,
},
{
    'label': 'XGBoost',
    'model': xgb,
},
{
    'label': 'RF',
    'model': rand_clf,
},
{
    'label': 'GBDT',
    'model': gbc,
}
]

means_roc = []
means_accuracy = [100*round(log_reg_acc,4), 100*round(dtc_acc,4),
100*round(svc_acc,4), 100*round(knn_acc,4), 100*round(xgb_acc,4),
                  100*round(rand_clf_acc,4), 100*round(gbc_acc,4)]

for m in models:
```

```
    model = m['model']
    model.fit(X_train, y_train)
    y_pred=model.predict(X_test)
    fpr1, tpr1, thresholds = metrics.roc_curve(y_test,
model.predict_proba(X_test)[:,1])
    auc = metrics.roc_auc_score(y_test,model.predict(X_test))
    auc = 100*round(auc,4)
    means_roc.append(auc)

print(means_accuracy)
print(means_roc)

# data to plot
n_groups = 7
means_accuracy = tuple(means_accuracy)
means_roc = tuple(means_roc)

# create plot
fig, ax = plt.subplots(figsize=(8,5))
index = np.arange(n_groups)
bar_width = 0.35
opacity = 0.8

rects1 = plt.bar(index, means_accuracy, bar_width,
alpha=opacity,
color='mediumpurple',
label='Accuracy (%)')

rects2 = plt.bar(index + bar_width, means_roc, bar_width,
alpha=opacity,
color='rebeccapurple',
label='ROC (%)')

plt.xlim([-1, 8])
plt.ylim([70, 104])

plt.title('Performance Evaluation - Breast Cancer Prediction',
fontsize=12)
plt.xticks(index, ('   LR', '   DT', '   SVM', '   KNN', 'XGBoost' , '
RF', '   GBDT'), rotation=40, ha='center', fontsize=12)
plt.legend(loc="upper right", fontsize=10)
plt.savefig("PE_breast_cancer.jpeg", format='jpeg', dpi=400,
bbox_inches='tight')
plt.show()
```

```
[96.49, 94.74000000000001, 98.25, 95.61, 95.61, 98.25, 96.49]
[96.06, 91.38, 97.87, 95.0, 95.32000000000001, 99.25, 96.38]
```



High Accuracy: All models, including Logistic Regression (LR), Decision Tree (DT), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), XGBoost, Random Forest (RF), and Gradient Boosting Decision Tree (GBDT), show high accuracy, mostly above 90%. ROC Performance: The ROC percentages are also high, indicating good model performance in distinguishing between the classes. Model Comparison: The bar chart suggests that some models may have a slight edge over others, but overall, they all perform well for this task. Clinical Application: The effectiveness of these models in predicting breast cancer is crucial for developing reliable diagnostic tools. These insights can help in selecting the most appropriate model for deployment in a clinical setting based on performance metric