breast cancer detectionn

August 28, 2021

1 Breast Cancer Detection

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
df = pd.read_csv("breast.csv")
```

[4]:
```
df
```

[4]:
```
           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
..        …       …          …           …             …         …
564    926424         M        21.56         22.39          142.00     1479.0
565    926682         M        20.13         28.25          131.20     1261.0
566    926954         M        16.60         28.08          108.30      858.1
567    927241         M        20.60         29.33          140.10     1265.0
568     92751         B         7.76         24.54           47.92      181.0

     smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0            0.11840           0.27760         0.30010              0.14710
1            0.08474           0.07864         0.08690              0.07017
2            0.10960           0.15990         0.19740              0.12790
3            0.14250           0.28390         0.24140              0.10520
4            0.10030           0.13280         0.19800              0.10430
..               …               …             …                  …
564          0.11100           0.11590         0.24390              0.13890
565          0.09780           0.10340         0.14400              0.09791
566          0.08455           0.10230         0.09251              0.05302
```

```
567         0.11780          0.27700        0.35140            0.15200
568         0.05263          0.04362        0.00000            0.00000

     …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0    …          17.33           184.60      2019.0           0.16220
1    …          23.41           158.80      1956.0           0.12380
2    …          25.53           152.50      1709.0           0.14440
3    …          26.50            98.87       567.7           0.20980
4    …          16.67           152.20      1575.0           0.13740
..   …            …                …          …                …
564  …          26.40           166.10      2027.0           0.14100
565  …          38.25           155.00      1731.0           0.11660
566  …          34.12           126.70      1124.0           0.11390
567  …          39.42           184.60      1821.0           0.16500
568  …          30.37            59.16       268.6           0.08996

     compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0              0.66560           0.7119                0.2654          0.4601
1              0.18660           0.2416                0.1860          0.2750
2              0.42450           0.4504                0.2430          0.3613
3              0.86630           0.6869                0.2575          0.6638
4              0.20500           0.4000                0.1625          0.2364
..                 …                …                     …               …
564            0.21130           0.4107                0.2216          0.2060
565            0.19220           0.3215                0.1628          0.2572
566            0.30940           0.3403                0.1418          0.2218
567            0.86810           0.9387                0.2650          0.4087
568            0.06444           0.0000                0.0000          0.2871

     fractal_dimension_worst  Unnamed: 32
0                    0.11890          NaN
1                    0.08902          NaN
2                    0.08758          NaN
3                    0.17300          NaN
4                    0.07678          NaN
..                       …            …
564                  0.07115          NaN
565                  0.06637          NaN
566                  0.07820          NaN
567                  0.12400          NaN
568                  0.07039          NaN

[569 rows x 33 columns]
```

[5]:
```
df.head()
```

```
[5]:          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

        …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
     0  …          17.33           184.60      2019.0            0.1622
     1  …          23.41           158.80      1956.0            0.1238
     2  …          25.53           152.50      1709.0            0.1444
     3  …          26.50            98.87       567.7            0.2098
     4  …          16.67           152.20      1575.0            0.1374

        compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
     0             0.6656           0.7119                0.2654          0.4601
     1             0.1866           0.2416                0.1860          0.2750
     2             0.4245           0.4504                0.2430          0.3613
     3             0.8663           0.6869                0.2575          0.6638
     4             0.2050           0.4000                0.1625          0.2364

        fractal_dimension_worst  Unnamed: 32
     0                  0.11890          NaN
     1                  0.08902          NaN
     2                  0.08758          NaN
     3                  0.17300          NaN
     4                  0.07678          NaN

     [5 rows x 33 columns]
```

```
[6]: df.columns
```

```
[6]: 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')
```

[7]:
```
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
```

[8]:
```
df['Unnamed: 32']
```

```
[8]: 0      NaN
     1      NaN
     2      NaN
     3      NaN
     4      NaN
             ..
     564    NaN
     565    NaN
     566    NaN
     567    NaN
     568    NaN
     Name: Unnamed: 32, Length: 569, dtype: float64
```

```
[9]: df = df.drop("Unnamed: 32", axis=1)
```

```
[10]: df.head()
```

```
[10]:          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]
```

```
[11]: df.drop('id', axis=1, inplace=True)
```

```
[12]: l=list(df.columns)
      l
```

```
[12]: ['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']
```

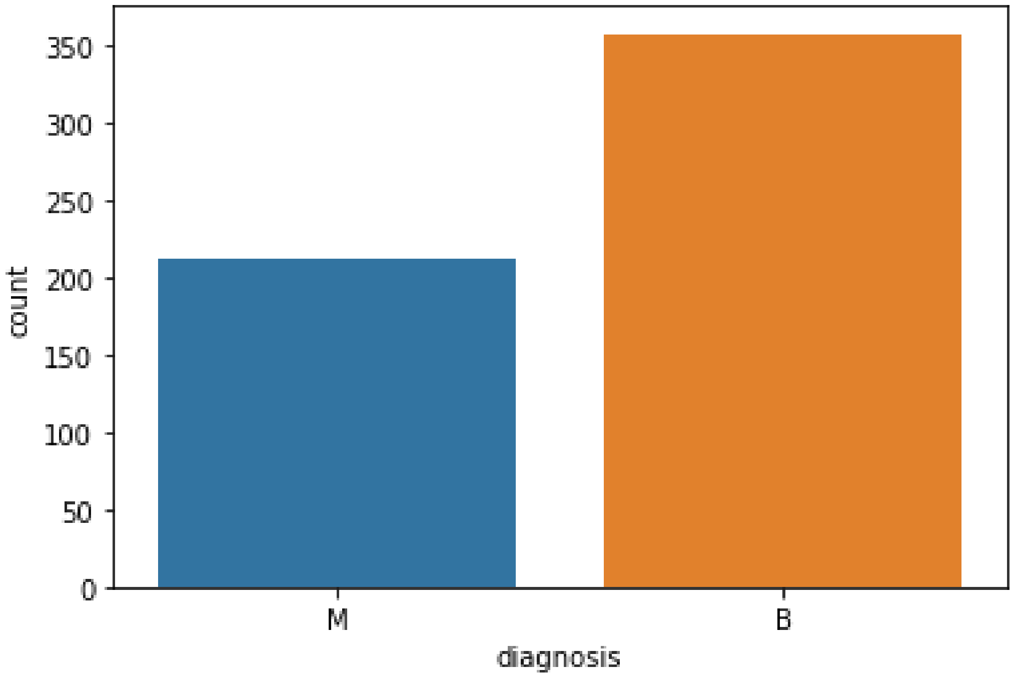
```
[13]: df.head(2)
```

```
[13]:   diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
      0         M        17.99         10.38           122.8     1001.0
      1         M        20.57         17.77           132.9     1326.0

         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

         symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
      0         0.2419  …         25.38          17.33            184.6
      1         0.1812  …         24.99          23.41            158.8

         area_worst  smoothness_worst  compactness_worst  concavity_worst  \
      0      2019.0            0.1622             0.6656           0.7119
      1      1956.0            0.1238             0.1866           0.2416

         concave points_worst  symmetry_worst  fractal_dimension_worst
      0                0.2654          0.4601                  0.11890
      1                0.1860          0.2750                  0.08902

      [2 rows x 31 columns]
```

```
[14]: df['diagnosis'].unique()
```

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

```
[15]: sns.countplot(df['diagnosis'], label="Count",);
```



[16]:
```
df['diagnosis'].value_counts()
```

[16]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

[17]:
```
df.shape
```

[17]:
```
(569, 31)
```

2 Explore The Data

[18]:
```
df.describe()
```

[18]:
```
       radius_mean  texture_mean  perimeter_mean    area_mean  \
count   569.000000    569.000000      569.000000   569.000000
mean     14.127292     19.289649       91.969033   654.889104
std       3.524049      4.301036       24.298981   351.914129
min       6.981000      9.710000       43.790000   143.500000
25%      11.700000     16.170000       75.170000   420.300000
50%      13.370000     18.840000       86.240000   551.100000
75%      15.780000     21.800000      104.100000   782.700000
max      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  fractal_dimension_mean  …  radius_worst  \
count     569.000000              569.000000  …    569.000000
mean        0.181162                0.062798  …     16.269190
std         0.027414                0.007060  …      4.833242
min         0.106000                0.049960  …      7.930000
25%         0.161900                0.057700  …     13.010000
50%         0.179200                0.061540  …     14.970000
75%         0.195700                0.066120  …     18.790000
max         0.304000                0.097440  …     36.040000
```

```
       texture_worst  perimeter_worst   area_worst  smoothness_worst  \
count     569.000000       569.000000   569.000000        569.000000   
mean       25.677223       107.261213   880.583128          0.132369   
std         6.146258        33.602542   569.356993          0.022832   
min        12.020000        50.410000   185.200000          0.071170   
25%        21.080000        84.110000   515.300000          0.116600   
50%        25.410000        97.660000   686.500000          0.131300   
75%        29.720000       125.400000  1084.000000          0.146000   
max        49.540000       251.200000  4254.000000          0.222600   

       compactness_worst  concavity_worst  concave points_worst  \
count         569.000000       569.000000            569.000000   
mean            0.254265         0.272188              0.114606   
std             0.157336         0.208624              0.065732   
min             0.027290         0.000000              0.000000   
25%             0.147200         0.114500              0.064930   
50%             0.211900         0.226700              0.099930   
75%             0.339100         0.382900              0.161400   
max             1.058000         1.252000              0.291000   

       symmetry_worst  fractal_dimension_worst  
count      569.000000               569.000000  
mean         0.290076                 0.083946  
std          0.061867                 0.018061  
min          0.156500                 0.055040  
25%          0.250400                 0.071460  
50%          0.282200                 0.080040  
75%          0.317900                 0.092080  
max          0.663800                 0.207500  

[8 rows x 30 columns]
```

[19]:
```python
#correlation plot
corr = df.corr()
corr
```

[19]:
```
                         radius_mean  texture_mean  perimeter_mean  area_mean  \
radius_mean                 1.000000      0.323782        0.997855   0.987357   
texture_mean                0.323782      1.000000        0.329533   0.321086   
perimeter_mean              0.997855      0.329533        1.000000   0.986507   
area_mean                   0.987357      0.321086        0.986507   1.000000   
smoothness_mean             0.170581     -0.023389        0.207278   0.177028   
compactness_mean            0.506124      0.236702        0.556936   0.498502   
concavity_mean              0.676764      0.302418        0.716136   0.685983   
concave points_mean         0.822529      0.293464        0.850977   0.823269   
symmetry_mean               0.147741      0.071401        0.183027   0.151293   
fractal_dimension_mean     -0.311631     -0.076437       -0.261477  -0.283110   
```

```
radius_se                  0.679090      0.275869         0.691765   0.732562
texture_se                -0.097317      0.386358        -0.086761  -0.066280
perimeter_se               0.674172      0.281673         0.693135   0.726628
area_se                    0.735864      0.259845         0.744983   0.800086
smoothness_se             -0.222600      0.006614        -0.202694  -0.166777
compactness_se             0.206000      0.191975         0.250744   0.212583
concavity_se               0.194204      0.143293         0.228082   0.207660
concave points_se          0.376169      0.163851         0.407217   0.372320
symmetry_se               -0.104321      0.009127        -0.081629  -0.072497
fractal_dimension_se      -0.042641      0.054458        -0.005523  -0.019887
radius_worst               0.969539      0.352573         0.969476   0.962746
texture_worst              0.297008      0.912045         0.303038   0.287489
perimeter_worst            0.965137      0.358040         0.970387   0.959120
area_worst                 0.941082      0.343546         0.941550   0.959213
smoothness_worst           0.119616      0.077503         0.150549   0.123523
compactness_worst          0.413463      0.277830         0.455774   0.390410
concavity_worst            0.526911      0.301025         0.563879   0.512606
concave points_worst       0.744214      0.295316         0.771241   0.722017
symmetry_worst             0.163953      0.105008         0.189115   0.143570
fractal_dimension_worst    0.007066      0.119205         0.051019   0.003738

                         smoothness_mean  compactness_mean  concavity_mean  \
radius_mean                     0.170581          0.506124        0.676764
texture_mean                   -0.023389          0.236702        0.302418
perimeter_mean                  0.207278          0.556936        0.716136
area_mean                       0.177028          0.498502        0.685983
smoothness_mean                 1.000000          0.659123        0.521984
compactness_mean                0.659123          1.000000        0.883121
concavity_mean                  0.521984          0.883121        1.000000
concave points_mean             0.553695          0.831135        0.921391
symmetry_mean                   0.557775          0.602641        0.500667
fractal_dimension_mean          0.584792          0.565369        0.336783
radius_se                       0.301467          0.497473        0.631925
texture_se                      0.068406          0.046205        0.076218
perimeter_se                    0.296092          0.548905        0.660391
area_se                         0.246552          0.455653        0.617427
smoothness_se                   0.332375          0.135299        0.098564
compactness_se                  0.318943          0.738722        0.670279
concavity_se                    0.248396          0.570517        0.691270
concave points_se               0.380676          0.642262        0.683260
symmetry_se                     0.200774          0.229977        0.178009
fractal_dimension_se            0.283607          0.507318        0.449301
radius_worst                    0.213120          0.535315        0.688236
texture_worst                   0.036072          0.248133        0.299879
perimeter_worst                 0.238853          0.590210        0.729565
area_worst                      0.206718          0.509604        0.675987
smoothness_worst                0.805324          0.565541        0.448822
```

```
compactness_worst              0.472468          0.865809        0.754968
concavity_worst                0.434926          0.816275        0.884103
concave points_worst           0.503053          0.815573        0.861323
symmetry_worst                 0.394309          0.510223        0.409464
fractal_dimension_worst        0.499316          0.687382        0.514930

                         concave points_mean  symmetry_mean  \
radius_mean                         0.822529       0.147741
texture_mean                        0.293464       0.071401
perimeter_mean                      0.850977       0.183027
area_mean                           0.823269       0.151293
smoothness_mean                     0.553695       0.557775
compactness_mean                    0.831135       0.602641
concavity_mean                      0.921391       0.500667
concave points_mean                 1.000000       0.462497
symmetry_mean                       0.462497       1.000000
fractal_dimension_mean              0.166917       0.479921
radius_se                           0.698050       0.303379
texture_se                          0.021480       0.128053
perimeter_se                        0.710650       0.313893
area_se                             0.690299       0.223970
smoothness_se                       0.027653       0.187321
compactness_se                      0.490424       0.421659
concavity_se                        0.439167       0.342627
concave points_se                   0.615634       0.393298
symmetry_se                         0.095351       0.449137
fractal_dimension_se                0.257584       0.331786
radius_worst                        0.830318       0.185728
texture_worst                       0.292752       0.090651
perimeter_worst                     0.855923       0.219169
area_worst                          0.809630       0.177193
smoothness_worst                    0.452753       0.426675
compactness_worst                   0.667454       0.473200
concavity_worst                     0.752399       0.433721
concave points_worst                0.910155       0.430297
symmetry_worst                      0.375744       0.699826
fractal_dimension_worst             0.368661       0.438413

                         fractal_dimension_mean  …  radius_worst  \
radius_mean                           -0.311631  …      0.969539
texture_mean                          -0.076437  …      0.352573
perimeter_mean                        -0.261477  …      0.969476
area_mean                             -0.283110  …      0.962746
smoothness_mean                        0.584792  …      0.213120
compactness_mean                       0.565369  …      0.535315
concavity_mean                         0.336783  …      0.688236
concave points_mean                    0.166917  …      0.830318
```

```
symmetry_mean                          0.479921  …      0.185728
fractal_dimension_mean                 1.000000  …     -0.253691
radius_se                              0.000111  …      0.715065
texture_se                             0.164174  …     -0.111690
perimeter_se                           0.039830  …      0.697201
area_se                               -0.090170  …      0.757373
smoothness_se                          0.401964  …     -0.230691
compactness_se                         0.559837  …      0.204607
concavity_se                           0.446630  …      0.186904
concave points_se                      0.341198  …      0.358127
symmetry_se                            0.345007  …     -0.128121
fractal_dimension_se                   0.688132  …     -0.037488
radius_worst                          -0.253691  …      1.000000
texture_worst                         -0.051269  …      0.359921
perimeter_worst                       -0.205151  …      0.993708
area_worst                            -0.231854  …      0.984015
smoothness_worst                       0.504942  …      0.216574
compactness_worst                      0.458798  …      0.475820
concavity_worst                        0.346234  …      0.573975
concave points_worst                   0.175325  …      0.787424
symmetry_worst                         0.334019  …      0.243529
fractal_dimension_worst                0.767297  …      0.093492

                         texture_worst  perimeter_worst  area_worst  \
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  \
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  \
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
```

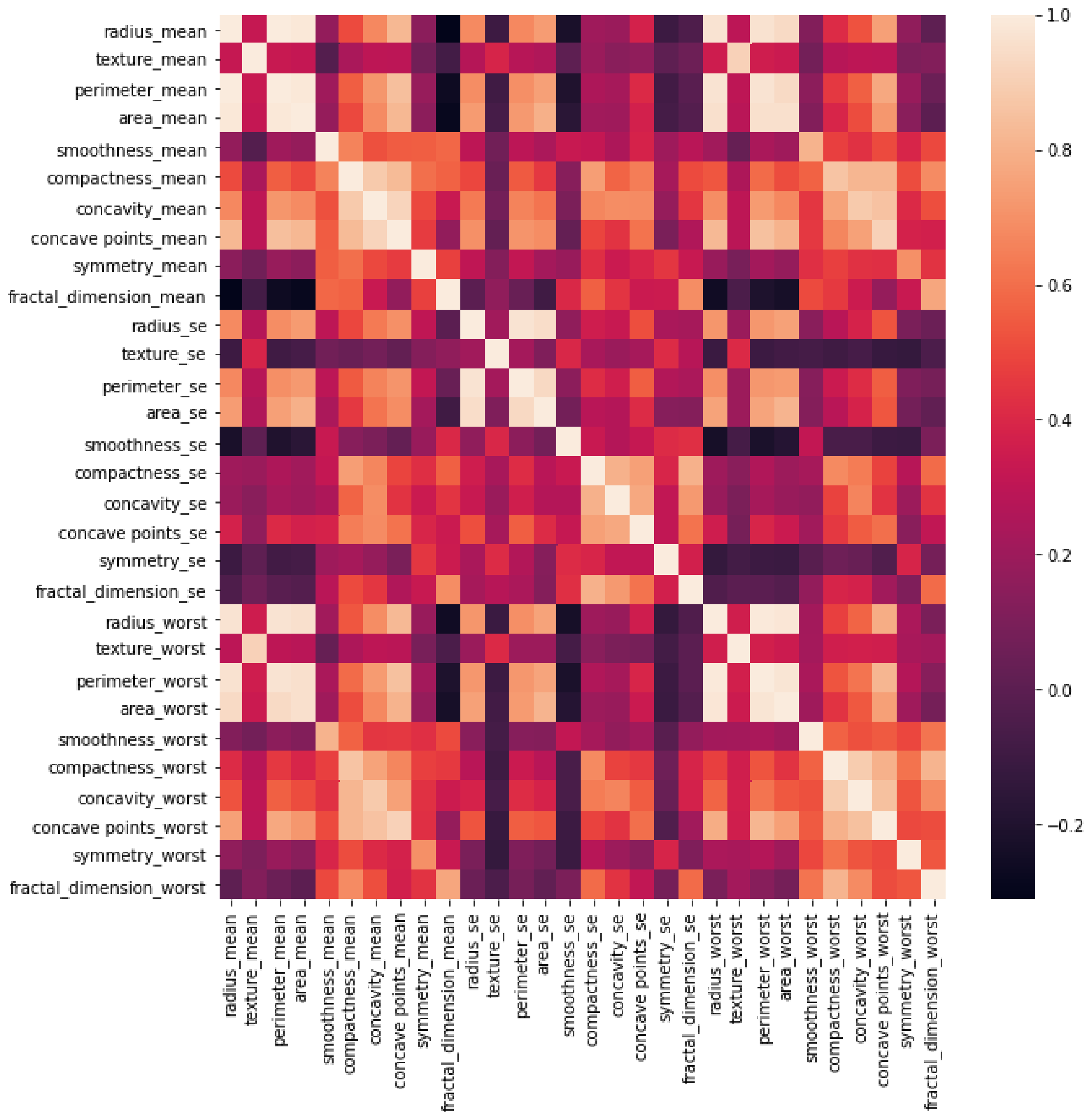
```
concavity_mean                        0.861323        0.409464
concave points_mean                   0.910155        0.375744
symmetry_mean                         0.430297        0.699826
fractal_dimension_mean                0.175325        0.334019
radius_se                             0.531062        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
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

[30 rows x 30 columns]
```

[20]:
```
corr.shape
```

[20]:
```
(30, 30)
```

[21]:
```
plt.figure(figsize=(10,10))
sns.heatmap(corr);
```



[22]:
```
#sns.pairplot(df)
#plt.show()
```

[23]:
```
df.head()
```

[23]:
```
  diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0         M        17.99         10.38          122.80     1001.0
1         M        20.57         17.77          132.90     1326.0
2         M        19.69         21.25          130.00     1203.0
3         M        11.42         20.38           77.58      386.1
4         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   

   symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
0         0.2419  …         25.38          17.33           184.60   
1         0.1812  …         24.99          23.41           158.80   
2         0.2069  …         23.57          25.53           152.50   
3         0.2597  …         14.91          26.50            98.87   
4         0.1809  …         22.54          16.67           152.20   

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119   
1      1956.0            0.1238             0.1866           0.2416   
2      1709.0            0.1444             0.4245           0.4504   
3       567.7            0.2098             0.8663           0.6869   
4      1575.0            0.1374             0.2050           0.4000   

   concave points_worst  symmetry_worst  fractal_dimension_worst  
0                0.2654          0.4601                  0.11890  
1                0.1860          0.2750                  0.08902  
2                0.2430          0.3613                  0.08758  
3                0.2575          0.6638                  0.17300  
4                0.1625          0.2364                  0.07678  

[5 rows x 31 columns]
```

[24]:
```
df['diagnosis'] = df['diagnosis'].map({'M':1, 'B':0})
df.to_csv('tits.csv')
df.head()
```

[24]:
```
   diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0          1        17.99         10.38          122.80     1001.0   
1          1        20.57         17.77          132.90     1326.0   
2          1        19.69         21.25          130.00     1203.0   
3          1        11.42         20.38           77.58      386.1   
4          1        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   
```

```
   symmetry_mean  …  radius_worst  texture_worst  perimeter_worst  \
0         0.2419  …         25.38          17.33           184.60
1         0.1812  …         24.99          23.41           158.80
2         0.2069  …         23.57          25.53           152.50
3         0.2597  …         14.91          26.50            98.87
4         0.1809  …         22.54          16.67           152.20

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   concave points_worst  symmetry_worst  fractal_dimension_worst
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 31 columns]
```

[25]:
```
df['diagnosis'].unique()
```

[25]:
```
array([1, 0], dtype=int64)
```

[26]:
```
X=df.drop('diagnosis',axis=1)
X.head()
```

[26]:
```
   radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
0        17.99         10.38          122.80     1001.0          0.11840
1        20.57         17.77          132.90     1326.0          0.08474
2        19.69         21.25          130.00     1203.0          0.10960
3        11.42         20.38           77.58      386.1          0.14250
4        20.29         14.34          135.10     1297.0          0.10030

   compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   fractal_dimension_mean  …  radius_worst  texture_worst  perimeter_worst  \
0                 0.07871  …         25.38          17.33           184.60
```

```
1                0.05667  …      24.99        23.41          158.80
2                0.05999  …      23.57        25.53          152.50
3                0.09744  …      14.91        26.50           98.87
4                0.05883  …      22.54        16.67          152.20

   area_worst  smoothness_worst  compactness_worst  concavity_worst  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000

   concave points_worst  symmetry_worst  fractal_dimension_worst
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

[27]:
```
y=df['diagnosis']
y.head()
```

[27]:
```
0    1
1    1
2    1
3    1
4    1
Name: diagnosis, dtype: int64
```

3 Train Test Split

[28]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)
```

[29]:
```
print(X_train.shape ,X_test.shape)
print(y_train.shape, y_test.shape)
```

```
(398, 30) (171, 30)
(398,) (171,)
```

[30]:
```
X_train.head(1)
```

[30]:
```
     radius_mean  texture_mean  perimeter_mean  area_mean  smoothness_mean  \
464        13.17         18.22           84.28      537.3          0.07466
```

```
     compactness_mean  concavity_mean  concave points_mean  symmetry_mean  \
464           0.05994         0.04859               0.0287         0.1454

     fractal_dimension_mean  …  radius_worst  texture_worst  \
464                 0.05549  …          14.9          23.89

     perimeter_worst  area_worst  smoothness_worst  compactness_worst  \
464             95.1       687.6            0.1282             0.1965

     concavity_worst  concave points_worst  symmetry_worst  \
464           0.1876                0.1045          0.2235

     fractal_dimension_worst
464                  0.06925

[1 rows x 30 columns]
```

[31]:
```
from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train=sc.fit_transform(X_train)
X_test=sc.transform(X_test)
```

[32]:
```
X_train
```

[32]:
```
array([[-0.29353072, -0.26265738, -0.33745191, …, -0.18152237,
        -1.08268903, -0.80716717],
       [-0.25734632,  0.51215806, -0.29881826, …, -0.91131724,
        -0.44632738, -0.95448921],
       [-0.73052691, -0.23027703, -0.75598312, …, -0.50778716,
         0.46072542, -0.28960885],
       …,
       [-0.56908882, -1.22019048, -0.62036291, …, -1.47903607,
        -0.99087566, -1.2971371 ],
       [-0.20167801, -1.5370553 , -0.28070874, …, -1.42501073,
        -1.03203338, -1.21161307],
       [ 1.74949605,  1.95539636,  1.67954657, …,  0.60018512,
        -0.76292522, -0.03303676]])
```

[33]:
```
X_test
```

[33]:
```
array([[ 0.00707813,  0.64630521,  0.15110279, …,  0.93520261,
        -0.11864862,  1.61785291],
       [-1.24406701,  2.04559876, -1.24695242, …, -1.02419701,
        -0.94655196, -0.64936154],
       [-0.30188096,  0.32018886, -0.26903815, …,  0.02371357,
        -0.56346858, -0.11800902],
       …,
       [-0.46888588, -0.29735061, -0.53303476, …, -1.51178328,
        -1.08585501, -1.58184936],
       [-0.56352199, -0.95883483, -0.58977794, …, -0.92006995,
        -0.40833564, -0.23553559],
       [ 0.05439619, -0.65122154,  0.0597503 , …,  0.37533103,
         0.21219612,  0.20753406]])
```

4 Machine learning Models

4.1 Logistic Regression

[34]:
```
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression(random_state = 5)
lr.fit(X_train,y_train)
```

[34]:
```
LogisticRegression(random_state=5)
```

[35]:
```
y_pred = lr.predict(X_test)
y_pred
```

[35]:
```
array([1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

[36]:
```
y_test
```

[36]:
```
62     1
459    0
466    0
334    0
31     1
      ..
346    0
552    0
360    0
324    0
340    0
Name: diagnosis, Length: 171, dtype: int64
```

[37]:
```
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix')
print(cm)
print("Accuracy Score : ",accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[115   0]
 [  1  55]]
Accuracy Score :  0.9941520467836257
              precision    recall  f1-score   support

           0    0.99138   1.00000   0.99567       115
           1    1.00000   0.98214   0.99099        56

    accuracy                        0.99415       171
   macro avg    0.99569   0.99107   0.99333       171
weighted avg    0.99420   0.99415   0.99414       171
```

[38]:
```
lr_acc = accuracy_score(y_test, y_pred)
```

[39]:
```
results = pd.DataFrame()
results
```

[39]:
```
Empty DataFrame
Columns: []
Index: []
```

[40]:
```
tempResult = pd.DataFrame({'Algorithm':['Logistic Regression Method'],␣
 ↪'Accuracy':[lr_acc]})
results = pd.concat([results, tempResult])
results
```

[40]:
```
                    Algorithm  Accuracy
0  Logistic Regression Method  0.994152
```

5 Decision Tree Classifier

[41]:
```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
```

[41]:
```
DecisionTreeClassifier()
```

[42]:
```
y_pred = dtc.predict(X_test)
y_pred
```

[42]:
```
array([1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1], dtype=int64)
```

[43]:
```
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix')
print(cm)
print("Accuracy Score : ",accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[109   6]
 [ 10  46]]
Accuracy Score :  0.9064327485380117
              precision    recall  f1-score   support

           0    0.91597   0.94783   0.93162       115
           1    0.88462   0.82143   0.85185        56

    accuracy                        0.90643       171
   macro avg    0.90029   0.88463   0.89174       171
weighted avg    0.90570   0.90643   0.90550       171
```

[44]:
```
dtc_acc = accuracy_score(y_test, y_pred)
```

[45]:
```
tempResult = pd.DataFrame({'Algorithm':['Decision Tree Classifier Method'],␣
 ↪'Accuracy':[dtc_acc]})
results = pd.concat([results, tempResult])
results = results[['Algorithm','Accuracy']]
results
```

[45]:
```
                         Algorithm  Accuracy
0       Logistic Regression Method  0.994152
0  Decision Tree Classifier Method  0.906433
```

6 Random Forest Classifier

[46]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators = 10, criterion = 'entropy',␣
 ↪random_state = 0)
rfc.fit(X_train, y_train)
```

[46]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

[47]:
```
y_pred = rfc.predict(X_test)
y_pred
```

[47]:
```
array([1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

[48]:
```
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix')
print(cm)
print("Accuracy Score : ",accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[112   3]
 [  4  52]]
Accuracy Score :  0.9590643274853801
              precision    recall  f1-score   support

           0    0.96552   0.97391   0.96970       115
           1    0.94545   0.92857   0.93694        56

    accuracy                        0.95906       171
   macro avg    0.95549   0.95124   0.95332       171
weighted avg    0.95895   0.95906   0.95897       171
```

[49]:
```
rfc_acc = accuracy_score(y_test, y_pred)
print(rfc_acc)
```

```
0.9590643274853801
```

[50]:
```
tempResults = pd.DataFrame({'Algorithm':['Random Forest Classifier Method'],␣
 ↪'Accuracy':[rfc_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

[50]:
```
                         Algorithm  Accuracy
0       Logistic Regression Method  0.994152
0  Decision Tree Classifier Method  0.906433
0  Random Forest Classifier Method  0.959064
```

7 Support Vector Classifier

[51]:
```
from sklearn import svm
svc = svm.SVC()
svc.fit(X_train,y_train)
```

[51]:
```
SVC()
```

[52]:
```
y_pred = svc.predict(X_test)
y_pred
```

[52]:
```
array([1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

[53]:
```
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix')
print(cm)
print("Accuracy Score : ",accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[115   0]
 [  3  53]]
Accuracy Score :  0.9824561403508771
              precision    recall  f1-score   support

           0    0.97458   1.00000   0.98712       115
           1     1.00000    0.94643    0.97248        56

    accuracy                           0.98246       171
   macro avg     0.98729    0.97321    0.97980       171
weighted avg     0.98290    0.98246    0.98233       171
```

[54]:
```python
svc_acc = accuracy_score(y_test, y_pred)
print(svc_acc)
```

```
0.9824561403508771
```

[55]:
```python
tempResults = pd.DataFrame({'Algorithm':['Support Vector Classifier Method'],␣
 ↪'Accuracy':[svc_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

[55]:
```
                          Algorithm  Accuracy
0          Logistic Regression Method  0.994152
0     Decision Tree Classifier Method  0.906433
0     Random Forest Classifier Method  0.959064
0    Support Vector Classifier Method  0.982456
```

8 KNN Classifier

[56]:
```python
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3, metric = 'euclidean', p = 2)
knn.fit(X_train, y_train)
```

[56]:
```
KNeighborsClassifier(metric='euclidean', n_neighbors=3)
```

[57]:
```python
y_pred = knn.predict(X_test)
y_pred
```

[57]:
```
array([1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

[58]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
cm = confusion_matrix(y_test, y_pred)
```

```
print('Confusion Matrix')
print(cm)
print("Accuracy Score : ",accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[112   3]
 [  3  53]]
Accuracy Score :  0.9649122807017544
              precision    recall  f1-score   support

           0    0.97391   0.97391   0.97391       115
           1    0.94643   0.94643   0.94643        56

    accuracy                        0.96491       171
   macro avg    0.96017   0.96017   0.96017       171
weighted avg    0.96491   0.96491   0.96491       171
```

[59]:
```
knn_acc = accuracy_score(y_test, y_pred)
print(knn_acc)
```

```
0.9649122807017544
```

[60]:
```
tempResults = pd.DataFrame({'Algorithm':['K-Nearest-Neighbor Classification␣
 ↪Method'], 'Accuracy':[knn_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

[60]:
```
                                  Algorithm  Accuracy
0              Logistic Regression Method  0.994152
0         Decision Tree Classifier Method  0.906433
0         Random Forest Classifier Method  0.959064
0        Support Vector Classifier Method  0.982456
0  K-Nearest-Neighbor Classification Method  0.964912
```

9 Neive Bayes Classifier

[61]:
```
from sklearn.naive_bayes import GaussianNB
nbc = GaussianNB()
nbc.fit(X_train, y_train)
```

[61]:
```
GaussianNB()
```

```
[62]: y_pred = nbc.predict(X_test)
      y_pred
```

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

```
[63]: from sklearn.metrics import confusion_matrix, accuracy_score,␣
       ↪classification_report
      cm = confusion_matrix(y_test, y_pred)
      print('Confusion Matrix')
      print(cm)
      print("Accuracy Score : ",accuracy_score(y_test, y_pred))
      print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[110   5]
 [  5  51]]
Accuracy Score :  0.9415204678362573
              precision    recall  f1-score   support

           0    0.95652   0.95652   0.95652       115
           1    0.91071   0.91071   0.91071        56

    accuracy                        0.94152       171
   macro avg    0.93362   0.93362   0.93362       171
weighted avg    0.94152   0.94152   0.94152       171
```

```
[64]: nbc_acc = accuracy_score(y_test, y_pred)
      print(nbc_acc)
```

```
0.9415204678362573
```

```
[65]: tempResults = pd.DataFrame({'Algorithm':['Neive Bayes Classification Method'],␣
       ↪'Accuracy':[nbc_acc]})
      results = pd.concat( [results, tempResults] )
      results = results[['Algorithm','Accuracy']]
      results
```

```
[65]:                                  Algorithm  Accuracy
      0                 Logistic Regression Method  0.994152
```

```
0           Decision Tree Classifier Method  0.906433
0           Random Forest Classifier Method  0.959064
0          Support Vector Classifier Method  0.982456
0  K-Nearest-Neighbor Classification Method  0.964912
0         Neive Bayes Classification Method  0.941520
```

10 ANN

[66]:
```
import tensorflow as tf
```

[67]:
```
ann = tf.keras.models.Sequential()
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))
ann.add(tf.keras.layers.Dense(units=6, activation='relu'))

ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

[68]:
```
ann.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics =␣
 ↪['accuracy'])
```

[69]:
```
ann.fit(X_train, y_train, batch_size = 32, epochs = 100)
```

```
Epoch 1/100
WARNING:tensorflow:AutoGraph could not transform <function
Model.make_train_function.<locals>.train_function at 0x0000024550D588C8> and
will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the
verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full
output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with
@tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <function
Model.make_train_function.<locals>.train_function at 0x0000024550D588C8> and
will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the
verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full
output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with
@tf.autograph.experimental.do_not_convert
13/13 [==============================] - 0s 4ms/step - loss: 0.6930 - accuracy:
0.4673
Epoch 2/100
13/13 [==============================] - 0s 4ms/step - loss: 0.6579 - accuracy:
0.5678
```

```
Epoch 3/100
13/13 [==============================] - 0s 4ms/step - loss: 0.6321 - accuracy:
0.6709
Epoch 4/100
13/13 [==============================] - 0s 4ms/step - loss: 0.6052 - accuracy:
0.7688
Epoch 5/100
13/13 [==============================] - 0s 4ms/step - loss: 0.5782 - accuracy:
0.8266
Epoch 6/100
13/13 [==============================] - 0s 4ms/step - loss: 0.5515 - accuracy:
0.8643
Epoch 7/100
13/13 [==============================] - 0s 3ms/step - loss: 0.5255 - accuracy:
0.8794
Epoch 8/100
13/13 [==============================] - 0s 4ms/step - loss: 0.4997 - accuracy:
0.8970
Epoch 9/100
13/13 [==============================] - 0s 4ms/step - loss: 0.4774 - accuracy:
0.9095
Epoch 10/100
13/13 [==============================] - 0s 4ms/step - loss: 0.4582 - accuracy:
0.9196
Epoch 11/100
13/13 [==============================] - 0s 4ms/step - loss: 0.4420 - accuracy:
0.9196
Epoch 12/100
13/13 [==============================] - 0s 4ms/step - loss: 0.4280 - accuracy:
0.9271
Epoch 13/100
13/13 [==============================] - 0s 4ms/step - loss: 0.4141 - accuracy:
0.9271
Epoch 14/100
13/13 [==============================] - 0s 3ms/step - loss: 0.4006 - accuracy:
0.9372
Epoch 15/100
13/13 [==============================] - 0s 4ms/step - loss: 0.3874 - accuracy:
0.9397
Epoch 16/100
13/13 [==============================] - 0s 4ms/step - loss: 0.3737 - accuracy:
0.9497
Epoch 17/100
13/13 [==============================] - 0s 4ms/step - loss: 0.3602 - accuracy:
0.9548
Epoch 18/100
13/13 [==============================] - 0s 3ms/step - loss: 0.3465 - accuracy:
0.9573
```

```
Epoch 19/100
13/13 [==============================] - 0s 4ms/step - loss: 0.3328 - accuracy:
0.9623
Epoch 20/100
13/13 [==============================] - 0s 4ms/step - loss: 0.3183 - accuracy:
0.9673
Epoch 21/100
13/13 [==============================] - 0s 4ms/step - loss: 0.3033 - accuracy:
0.9673
Epoch 22/100
13/13 [==============================] - 0s 3ms/step - loss: 0.2861 - accuracy:
0.9698
Epoch 23/100
13/13 [==============================] - ETA: 0s - loss: 0.2969 - accuracy: 0.93
- 0s 4ms/step - loss: 0.2670 - accuracy: 0.9698
Epoch 24/100
13/13 [==============================] - 0s 4ms/step - loss: 0.2451 - accuracy:
0.9698
Epoch 25/100
13/13 [==============================] - 0s 4ms/step - loss: 0.2207 - accuracy:
0.9698
Epoch 26/100
13/13 [==============================] - 0s 3ms/step - loss: 0.1967 - accuracy:
0.9698
Epoch 27/100
13/13 [==============================] - 0s 4ms/step - loss: 0.1736 - accuracy:
0.9698
Epoch 28/100
13/13 [==============================] - 0s 4ms/step - loss: 0.1532 - accuracy:
0.9698
Epoch 29/100
13/13 [==============================] - 0s 3ms/step - loss: 0.1354 - accuracy:
0.9749
Epoch 30/100
13/13 [==============================] - 0s 3ms/step - loss: 0.1214 - accuracy:
0.9774
Epoch 31/100
13/13 [==============================] - 0s 3ms/step - loss: 0.1105 - accuracy:
0.9774
Epoch 32/100
13/13 [==============================] - 0s 4ms/step - loss: 0.1020 - accuracy:
0.9774
Epoch 33/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0951 - accuracy:
0.9774
Epoch 34/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0896 - accuracy:
0.9799
```

```
Epoch 35/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0853 - accuracy:
0.9824
Epoch 36/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0815 - accuracy:
0.9824
Epoch 37/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0785 - accuracy:
0.9824
Epoch 38/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0760 - accuracy:
0.9824
Epoch 39/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0735 - accuracy:
0.9849
Epoch 40/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0716 - accuracy:
0.9824
Epoch 41/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0698 - accuracy:
0.9849
Epoch 42/100
13/13 [==============================] - 0s 5ms/step - loss: 0.0680 - accuracy:
0.9849
Epoch 43/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0667 - accuracy:
0.9849
Epoch 44/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0654 - accuracy:
0.9849
Epoch 45/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0644 - accuracy:
0.9849
Epoch 46/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0633 - accuracy:
0.9849
Epoch 47/100
13/13 [==============================] - 0s 4ms/step - loss: 0.0622 - accuracy:
0.9849
Epoch 48/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0613 - accuracy:
0.9874
Epoch 49/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0602 - accuracy:
0.9874
Epoch 50/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0595 - accuracy:
0.9874
```

```
Epoch 51/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0586 - accuracy:
0.9874
Epoch 52/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0572 - accuracy:
0.9874
Epoch 53/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0565 - accuracy:
0.9874
Epoch 54/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0556 - accuracy:
0.9874
Epoch 55/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0547 - accuracy:
0.9874
Epoch 56/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0540 - accuracy:
0.9874
Epoch 57/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0531 - accuracy:
0.9874
Epoch 58/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0524 - accuracy:
0.9874
Epoch 59/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0521 - accuracy:
0.9874
Epoch 60/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0513 - accuracy:
0.9899
Epoch 61/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0507 - accuracy:
0.9899
Epoch 62/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0500 - accuracy:
0.9899
Epoch 63/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0494 - accuracy:
0.9899
Epoch 64/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0487 - accuracy:
0.9899
Epoch 65/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0483 - accuracy:
0.9899
Epoch 66/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0478 - accuracy:
0.9899
```

```
Epoch 67/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0472 - accuracy:
0.9899
Epoch 68/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0468 - accuracy:
0.9899
Epoch 69/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0463 - accuracy:
0.9899
Epoch 70/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0459 - accuracy:
0.9899
Epoch 71/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0455 - accuracy:
0.9899
Epoch 72/100
13/13 [==============================] - 0s 1ms/step - loss: 0.0450 - accuracy:
0.9899
Epoch 73/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0447 - accuracy:
0.9899
Epoch 74/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0442 - accuracy:
0.9899
Epoch 75/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0437 - accuracy:
0.9899
Epoch 76/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0434 - accuracy:
0.9899
Epoch 77/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0430 - accuracy:
0.9899
Epoch 78/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0426 - accuracy:
0.9899
Epoch 79/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0422 - accuracy:
0.9899
Epoch 80/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0420 - accuracy:
0.9899
Epoch 81/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0416 - accuracy:
0.9925
Epoch 82/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0411 - accuracy:
0.9899
```

```
Epoch 83/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0409 - accuracy:
0.9925
Epoch 84/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0403 - accuracy:
0.9925
Epoch 85/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0399 - accuracy:
0.9925
Epoch 86/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0396 - accuracy:
0.9925
Epoch 87/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0393 - accuracy:
0.9925
Epoch 88/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0390 - accuracy:
0.9925
Epoch 89/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0386 - accuracy:
0.9925
Epoch 90/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0383 - accuracy:
0.9925
Epoch 91/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0379 - accuracy:
0.9925
Epoch 92/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0377 - accuracy:
0.9925
Epoch 93/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0373 - accuracy:
0.9925
Epoch 94/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0369 - accuracy:
0.9925
Epoch 95/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0368 - accuracy:
0.9925
Epoch 96/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0364 - accuracy:
0.9925
Epoch 97/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0361 - accuracy:
0.9925
Epoch 98/100
13/13 [==============================] - 0s 3ms/step - loss: 0.0358 - accuracy:
0.9925
```

```
Epoch 99/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0354 - accuracy:
0.9925
Epoch 100/100
13/13 [==============================] - 0s 2ms/step - loss: 0.0352 - accuracy:
0.9925
```

[69]:
```
<tensorflow.python.keras.callbacks.History at 0x24551194198>
```

[70]:
```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
type(y_pred)
y_pred = y_pred+0
y_pred = y_pred.reshape(len(y_pred))
y_pred
```

```
WARNING:tensorflow:AutoGraph could not transform <function
Model.make_predict_function.<locals>.predict_function at 0x000002455B003620> and
will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the
verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full
output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with
@tf.autograph.experimental.do_not_convert
WARNING: AutoGraph could not transform <function
Model.make_predict_function.<locals>.predict_function at 0x000002455B003620> and
will run it as-is.
Please report this to the TensorFlow team. When filing the bug, set the
verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full
output.
Cause: 'arguments' object has no attribute 'posonlyargs'
To silence this warning, decorate the function with
@tf.autograph.experimental.do_not_convert
```

[70]:
```
array([1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0])
```

[71]:
```
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
cm = confusion_matrix(y_test, y_pred)
```

```
print('Confusion Matrix')
print(cm)
print("Accuracy Score : ",accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[114   1]
 [  3  53]]
Accuracy Score :  0.9766081871345029
              precision    recall  f1-score   support

           0    0.97436   0.99130   0.98276       115
           1    0.98148   0.94643   0.96364        56

    accuracy                        0.97661       171
   macro avg    0.97792   0.96887   0.97320       171
weighted avg    0.97669   0.97661   0.97650       171
```

[72]:
```
ann_acc = accuracy_score(y_test, y_pred)
print(ann_acc)
```

```
0.9766081871345029
```

[73]:
```
tempResults = pd.DataFrame({'Algorithm':['Artificial Neural Network Method'],␣
 ↪'Accuracy':[ann_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

[73]:
```
                                  Algorithm  Accuracy
0                Logistic Regression Method  0.994152
0             Decision Tree Classifier Method  0.906433
0             Random Forest Classifier Method  0.959064
0            Support Vector Classifier Method  0.982456
0  K-Nearest-Neighbor Classification Method  0.964912
0           Neive Bayes Classification Method  0.941520
0            Artificial Neural Network Method  0.976608
```

11 Stochastic Gradient Descent

[74]:
```
from sklearn.linear_model import SGDClassifier
sgd = SGDClassifier()
sgd.fit(X_train, y_train)
```

[74]:
```
SGDClassifier()
```

```
[75]: y_pred = sgd.predict(X_test)
      y_pred
```

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

```
[76]: from sklearn.metrics import confusion_matrix, accuracy_score,␣
       ↪classification_report
      cm = confusion_matrix(y_test, y_pred)
      print('Confusion Matrix')
      print(cm)
      print("Accuracy Score : ",accuracy_score(y_test, y_pred))
      print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[113   2]
 [  2  54]]
Accuracy Score :  0.9766081871345029
              precision    recall  f1-score   support

           0    0.98261   0.98261   0.98261       115
           1    0.96429   0.96429   0.96429        56

    accuracy                        0.97661       171
   macro avg    0.97345   0.97345   0.97345       171
weighted avg    0.97661   0.97661   0.97661       171
```

```
[77]: sgd_acc = accuracy_score(y_test, y_pred)
      print(sgd_acc)
```

```
0.9766081871345029
```

```
[78]: tempResults = pd.DataFrame({'Algorithm':['Stochastic Gradient Descent Method'],␣
       ↪'Accuracy':[sgd_acc]})
      results = pd.concat( [results, tempResults] )
      results = results[['Algorithm','Accuracy']]
      results
```

```
[78]:                                 Algorithm  Accuracy
      0                 Logistic Regression Method  0.994152
```

```
0           Decision Tree Classifier Method  0.906433
0           Random Forest Classifier Method  0.959064
0          Support Vector Classifier Method  0.982456
0  K-Nearest-Neighbor Classification Method  0.964912
0         Neive Bayes Classification Method  0.941520
0          Artificial Neural Network Method  0.976608
0        Stochastic Gradient Descent Method  0.976608
```

12 Adaboost

[79]:
```
from sklearn.ensemble import AdaBoostClassifier
ab=AdaBoostClassifier(n_estimators=2500)
ab.fit(X_train, y_train)
```

[79]:
```
AdaBoostClassifier(n_estimators=2500)
```

[80]:
```
y_pred = ab.predict(X_test)
y_pred
```

[80]:
```
array([1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

[81]:
```
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix')
print(cm)
print("Accuracy Score : ",accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[115   0]
 [  2  54]]
Accuracy Score :  0.9883040935672515
              precision    recall  f1-score   support

           0    0.98291   1.00000   0.99138       115
           1    1.00000   0.96429   0.98182        56

    accuracy                        0.98830       171
```

```
   macro avg     0.99145   0.98214   0.98660       171
weighted avg     0.98850   0.98830   0.98825       171
```

[82]:
```
ab_acc = accuracy_score(y_test, y_pred)
print(ab_acc)
```

```
0.9883040935672515
```

[83]:
```
tempResults = pd.DataFrame({'Algorithm':['AdaBoost Method'], 'Accuracy':
 ↪[ab_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

[83]:
```
                                  Algorithm  Accuracy
0                  Logistic Regression Method  0.994152
0              Decision Tree Classifier Method  0.906433
0              Random Forest Classifier Method  0.959064
0             Support Vector Classifier Method  0.982456
0   K-Nearest-Neighbor Classification Method  0.964912
0           Neive Bayes Classification Method  0.941520
0            Artificial Neural Network Method  0.976608
0          Stochastic Gradient Descent Method  0.976608
0                             AdaBoost Method  0.988304
```

13 Multi Layer Neuron Classifier

[84]:
```
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2),␣
 ↪random_state=50)
mlp.fit(X_train, y_train)
```

[84]:
```
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(5, 2), random_state=50,
              solver='lbfgs')
```

[85]:
```
y_pred = mlp.predict(X_test)
y_pred
```

[85]:
```
array([1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0], dtype=int64)
```

[86]:
```
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report
cm = confusion_matrix(y_test, y_pred)
print('Confusion Matrix')
print(cm)
print("Accuracy Score : ",accuracy_score(y_test, y_pred))
print(classification_report(y_test,y_pred, digits=5))
```

```
Confusion Matrix
[[113   2]
 [  3  53]]
Accuracy Score :  0.9707602339181286
              precision    recall  f1-score   support

           0    0.97414   0.98261   0.97835       115
           1    0.96364   0.94643   0.95495        56

    accuracy                        0.97076       171
   macro avg    0.96889   0.96452   0.96665       171
weighted avg    0.97070   0.97076   0.97069       171
```

[87]:
```
mlp_acc = accuracy_score(y_test, y_pred)
print(mlp_acc)
```

```
0.9707602339181286
```

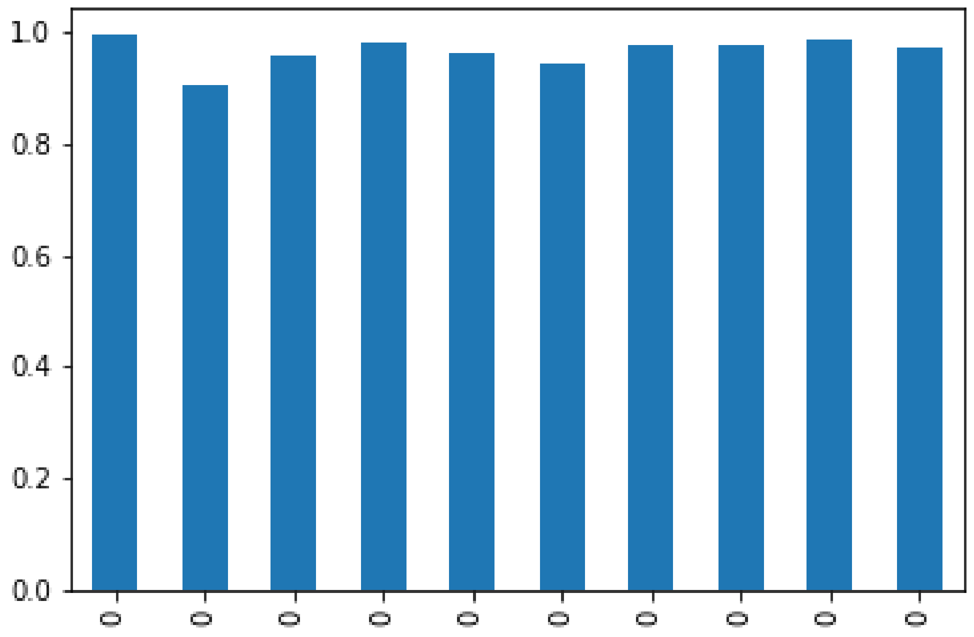
[88]:
```
tempResults = pd.DataFrame({'Algorithm':['Multi Layer Neuron Classification␣
 ↪Method'], 'Accuracy':[mlp_acc]})
results = pd.concat( [results, tempResults] )
results = results[['Algorithm','Accuracy']]
results
```

[88]:
```
                                Algorithm  Accuracy
0                Logistic Regression Method  0.994152
0             Decision Tree Classifier Method  0.906433
0             Random Forest Classifier Method  0.959064
0            Support Vector Classifier Method  0.982456
0   K-Nearest-Neighbor Classification Method  0.964912
0          Neive Bayes Classification Method  0.941520
0            Artificial Neural Network Method  0.976608
0          Stochastic Gradient Descent Method  0.976608
0                          AdaBoost Method  0.988304
0   Multi Layer Neuron Classification Method  0.970760
```

[89]:
```
results.to_csv('accuracy.csv')
```

[90]:
```
results.Accuracy.plot.bar()
```

[90]:
```
<AxesSubplot:>
```



[91]:
```
results['Accuracy'].value_counts()
```

[91]:
```
0.976608    2
0.964912    1
0.988304    1
0.970760    1
0.994152    1
0.959064    1
0.941520    1
0.906433    1
0.982456    1
Name: Accuracy, dtype: int64
```

14 Input The Values

[92]:
```
#input
l = [13.54,14.36,87.46,566.3,0.09779,0.08129,0.06664,0.04781,
     0.1885,0.05766,0.2699,0.7886,2.058,23.56,0.008462,0.0146,0.02387,0.01315,
    0.0198,0.0023,15.11,19.26,99.7,711.2,0.144,0.1773,0.239,0.1288,0.2977,0.
 ↪07259]
```

15 Prediction By all the Classifiers

[93]:
```
print("Breast Cancer Detection by Logistic Regression Classifier : ",lr.
 ↪predict(sc.transform([l])))
print("Breast Cancer Detection by Desicion Tree Classifier : ",dtc.predict(sc.
 ↪transform([l])))
print("Breast Cancer Detection by Random Forest Classifier : ",rfc.predict(sc.
 ↪transform([l])))
print("Breast Cancer Detection by Support Vector Machine Classifier : ",svc.
 ↪predict(sc.transform([l])))
print("Breast Cancer Detection by K-Nearest-Neighbor Classifier : ",knn.
 ↪predict(sc.transform([l])))
print("Breast Cancer Detection by Neive Based Classifier : ",nbc.predict(sc.
 ↪transform([l])))
print("Breast Cancer Detection by Artificial Neural Network Classifier : ",ann.
 ↪predict(sc.transform([l])))
print("Breast Cancer Detection by Stochastic Gradient Descent Classifier :␣
 ↪",sgd.predict(sc.transform([l])))
print("Breast Cancer Detection by AdaBoost Classifier : ",ab.predict(sc.
 ↪transform([l])))
print("Breast Cancer Detection by Multi Layer Neuron Classifier : ",mlp.
 ↪predict(sc.transform([l])))
```

```
Breast Cancer Detection by Logistic Regression Classifier :  [0]
Breast Cancer Detection by Desicion Tree Classifier :  [0]
Breast Cancer Detection by Random Forest Classifier :  [0]
Breast Cancer Detection by Support Vector Machine Classifier :  [0]
Breast Cancer Detection by K-Nearest-Neighbor Classifier :  [0]
Breast Cancer Detection by Neive Based Classifier :  [0]
Breast Cancer Detection by Artificial Neural Network Classifier :
[[0.01814982]]
Breast Cancer Detection by Stochastic Gradient Descent Classifier :  [0]
Breast Cancer Detection by AdaBoost Classifier :  [0]
Breast Cancer Detection by Multi Layer Neuron Classifier :  [0]
```

[ ]:
```
```