In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot
from tensorflow import keras
```

In [9]:

```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.preprocessing import MinMaxScaler,LabelEncoder


from sklearn.model_selection import cross_val_score

from sklearn.metrics import accuracy_score,precision_score,recall_score,f1_score, confu
sion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
```

1. Veri önişleme

In [69]:

```
# veri yükleme işlemi
df = pd.read_csv("datasets_2_breast-cancer.csv")
```

In [70]:

```
df[0:50]
```

Out[70]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothne
0	842302	M	17.990	10.38	122.80	1001.0	
1	842517	M	20.570	17.77	132.90	1326.0	
2	84300903	M	19.690	21.25	130.00	1203.0	
3	84348301	M	11.420	20.38	77.58	386.1	
4	84358402	M	20.290	14.34	135.10	1297.0	
5	843786	M	12.450	15.70	82.57	477.1	
6	844359	M	18.250	19.98	119.60	1040.0	
7	84458202	M	13.710	20.83	90.20	577.9	
8	844981	M	13.000	21.82	87.50	519.8	
9	84501001	M	12.460	24.04	83.97	475.9	
10	845636	M	16.020	23.24	102.70	797.8	
11	84610002	M	15.780	17.89	103.60	781.0	
12	846226	M	19.170	24.80	132.40	1123.0	
13	846381	M	15.850	23.95	103.70	782.7	
14	84667401	M	13.730	22.61	93.60	578.3	
15	84799002	M	14.540	27.54	96.73	658.8	
16	848406	M	14.680	20.13	94.74	684.5	
17	84862001	M	16.130	20.68	108.10	798.8	
18	849014	M	19.810	22.15	130.00	1260.0	
19	8510426	B	13.540	14.36	87.46	566.3	
20	8510653	B	13.080	15.71	85.63	520.0	
21	8510824	B	9.504	12.44	60.34	273.9	
22	8511133	M	15.340	14.26	102.50	704.4	
23	851509	M	21.160	23.04	137.20	1404.0	
24	852552	M	16.650	21.38	110.00	904.6	
25	852631	M	17.140	16.40	116.00	912.7	
26	852763	M	14.580	21.53	97.41	644.8	
27	852781	M	18.610	20.25	122.10	1094.0	
28	852973	M	15.300	25.27	102.40	732.4	
29	853201	M	17.570	15.05	115.00	955.1	
30	853401	M	18.630	25.11	124.80	1088.0	
31	853612	M	11.840	18.70	77.93	440.6	
32	85382601	M	17.020	23.98	112.80	899.3	
33	854002	M	19.270	26.47	127.90	1162.0	
34	854039	M	16.130	17.88	107.00	807.2	
35	854253	M	16.740	21.59	110.10	869.5	

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothne
36	854268	M	14.250	21.72	93.63	633.0	
37	854941	B	13.030	18.42	82.61	523.8	
38	855133	M	14.990	25.20	95.54	698.8	
39	855138	M	13.480	20.82	88.40	559.2	
40	855167	M	13.440	21.58	86.18	563.0	
41	855563	M	10.950	21.35	71.90	371.1	
42	855625	M	19.070	24.81	128.30	1104.0	
43	856106	M	13.280	20.28	87.32	545.2	
44	85638502	M	13.170	21.81	85.42	531.5	
45	857010	M	18.650	17.60	123.70	1076.0	
46	85713702	B	8.196	16.84	51.71	201.9	
47	85715	M	13.170	18.66	85.98	534.6	
48	857155	B	12.050	14.63	78.04	449.3	
49	857156	B	13.490	22.30	86.91	561.0	

50 rows × 33 columns

In [71]:

```
labelencoder = LabelEncoder()   # diagnosis kolonu kategorik veri olduğu için(M ve B )
 numerik veriye dönüştürme işlemi yapılır.
```

In [72]:

```
df['diagnosis'] = labelencoder.fit_transform(df['diagnosis']) # dönüştürüp aynı sütuna
 numerik veri yazılır
```

In [73]:

```
df[0:50]
```

Out[73]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothne
0	842302	1	17.990	10.38	122.80	1001.0	
1	842517	1	20.570	17.77	132.90	1326.0	
2	84300903	1	19.690	21.25	130.00	1203.0	
3	84348301	1	11.420	20.38	77.58	386.1	
4	84358402	1	20.290	14.34	135.10	1297.0	
5	843786	1	12.450	15.70	82.57	477.1	
6	844359	1	18.250	19.98	119.60	1040.0	
7	84458202	1	13.710	20.83	90.20	577.9	
8	844981	1	13.000	21.82	87.50	519.8	
9	84501001	1	12.460	24.04	83.97	475.9	
10	845636	1	16.020	23.24	102.70	797.8	
11	84610002	1	15.780	17.89	103.60	781.0	
12	846226	1	19.170	24.80	132.40	1123.0	
13	846381	1	15.850	23.95	103.70	782.7	
14	84667401	1	13.730	22.61	93.60	578.3	
15	84799002	1	14.540	27.54	96.73	658.8	
16	848406	1	14.680	20.13	94.74	684.5	
17	84862001	1	16.130	20.68	108.10	798.8	
18	849014	1	19.810	22.15	130.00	1260.0	
19	8510426	0	13.540	14.36	87.46	566.3	
20	8510653	0	13.080	15.71	85.63	520.0	
21	8510824	0	9.504	12.44	60.34	273.9	
22	8511133	1	15.340	14.26	102.50	704.4	
23	851509	1	21.160	23.04	137.20	1404.0	
24	852552	1	16.650	21.38	110.00	904.6	
25	852631	1	17.140	16.40	116.00	912.7	
26	852763	1	14.580	21.53	97.41	644.8	
27	852781	1	18.610	20.25	122.10	1094.0	
28	852973	1	15.300	25.27	102.40	732.4	
29	853201	1	17.570	15.05	115.00	955.1	
30	853401	1	18.630	25.11	124.80	1088.0	
31	853612	1	11.840	18.70	77.93	440.6	
32	85382601	1	17.020	23.98	112.80	899.3	
33	854002	1	19.270	26.47	127.90	1162.0	
34	854039	1	16.130	17.88	107.00	807.2	
35	854253	1	16.740	21.59	110.10	869.5	

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothne
36	854268	1	14.250	21.72	93.63	633.0	
37	854941	0	13.030	18.42	82.61	523.8	
38	855133	1	14.990	25.20	95.54	698.8	
39	855138	1	13.480	20.82	88.40	559.2	
40	855167	1	13.440	21.58	86.18	563.0	
41	855563	1	10.950	21.35	71.90	371.1	
42	855625	1	19.070	24.81	128.30	1104.0	
43	856106	1	13.280	20.28	87.32	545.2	
44	85638502	1	13.170	21.81	85.42	531.5	
45	857010	1	18.650	17.60	123.70	1076.0	
46	85713702	0	8.196	16.84	51.71	201.9	
47	85715	1	13.170	18.66	85.98	534.6	
48	857155	0	12.050	14.63	78.04	449.3	
49	857156	0	13.490	22.30	86.91	561.0	

50 rows × 33 columns

In [74]:

```
print(df.nunique()) # sutunlarda yeralan verilerin benzersiz sayıları
```

```
id                         569
diagnosis                    2
radius_mean                456
texture_mean               479
perimeter_mean             522
area_mean                  539
smoothness_mean            474
compactness_mean           537
concavity_mean             537
concave points_mean        542
symmetry_mean              432
fractal_dimension_mean     499
radius_se                  540
texture_se                 519
perimeter_se               533
area_se                    528
smoothness_se              547
compactness_se             541
concavity_se               533
concave points_se          507
symmetry_se                498
fractal_dimension_se       545
radius_worst               457
texture_worst              511
perimeter_worst            514
area_worst                 544
smoothness_worst           411
compactness_worst          529
concavity_worst            539
concave points_worst       492
symmetry_worst             500
fractal_dimension_worst    535
Unnamed: 32                  0
dtype: int64
```

In [67]:

```
df.shape # 33 adet features ve 569 satır
```

Out[67]:

```
(569, 31)
```

In [75]:

```
# id ve unnamed kolanları atılması gerekir.

df = df.drop(columns=["id","Unnamed: 32"])
```

In [76]:

```
df.shape  # 31 features
```

Out[76]:

```
(569, 31)
```

In [77]:

```
kopya_satir = df.duplicated()      # kopya varsa true dönecektir...
print(kopya_satir.any())

# df.drop_duplicates(inplace=True)   benzer satırları silmek için kullanılır
```

```
False
```

In [21]:

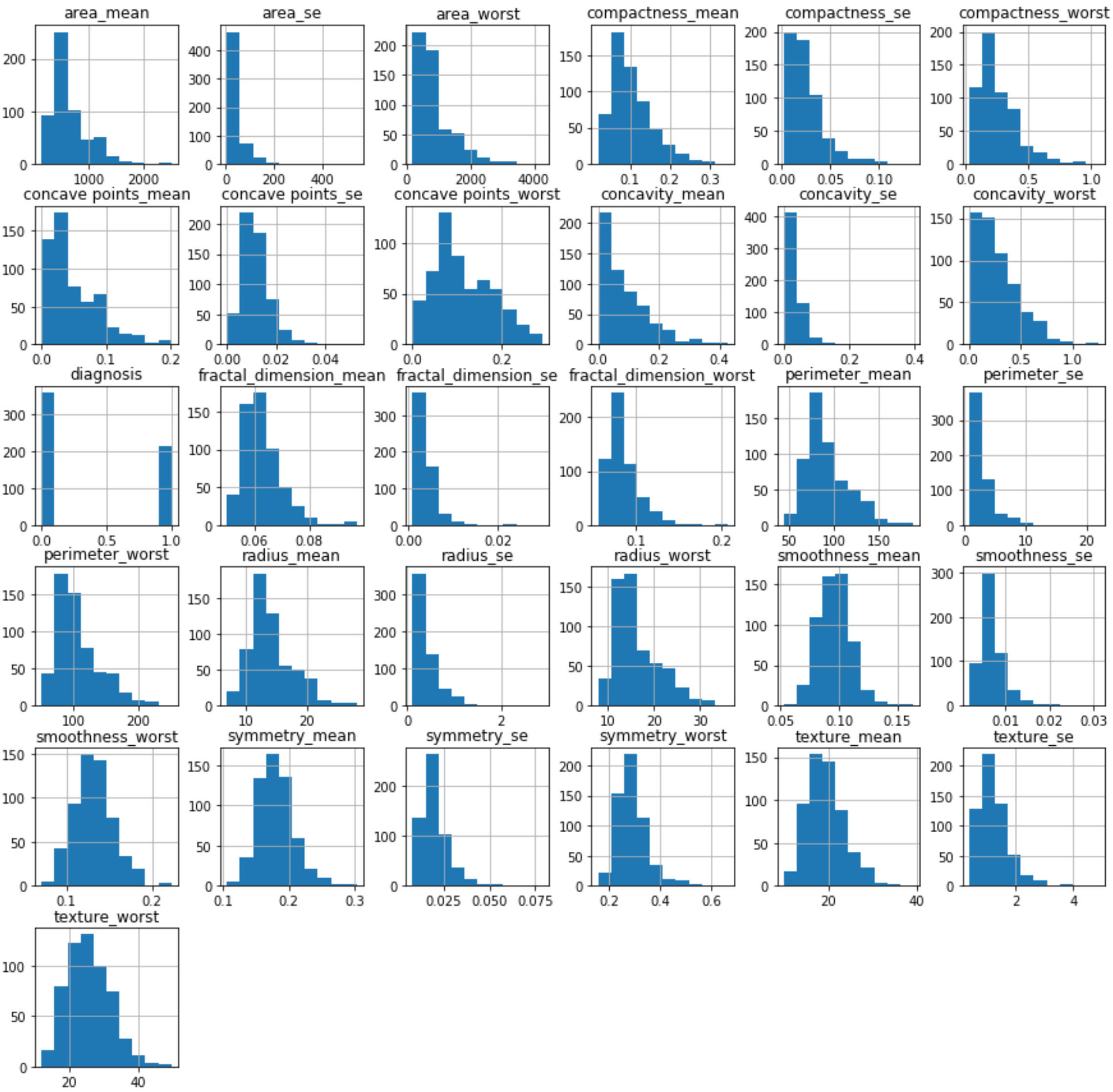
```
print(df.describe())  # dataset ile ilgili istatiksel bilgiler görüntülenir.
```

```
       diagnosis  radius_mean  texture_mean  perimeter_mean    area_mean  
\
count  569.000000   569.000000    569.000000      569.000000   569.000000  
mean     0.372583    14.127292     19.289649       91.969033   654.889104  
std      0.483918     3.524049      4.301036       24.298981   351.914129  
min      0.000000     6.981000      9.710000       43.790000   143.500000  
25%      0.000000    11.700000     16.170000       75.170000   420.300000  
50%      0.000000    13.370000     18.840000       86.240000   551.100000  
75%      1.000000    15.780000     21.800000      104.100000   782.700000  
max      1.000000    28.110000     39.280000      188.500000  2501.000000  

       smoothness_mean  compactness_mean  concavity_mean  concave points_m
ean  \
count       569.000000        569.000000      569.000000           569.000
000   
mean          0.096360          0.104341        0.088799             0.048
919   
std           0.014064          0.052813        0.079720             0.038
803   
min           0.052630          0.019380        0.000000             0.000
000   
25%           0.086370          0.064920        0.029560             0.020
310   
50%           0.095870          0.092630        0.061540             0.033
500   
75%           0.105300          0.130400        0.130700             0.074
000   
max           0.163400          0.345400        0.426800             0.201
200   

       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]
```

In [78]:

```
df.hist(figsize=(15,15))
pyplot.show()

# eğer çok fazla kolon normal dağılıma uymuyorsa transform işlemi yapılabilir.
```



In [79]:

```
dataset = df.values # kolon isimleri atılır.
```

In [80]:

```
dataset
```

Out[80]:

```
array([[ 1.      , 17.99    , 10.38    , ...,  0.2654 ,  0.4601 ,  0.1189 ],
       [ 1.      , 20.57    , 17.77    , ...,  0.186  ,  0.275  ,  0.08902],
       [ 1.      , 19.69    , 21.25    , ...,  0.243  ,  0.3613 ,  0.08758],
       ...,
       [ 1.      , 16.6     , 28.08    , ...,  0.1418 ,  0.2218 ,  0.0782 ],
       [ 1.      , 20.6     , 29.33    , ...,  0.265  ,  0.4087 ,  0.124  ],
       [ 0.      ,  7.76    , 24.54    , ...,  0.      ,  0.2871 ,  0.07039]])
```

In [25]:

```
dataset.shape
```

Out[25]:

```
(569, 31)
```

In [81]:

```
# kolon içierisinde yeralan değerler sayısal olarak büyük olması (örneğin : area_mean k
olonu)
#hesaplama esnasında nerönların sönmesine sebep olabilir.
# kolon değerlerini normalleştirme işlemi yapılır.

min_max_scaler = MinMaxScaler()              # 0 ile 1 arasıda değerleri trasnform ede
r
dataset = min_max_scaler.fit_transform(dataset)
```

In [82]:

```
dataset[:,4] # area_mean kolonu
```

Out[82]:

```
array([0.36373277, 0.50159067, 0.44941676, 0.10290562, 0.4892895 ,
       0.14150583, 0.38027572, 0.18426299, 0.15961824, 0.14099682,
       0.27753977, 0.27041357, 0.4154825 , 0.27113468, 0.18443266,
       0.218579  , 0.22948038, 0.27796394, 0.47359491, 0.17934252,
       0.15970308, 0.05531283, 0.23792153, 0.53467656, 0.32284199,
       0.32627784, 0.21264051, 0.40318134, 0.24979852, 0.34426299,
       0.40063627, 0.12602333, 0.32059385, 0.43202545, 0.28152704,
       0.30795334, 0.20763521, 0.16131495, 0.23554613, 0.17633086,
       0.17794274, 0.09654295, 0.40742312, 0.17039236, 0.16458112,
       0.39554613, 0.024772  , 0.16589608, 0.12971368, 0.17709438,
       0.12063627, 0.1816755 , 0.1247508 , 0.37730647, 0.24148462,
       0.1126193 , 0.42778367, 0.21777306, 0.16275716, 0.03435843,
       0.0714316 , 0.03321315, 0.21302227, 0.04979852, 0.15079533,
       0.2226087 , 0.05340403, 0.10629905, 0.04538706, 0.15227996,
       0.41845175, 0.0426299 , 0.33336161, 0.1868929 , 0.13887593,
       0.28598091, 0.17633086, 0.36585366, 0.46723224, 0.15389183,
       0.10943796, 0.15970308, 0.73573701, 0.41930011, 0.12716861,
       0.39512195, 0.21408271, 0.39554613, 0.13683987, 0.21565217,
       0.2202333 , 0.24801697, 0.17314952, 0.17459173, 0.23843054,
       0.47529162, 0.1304772 , 0.0640509 , 0.11414634, 0.2116649 ,
       0.18629905, 0.        , 0.13370095, 0.06566278, 0.08169671,
       0.16402969, 0.11410392, 0.13709438, 0.57921527, 0.10731707,
       0.06222694, 0.14290562, 0.20632025, 0.08089077, 0.03707317,
       0.12517497, 0.04313892, 0.22863203, 0.27109226, 0.35567338,
       0.11020148, 0.39597031, 0.68610817, 0.21090138, 0.17391304,
       0.18884411, 0.18201485, 0.42184517, 0.2252386 , 0.4447508 ,
       0.13247084, 0.25679745, 0.2826299 , 0.26222694, 0.39512195,
       0.15389183, 0.1188123 , 0.10871686, 0.22676564, 0.10235419,
       0.06150583, 0.28398727, 0.10795334, 0.15639449, 0.08984093,
       0.12271474, 0.1223754 , 0.23160127, 0.21064687, 0.18727466,
       0.15944857, 0.02562036, 0.06646872, 0.10112407, 0.16772004,
       0.13437964, 0.34791092, 0.31249205, 0.12941676, 0.09471898,
       0.11720042, 0.42990456, 0.45408271, 0.13616119, 0.6542948 ,
       0.2318982 , 0.09081654, 0.31507953, 0.35677625, 0.23007423,
       0.13599152, 0.17896076, 0.25170732, 0.09722163, 0.08742312,
       0.03550371, 0.06740191, 0.29242842, 0.16241782, 0.15495228,
       0.89353128, 0.495228  , 0.26430541, 0.10965005, 0.24055143,
       0.073807  , 0.38069989, 0.11741251, 0.12106045, 0.13582185,
       0.19783669, 0.15435843, 0.06133616, 0.14163309, 0.22392365,
       0.15817603, 0.18892895, 0.37348887, 0.42608696, 0.21174973,
       0.13467656, 0.34277837, 0.65387063, 0.19270414, 0.14354189,
       0.2430965 , 0.06443266, 0.32271474, 0.16369035, 0.24687169,
       0.48632025, 0.12067869, 0.99915164, 0.34125133, 0.19817603,
       0.18468717, 0.12246023, 0.07537646, 0.46086957, 0.45790032,
       0.18044539, 0.17722163, 0.07194062, 0.26205726, 0.17090138,
       0.21111347, 0.07893955, 0.22948038, 0.14969247, 0.15257688,
       0.31876988, 0.10697773, 0.10320255, 0.49862142, 0.05773065,
       0.19507953, 0.64750795, 0.4931071 , 0.20377519, 0.32962884,
       0.18316013, 0.14125133, 0.10430541, 0.18939555, 0.4290562 ,
       0.0823754 , 0.16886532, 0.15639449, 0.08632025, 0.11147402,
       0.51770944, 0.11194062, 0.45068929, 0.3328526 , 0.4349947 ,
       0.19465536, 0.45111347, 0.24169671, 0.26723224, 0.25510074,
       0.4854719 , 0.33493107, 0.34116649, 0.26091198, 0.33289502,
       0.54103924, 0.08606575, 0.17709438, 0.15639449, 0.08542948,
       0.20746554, 0.10371156, 0.57158006, 0.06209968, 0.36284199,
       0.12390244, 0.10735949, 0.40657476, 0.18188759, 0.18829268,
       0.42184517, 0.12038176, 0.42481442, 0.28063627, 0.15826087,
       0.14655355, 0.126193  , 0.15796394, 0.10629905, 0.10710498,
       0.21527041, 0.23066808, 0.15703075, 0.12267232, 0.1478685 ,
```

```
       0.18629905, 0.09340403, 0.12199364, 0.20767762, 0.08089077,
       0.45535525, 0.1390456 , 0.46808059, 0.08093319, 0.11011665,
       0.11609756, 0.16704136, 0.04360551, 0.17930011, 0.16419936,
       0.11673383, 0.22116649, 0.15295864, 0.11266172, 0.03295864,
       0.14341463, 0.13484624, 0.3747614 , 0.04284199, 0.14159067,
       0.07664899, 0.4795334 , 0.15325557, 0.47529162, 0.13336161,
       0.14693531, 0.20063627, 0.12831389, 0.2842842 , 0.28984093,
       0.27558855, 0.15715801, 0.10341463, 0.10455992, 0.13611877,
       0.32878049, 0.15728526, 0.40233298, 0.07096501, 0.68016967,
       0.21111347, 0.05811241, 0.09773065, 0.44559915, 0.11741251,
       0.07554613, 0.12801697, 0.22277837, 0.10994698, 0.12012725,
       0.11770944, 0.26091198, 0.79172853, 0.2430965 , 0.10226935,
       0.14519618, 0.15630965, 0.19096501, 0.04135737, 0.05730647,
       0.14778367, 0.17077413, 0.14977731, 0.29463415, 0.17344645,
       0.48759279, 0.46256628, 0.1335737 , 0.59490986, 0.56776246,
       0.29560976, 0.24106045, 0.52704136, 0.50540827, 0.184772  ,
       0.27359491, 0.0826299 , 0.17756098, 0.18540827, 0.09251326,
       0.10299046, 0.09722163, 0.12907741, 0.1354825 , 0.16895016,
       0.22108165, 0.13510074, 0.19219512, 0.10540827, 0.43711559,
       0.07554613, 0.03851538, 0.25501591, 0.5359491 , 0.12839873,
       0.19749735, 0.17586426, 0.15474019, 0.09955461, 0.12233298,
       0.36076352, 0.1269141 , 0.1619088 , 0.15444327, 0.13811241,
       0.09607635, 0.27847296, 0.1573701 , 0.35978791, 0.13683987,
       0.10871686, 0.09743372, 0.05314952, 0.23338282, 0.24432662,
       0.12313892, 0.05416755, 0.27978791, 0.14909862, 0.10044539,
       0.11291622, 0.21743372, 0.11227996, 0.18316013, 0.06201485,
       0.06948038, 0.08063627, 0.09179215, 0.10078473, 0.15177094,
       0.22969247, 0.13756098, 0.46935313, 0.40996819, 0.22489926,
       0.19342524, 0.15512195, 0.19838812, 0.19049841, 0.19639449,
       0.09671262, 0.3331071 , 0.18765642, 0.08373277, 0.35906681,
       0.12632025, 0.35550371, 0.22536585, 0.21896076, 0.526193  ,
       0.1223754 , 0.44432662, 0.12682927, 0.212386  , 0.14820785,
       0.1754825 , 0.11520679, 0.16729586, 0.15978791, 0.06252386,
       0.33399788, 1.        , 0.21319194, 0.11418876, 0.16704136,
       0.16941676, 0.16687169, 0.06057264, 0.35503712, 0.11253446,
       0.06176034, 0.12996819, 0.23049841, 0.13654295, 0.09136797,
       0.15414634, 0.20144221, 0.19338282, 0.11088017, 0.28517497,
       0.13225875, 0.19486744, 0.17085896, 0.18137858, 0.25607635,
       0.14133616, 0.22163309, 0.43414634, 0.11749735, 0.30290562,
       0.13700954, 0.35995758, 0.36627784, 0.14159067, 0.16763521,
       0.22795334, 0.14511135, 0.14277837, 0.3921527 , 0.4990456 ,
       0.23155885, 0.1918982 , 0.14116649, 0.65259809, 0.04462354,
       0.05471898, 0.13132556, 0.09459173, 0.28687169, 0.24933192,
       0.11983033, 0.2278685 , 0.17527041, 0.218579  , 0.23686108,
       0.10506893, 0.38536585, 0.45408271, 0.14829268, 0.14858961,
       0.04848356, 0.72004242, 0.10375398, 0.18133616, 0.06349947,
       0.03300106, 0.17289502, 0.1378579 , 0.19117709, 0.12797455,
       0.11851538, 0.11567338, 0.18324496, 0.49013786, 0.09420997,
       0.49395546, 0.20627784, 0.11151644, 0.01497349, 0.01141039,
       0.11003181, 0.21756098, 0.22273595, 0.16750795, 0.18718982,
       0.18226935, 0.07694592, 0.07520679, 0.06031813, 0.09251326,
       0.09204666, 0.09963945, 0.15457052, 0.05111347, 0.15728526,
       0.07546129, 0.07134677, 0.05420997, 0.2178579 , 0.11028632,
       0.193807  , 0.1028632 , 0.24322375, 0.51049841, 0.56648993,
       0.47401909, 0.30311771, 0.4757158 , 0.01590668])
```

In [83]:

```
# datasetinnin train ve test seti olarak ayıralım
train_size = int(len(dataset) * 0.80)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size,:], dataset[train_size:len(dataset),:]
print(len(train), len(test))
```

```
455 114
```

In [84]:

```
    # train setimizi Label larından ayırma işlemi gerçekleştirilir. (diagnosis ayrı bir dizi olacaktır.)

    dataX, dataY = [],[]

    for i in range(len(train)):
        a = train[i , 1:32]
        dataX.append(a)
        dataY.append(train[i ,0:1])

trainX = np.array(dataX)
trainY = np.array(dataY)
```

In [85]:

```
trainY[0:20]
```

Out[85]:

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

In [86]:

```
# diagnosis kolonu ayrılmışmı kontrol edelim.
trainX.shape
```

Out[86]:

```
(455, 30)
```

In [87]:

```
    dataX, dataY = [], []            # test setimizi label lardan ayırıyoruz (label = d
iagnosis)

    for i in range(len(test)):
        a = test[i , 1:32]
        dataX.append(a)
        dataY.append(test[i , 0:1])

testX = np.array(dataX)
testY =np.array(dataY)
```

In [88]:

```
testY[0:10]
```

Out[88]:

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

In [89]:

```
testX.shape
```

Out[89]:

```
(114, 30)
```

2. Model kurulumu ve Tahminleme

In [90]:

```
classifiers = [

    KNeighborsClassifier(3),

    SVC( C=1.0, kernel="rbf", gamma='auto', probability=True),

    DecisionTreeClassifier(),

    GaussianNB() ]
```

In [91]:

```
log_cols=["Classifier", "Accuracy", "Log Loss"]
log = pd.DataFrame(columns=log_cols)

for clf in classifiers:
    clf.fit(trainX, trainY)
    name = clf.__class__.__name__

    print("="*30)
    print(name)

    print('****Results****')
    train_predictions = clf.predict(testX)
    acc = accuracy_score(testY, train_predictions)
    print("Accuracy: {:.4%}".format(acc))



print("*"*39)
```

```
==============================
KNeighborsClassifier
****Results****
Accuracy: 94.7368%
==============================
SVC
****Results****
Accuracy: 97.3684%
==============================
DecisionTreeClassifier
****Results****
Accuracy: 86.8421%
==============================
GaussianNB
****Results****
Accuracy: 92.1053%
***************************************

C:\Users\enginseven\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: D
ataConversionWarning: A column-vector y was passed when a 1d array was exp
ected. Please change the shape of y to (n_samples, ), for example using ra
vel().
  """
C:\Users\enginseven\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:724: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
C:\Users\enginseven\Anaconda3\lib\site-packages\sklearn\utils\validation.p
y:724: DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples, ), for example u
sing ravel().
  y = column_or_1d(y, warn=True)
```

3. Yapay Sinir Ağları Model kurulum ve Tahminleme

In [112]:

```
############################################################

###################### Nerual Networks ####################


model = Sequential()
model.add(Dense(64, input_dim=30, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [113]:

```
#  opt = keras.optimizers.Adam(learning_rate=0.0007)  learning rate manuel ayarlama içi
n kullanılabilir.
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy']) # lea
rninRate varsayılan = 0.001
```

In [114]:

```
# fit model
history = model.fit(trainX, trainY, epochs=50, batch_size=10, verbose = 1)
```

```
Epoch 1/50
455/455 [==============================] - 1s 1ms/step - loss: 0.6522 - ac
curacy: 0.8198
Epoch 2/50
455/455 [==============================] - 0s 189us/step - loss: 0.4828 -
accuracy: 0.8989
Epoch 3/50
455/455 [==============================] - 0s 198us/step - loss: 0.2805 -
accuracy: 0.9143
Epoch 4/50
455/455 [==============================] - 0s 213us/step - loss: 0.1916 -
accuracy: 0.9275
Epoch 5/50
455/455 [==============================] - 0s 209us/step - loss: 0.1553 -
accuracy: 0.9407
Epoch 6/50
455/455 [==============================] - 0s 204us/step - loss: 0.1458 -
accuracy: 0.9407
Epoch 7/50
455/455 [==============================] - 0s 218us/step - loss: 0.1163 -
accuracy: 0.9560
Epoch 8/50
455/455 [==============================] - 0s 202us/step - loss: 0.1044 -
accuracy: 0.9626
Epoch 9/50
455/455 [==============================] - 0s 202us/step - loss: 0.1022 -
accuracy: 0.9582
Epoch 10/50
455/455 [==============================] - 0s 204us/step - loss: 0.0881 -
accuracy: 0.9648
Epoch 11/50
455/455 [==============================] - 0s 200us/step - loss: 0.0872 -
accuracy: 0.9626
Epoch 12/50
455/455 [==============================] - 0s 209us/step - loss: 0.0805 -
accuracy: 0.9692
Epoch 13/50
455/455 [==============================] - 0s 198us/step - loss: 0.0809 -
accuracy: 0.9626
Epoch 14/50
455/455 [==============================] - 0s 226us/step - loss: 0.0775 -
accuracy: 0.9736
Epoch 15/50
455/455 [==============================] - 0s 204us/step - loss: 0.0617 -
accuracy: 0.9780
Epoch 16/50
455/455 [==============================] - ETA: 0s - loss: 0.0849 - accura
cy: 0.95 - 0s 198us/step - loss: 0.0723 - accuracy: 0.9692
Epoch 17/50
455/455 [==============================] - 0s 259us/step - loss: 0.0670 -
accuracy: 0.9780
Epoch 18/50
455/455 [==============================] - 0s 218us/step - loss: 0.0747 -
accuracy: 0.9758
Epoch 19/50
455/455 [==============================] - 0s 200us/step - loss: 0.0610 -
accuracy: 0.9802
Epoch 20/50
455/455 [==============================] - 0s 215us/step - loss: 0.0678 -
accuracy: 0.9736
Epoch 21/50
```

```
455/455 [==============================] - 0s 213us/step - loss: 0.0830 -
accuracy: 0.9670
Epoch 22/50
455/455 [==============================] - 0s 200us/step - loss: 0.0576 -
accuracy: 0.9780
Epoch 23/50
455/455 [==============================] - 0s 200us/step - loss: 0.0588 -
accuracy: 0.9780
Epoch 24/50
455/455 [==============================] - 0s 242us/step - loss: 0.0571 -
accuracy: 0.9736
Epoch 25/50
455/455 [==============================] - 0s 200us/step - loss: 0.0645 -
accuracy: 0.9824
Epoch 26/50
455/455 [==============================] - 0s 198us/step - loss: 0.0541 -
accuracy: 0.9824
Epoch 27/50
455/455 [==============================] - 0s 224us/step - loss: 0.0567 -
accuracy: 0.9780
Epoch 28/50
455/455 [==============================] - 0s 198us/step - loss: 0.0544 -
accuracy: 0.9780
Epoch 29/50
455/455 [==============================] - 0s 202us/step - loss: 0.0820 -
accuracy: 0.9736
Epoch 30/50
455/455 [==============================] - 0s 200us/step - loss: 0.0654 -
accuracy: 0.9714
Epoch 31/50
455/455 [==============================] - 0s 211us/step - loss: 0.0611 -
accuracy: 0.9802
Epoch 32/50
455/455 [==============================] - 0s 209us/step - loss: 0.0491 -
accuracy: 0.9780
Epoch 33/50
455/455 [==============================] - 0s 215us/step - loss: 0.0650 -
accuracy: 0.9736
Epoch 34/50
455/455 [==============================] - 0s 207us/step - loss: 0.0639 -
accuracy: 0.9758
Epoch 35/50
455/455 [==============================] - 0s 209us/step - loss: 0.0544 -
accuracy: 0.9824
Epoch 36/50
455/455 [==============================] - 0s 220us/step - loss: 0.0568 -
accuracy: 0.9802
Epoch 37/50
455/455 [==============================] - 0s 213us/step - loss: 0.0473 -
accuracy: 0.9802
Epoch 38/50
455/455 [==============================] - 0s 165us/step - loss: 0.0639 -
accuracy: 0.9758
Epoch 39/50
455/455 [==============================] - 0s 185us/step - loss: 0.0500 -
accuracy: 0.9780
Epoch 40/50
455/455 [==============================] - 0s 237us/step - loss: 0.0480 -
accuracy: 0.9802
Epoch 41/50
455/455 [==============================] - 0s 255us/step - loss: 0.0553 -
```

```
accuracy: 0.9802
Epoch 42/50
455/455 [==============================] - 0s 286us/step - loss: 0.0484 -
accuracy: 0.9802
Epoch 43/50
455/455 [==============================] - 0s 266us/step - loss: 0.0482 -
accuracy: 0.9802
Epoch 44/50
455/455 [==============================] - 0s 273us/step - loss: 0.0522 -
accuracy: 0.9802
Epoch 45/50
455/455 [==============================] - 0s 251us/step - loss: 0.0503 -
accuracy: 0.9824
Epoch 46/50
455/455 [==============================] - 0s 264us/step - loss: 0.0882 -
accuracy: 0.9560
Epoch 47/50
455/455 [==============================] - 0s 306us/step - loss: 0.0617 -
accuracy: 0.9758
Epoch 48/50
455/455 [==============================] - 0s 275us/step - loss: 0.0552 -
accuracy: 0.9758
Epoch 49/50
455/455 [==============================] - 0s 253us/step - loss: 0.0497 -
accuracy: 0.9846
Epoch 50/50
455/455 [==============================] - 0s 275us/step - loss: 0.0514 -
accuracy: 0.9802
```

In [115]:

```
testPredict = model.predict_classes(testX)
```

In [116]:

```
acc = accuracy_score(testY, testPredict)
print(acc)
```

```
0.9824561403508771
```

In [117]:

```
precision = precision_score(testY, testPredict, average='binary')


recall = recall_score(testY, testPredict, average='binary')

f1score = f1_score(testY, testPredict, average='binary')

results = confusion_matrix(testY, testPredict)
```

In [118]:

```
f1score
```

Out[118]:

```
0.9615384615384616
```

In [119]:

```
results
```

Out[119]:

```
array([[87,  1],
       [ 1, 25]], dtype=int64)
```

In [120]:

```
recall
```

Out[120]:

```
0.9615384615384616
```

In [122]:

```
precision
```

Out[122]:

```
0.9615384615384616
```

In [ ]:

```

```