Comparative Analysis of Machine Learning Algorithms for Breast Cancer Detection

In [ ]:

```
# import libraries
```

In [1]:

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

Load the breast cancer dataset ¶

In [2]:

```
from sklearn.datasets import load_breast_cancer
cancer_dataset = load_breast_cancer()
```

In [3]:

```
cancer_dataset
```

Out[3]:

```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 1,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
        1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
        1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
        1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
        0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
        1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
        0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
        1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
        1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
        0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
        0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
        1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
        1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
        1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
        1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
        1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
        1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnosti
c) dataset\n--------------------------------------------\n\n**Data Set Chara
cteristics:**\n\n    :Number of Instances: 569\n\n    :Number of Attributes:
30 numeric, predictive attributes and the class\n\n    :Attribute Informatio
n:\n        - radius (mean of distances from center to points on the perimet
er)\n        - texture (standard deviation of gray-scale values)\n        -
perimeter\n        - area\n        - smoothness (local variation in radius l
engths)\n        - compactness (perimeter^2 / area - 1.0)\n        - concavi
ty (severity of concave portions of the contour)\n        - concave points
(number of concave portions of the contour)\n        - symmetry\n        - f
ractal dimension ("coastline approximation" - 1)\n\n        The mean, standa
rd error, and "worst" or largest (mean of the three\n        worst/largest v
alues) of these features were computed for each image,\n        resulting in
```

```
30 features.  For instance, field 0 is Mean Radius, field\n        10 is Rad
ius SE, field 20 is Worst Radius.\n\n        - class:\n                - WDB
C-Malignant\n                - WDBC-Benign\n\n    :Summary Statistics:\n\n
===================================== ====== ======\n
Min    Max\n    ===================================== ====== ======\n    rad
ius (mean):                        6.981  28.11\n    texture (mean):
9.71   39.28\n    perimeter (mean):                     43.79  188.5\n    ar
ea (mean):                          143.5  2501.0\n    smoothness (mean):
0.053  0.163\n    compactness (mean):                   0.019  0.345\n    co
ncavity (mean):                     0.0    0.427\n    concave points (mean):
0.0    0.201\n    symmetry (mean):                      0.106  0.304\n    fr
actal dimension (mean):             0.05   0.097\n    radius (standard erro
r):              0.112  2.873\n    texture (standard error):            0.3
6   4.885\n    perimeter (standard error):          0.757  21.98\n    area
(standard error):                6.802  542.2\n    smoothness (standard erro
r):          0.002  0.031\n    compactness (standard error):        0.002
0.135\n    concavity (standard error):           0.0    0.396\n    concave p
oints (standard error):      0.0    0.053\n    symmetry (standard error):
0.008  0.079\n    fractal dimension (standard error):   0.001  0.03\n    rad
ius (worst):                       7.93   36.04\n    texture (worst):
12.02  49.54\n    perimeter (worst):                    50.41  251.2\n    ar
ea (worst):                         185.2  4254.0\n    smoothness (worst):
0.071  0.223\n    compactness (worst):                  0.027  1.058\n    co
ncavity (worst):                    0.0    1.252\n    concave points (wors
t):              0.0    0.291\n    symmetry (worst):                    0.
156  0.664\n    fractal dimension (worst):           0.055  0.208\n    ====
================================= ====== ======\n\n    :Missing Attribute Va
lues: None\n\n    :Class Distribution: 212 - Malignant, 357 - Benign\n\n
:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian\n\n
:Donor: Nick Street\n\n    :Date: November, 1995\n\nThis is a copy of UCI ML
Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://goo.gl/U2Uwz2\n\nFea
tures are computed from a digitized image of a fine needle\naspirate (FNA) o
f a breast mass.  They describe\ncharacteristics of the cell nuclei present
in the image.\n\nSeparating plane described above was obtained using\nMultis
urface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via
Linear Programming." Proceedings of the 4th\nMidwest Artificial Intelligence
and Cognitive Science Society,\npp. 97-101, 1992], a classification method w
hich uses linear\nprogramming to construct a decision tree.  Relevant featur
es\nwere selected using an exhaustive search in the space of 1-4\nfeatures a
nd 1-3 separating planes.\n\nThe actual linear program used to obtain the se
parating plane\nin the 3-dimensional space is that described in:\n[K. P. Ben
nett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Tw
o Linearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23
-34].\n\nThis database is also available through the UW CS ftp server:\n\nft
p ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topi
c:: References\n\n   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nucle
ar feature extraction \n     for breast tumor diagnosis. IS&T/SPIE 1993 Inte
rnational Symposium on \n     Electronic Imaging: Science and Technology, vo
lume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L. Mangasarian,
W.N. Street and W.H. Wolberg. Breast cancer diagnosis and \n     prognosis v
ia linear programming. Operations Research, 43(4), pages 570-577, \n     Jul
y-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machin
e learning techniques\n     to diagnose breast cancer from fine-needle aspir
ates. Cancer Letters 77 (1994) \n     163-171.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'm
ean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',
        'worst perimeter', 'worst area', 'worst smoothness',
        'worst compactness', 'worst concavity', 'worst concave points',
        'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'C:\\Users\\sharm\\anaconda3\\lib\\site-packages\\sklearn\\data
sets\\data\\breast_cancer.csv'}
```

In [4]:

```
type(cancer_dataset)
```

Out[4]:

```
sklearn.utils.Bunch
```

Keys in the dataset

In [6]:

```
cancer_dataset.keys()
```

Out[6]:

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_name
s', 'filename'])
```

feature of each cells in numeric format

In [7]:

```
cancer_dataset['data']
```

Out[7]:

```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]])
```

In [8]:

```
type(cancer_dataset['data'])
```

Out[8]:

```
numpy.ndarray
```

malignant or benign value

In [9]:

```
cancer_dataset['target']
```

Out[9]:

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

target value name malignant or benign tumor

In [10]:

```
cancer_dataset['target_names']
```

Out[10]:

```
array(['malignant', 'benign'], dtype='<U9')
```

description of the data

In [11]:

```
cancer_dataset['DESCR']
```

Out[11]:

```
'.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset
\n--------------------------------------------\n\n**Data Set Characteristic
s:**\n\n    :Number of Instances: 569\n\n    :Number of Attributes: 30 numer
ic, predictive attributes and the class\n\n    :Attribute Information:\n
- radius (mean of distances from center to points on the perimeter)\n
- texture (standard deviation of gray-scale values)\n        - perimeter\n
- area\n        - smoothness (local variation in radius lengths)\n        -
compactness (perimeter^2 / area - 1.0)\n        - concavity (severity of con
cave portions of the contour)\n        - concave points (number of concave p
ortions of the contour)\n        - symmetry\n        - fractal dimension ("c
oastline approximation" - 1)\n\n        The mean, standard error, and "wors
t" or largest (mean of the three\n        worst/largest values) of these fea
tures were computed for each image,\n        resulting in 30 features.  For
instance, field 0 is Mean Radius, field\n        10 is Radius SE, field 20 i
s Worst Radius.\n\n        - class:\n                - WDBC-Malignant\n
- WDBC-Benign\n\n    :Summary Statistics:\n\n    ===========================
========== ====== ======\n                                           Min
Max\n    ===================================== ====== ======\n    radius (me
an):                        6.981  28.11\n    texture (mean):
9.71   39.28\n    perimeter (mean):                     43.79  188.5\n    ar
ea (mean):                          143.5  2501.0\n    smoothness (mean):
0.053  0.163\n    compactness (mean):                   0.019  0.345\n    co
ncavity (mean):                     0.0    0.427\n    concave points (mean):
0.0    0.201\n    symmetry (mean):                      0.106  0.304\n    fr
actal dimension (mean):             0.05   0.097\n    radius (standard erro
r):              0.112  2.873\n    texture (standard error):             0.3
6   4.885\n    perimeter (standard error):           0.757  21.98\n    area
(standard error):                6.802  542.2\n    smoothness (standard erro
r):          0.002  0.031\n    compactness (standard error):         0.002
0.135\n    concavity (standard error):           0.0    0.396\n    concave p
oints (standard error):      0.0    0.053\n    symmetry (standard error):
0.008  0.079\n    fractal dimension (standard error):   0.001  0.03\n    rad
ius (worst):                       7.93   36.04\n    texture (worst):
12.02  49.54\n    perimeter (worst):                    50.41  251.2\n    ar
ea (worst):                         185.2  4254.0\n    smoothness (worst):
0.071  0.223\n    compactness (worst):                  0.027  1.058\n    co
ncavity (worst):                    0.0    1.252\n    concave points (wors
t):               0.0    0.291\n    symmetry (worst):                     0.
156  0.664\n    fractal dimension (worst):            0.055  0.208\n    ====
================================= ====== ======\n\n    :Missing Attribute Va
lues: None\n\n    :Class Distribution: 212 - Malignant, 357 - Benign\n\n
:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian\n\n
:Donor: Nick Street\n\n    :Date: November, 1995\n\nThis is a copy of UCI ML
Breast Cancer Wisconsin (Diagnostic) datasets.\nhttps://goo.gl/U2Uwz2\n\nFea
tures are computed from a digitized image of a fine needle\naspirate (FNA) o
f a breast mass.  They describe\ncharacteristics of the cell nuclei present
in the image.\n\nSeparating plane described above was obtained using\nMultis
urface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via
Linear Programming." Proceedings of the 4th\nMidwest Artificial Intelligence
and Cognitive Science Society,\npp. 97-101, 1992], a classification method w
hich uses linear\nprogramming to construct a decision tree.  Relevant featur
es\nwere selected using an exhaustive search in the space of 1-4\nfeatures a
nd 1-3 separating planes.\n\nThe actual linear program used to obtain the se
parating plane\nin the 3-dimensional space is that described in:\n[K. P. Ben
nett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Tw
o Linearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 23
-34].\n\nThis database is also available through the UW CS ftp server:\n\nft
p ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/\n\n.. topi
c:: References\n\n   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nucle
ar feature extraction \n     for breast tumor diagnosis. IS&T/SPIE 1993 Inte
rnational Symposium on \n     Electronic Imaging: Science and Technology, vo
lume 1905, pages 861-870,\n     San Jose, CA, 1993.\n   - O.L. Mangasarian,
W.N. Street and W.H. Wolberg. Breast cancer diagnosis and \n     prognosis v
ia linear programming. Operations Research, 43(4), pages 570-577, \n     Jul
y-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machin
e learning techniques\n     to diagnose breast cancer from fine-needle aspir
ates. Cancer Letters 77 (1994) \n     163-171.'
```

name of features

In [12]:

```
cancer_dataset['feature_names']
```

Out[12]:

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

location/path of data file

In [13]:

```
cancer_dataset['filename']
```

Out[13]:

```
'C:\\Users\\sharm\\anaconda3\\lib\\site-packages\\sklearn\\datasets\\data\\b
reast_cancer.csv'
```

create dataframe

In [16]:

```
cancer_df = pd.DataFrame(np.c_[cancer_dataset['data'],cancer_dataset['target']],
            columns = np.append(cancer_dataset['feature_names'], ['target']))
```

dataframe to CSV file

In [17]:

```
cancer_df.to_csv('breast_cancer_dataframe.csv')
```

Head of cancer Dataframe

In [18]:

```
cd = cancer_df.head(6)
```

In [19]:

```
cd
```

Out[19]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809
5	12.45	15.70	82.57	477.1	0.12780	0.17000	0.1578	0.08089	0.2087

6 rows × 31 columns

Tail of Cancer Dataframe

In [20]:

```
cancer_df.tail(6)
```

Out[20]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	me symme
563	20.92	25.09	143.00	1347.0	0.10990	0.22360	0.31740	0.14740	0.21
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.17
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.17
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.15
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.23
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.15

6 rows × 31 columns

Information of Cancer Dataframe

In [21]:

```
cancer_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    float64
dtypes: float64(31)
memory usage: 137.9 KB
```

Numeric Distribution of data

In [22]:

```
cancer_df.describe()
```

Out[22]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800

8 rows × 31 columns

In [23]:

```
cancer_df.isnull().sum()
```

Out[23]:

```
mean radius                0
mean texture               0
mean perimeter             0
mean area                  0
mean smoothness            0
mean compactness           0
mean concavity             0
mean concave points        0
mean symmetry              0
mean fractal dimension     0
radius error               0
texture error              0
perimeter error            0
area error                 0
smoothness error           0
compactness error          0
concavity error            0
concave points error       0
symmetry error             0
fractal dimension error    0
worst radius               0
worst texture              0
worst perimeter            0
worst area                 0
worst smoothness           0
worst compactness          0
worst concavity            0
worst concave points       0
worst symmetry             0
worst fractal dimension    0
target                     0
dtype: int64
```

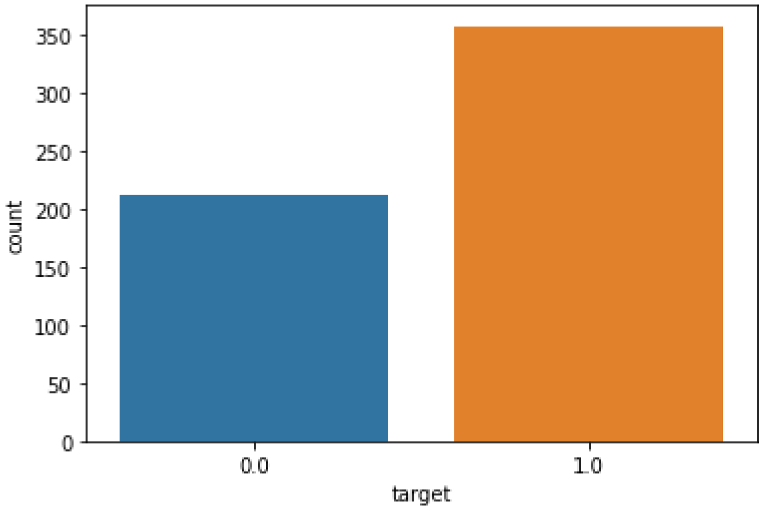
Count the target class

In [27]:

```
print(sns.countplot(cancer_df['target']))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```

```
C:\Users\sharm\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



Counter Plot of feature mean radius

In [28]:

```
plt.figure(figsize = (20,8))
```

Out[28]:

```
<Figure size 1440x576 with 0 Axes>
```

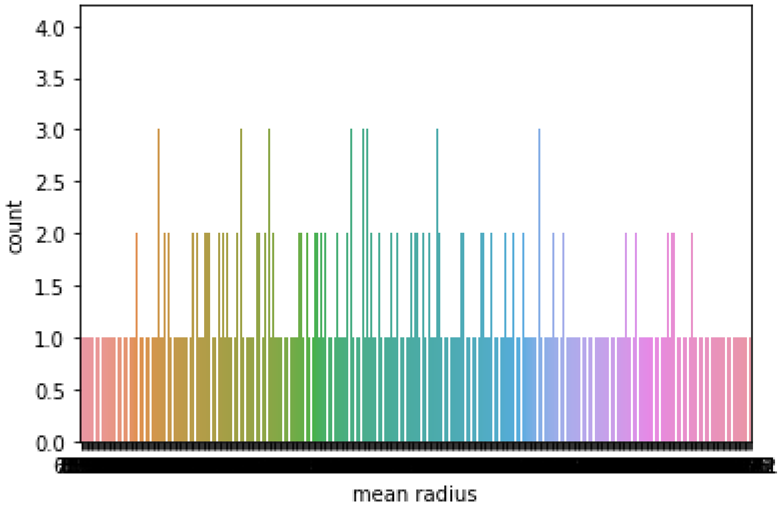
```
<Figure size 1440x576 with 0 Axes>
```

In [29]:

```
print(sns.countplot(cancer_df['mean radius']))
```

```
C:\Users\sharm\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variable as a keyword arg: x. From version 0.12,
the only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



Heatmap of the Dataframe

In [30]:

```
plt.figure(figsize = (16,9))
```

Out[30]:

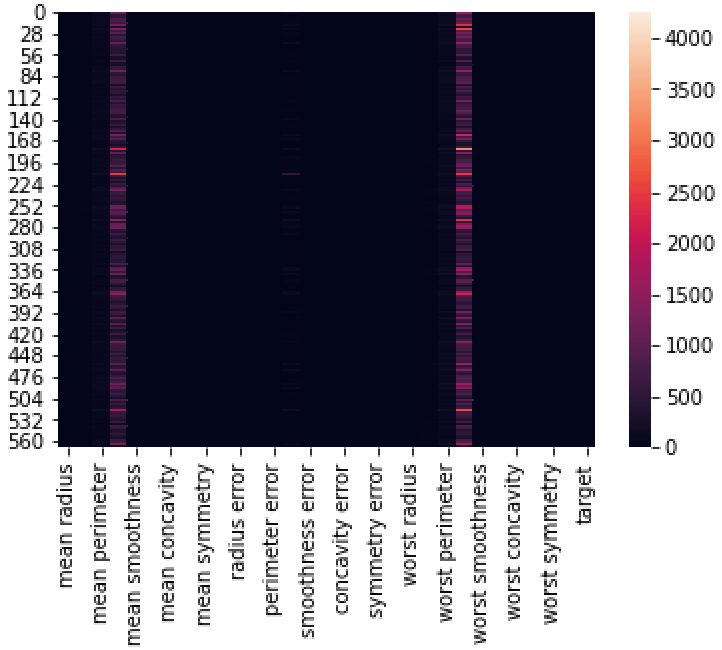
```
<Figure size 1152x648 with 0 Axes>
```

```
<Figure size 1152x648 with 0 Axes>
```

In [31]:

```
print(sns.heatmap(cancer_df))
```

```
AxesSubplot(0.125,0.125;0.62x0.755)
```



Heatmap of a corelation matrix

In [32]:

```
plt.figure(figsize = (20,20))
```

Out[32]:

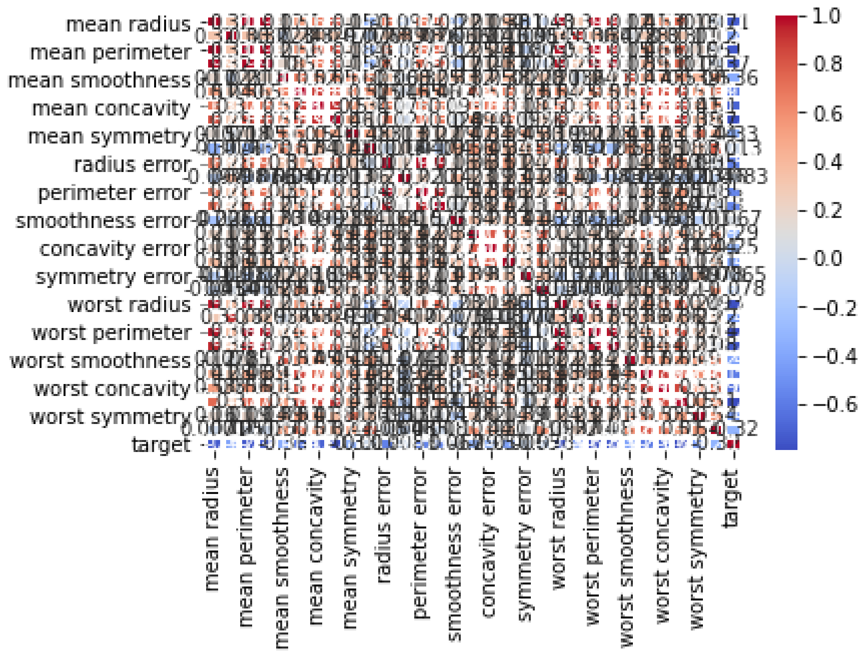
```
<Figure size 1440x1440 with 0 Axes>
```

```
<Figure size 1440x1440 with 0 Axes>
```

In [37]:

```
print(sns.heatmap(cancer_df.corr(), annot = True, cmap = 'coolwarm', linewidths = 2))
```

```
AxesSubplot(0.125,0.125;0.62x0.755)
```



Correlation Barplot

create second DataFrame by droping target

Taking the correlation of each feature with the target and the visualize barplot

In [38]:

```
cancer_df2 = cancer_df.drop(['target'], axis = 1)
print("The shape of 'cancer_df2' is : ", cancer_df2.shape)
```

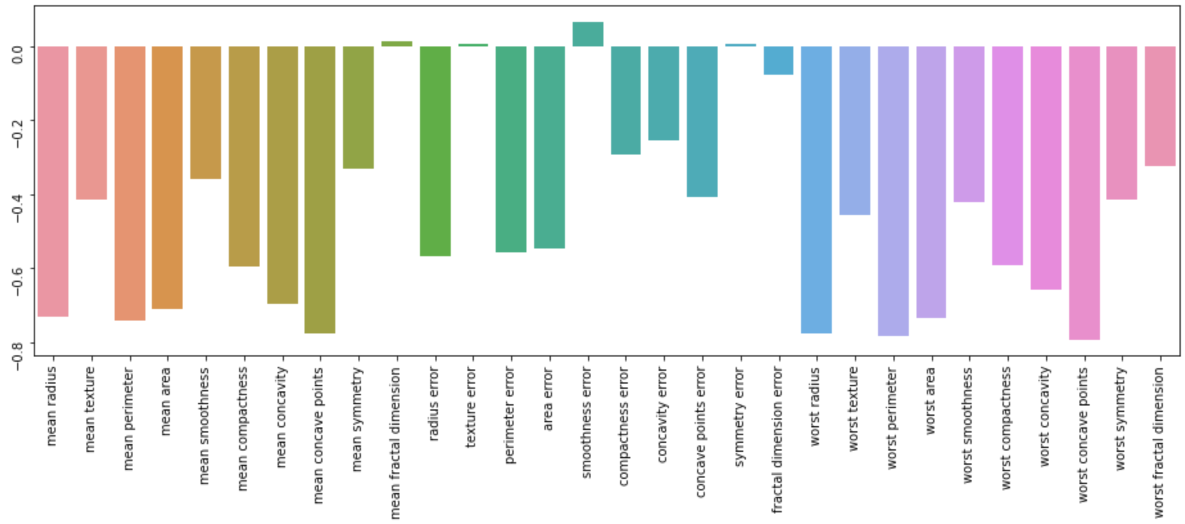
```
The shape of 'cancer_df2' is :  (569, 30)
```

visualize correlation barplot

In [39]:

```
plt.figure(figsize = (16,5))
ax = sns.barplot(cancer_df2.corrwith(cancer_df.target).index, cancer_df2.corrwith(cancer_df
ax.tick_params(labelrotation = 90)
```

```
C:\Users\sharm\anaconda3\lib\site-packages\seaborn\_decorators.py:36: Future
Warning: Pass the following variables as keyword args: x, y. From version 0.
12, the only valid positional argument will be `data`, and passing other arg
uments without an explicit keyword will result in an error or misinterpretat
ion.
  warnings.warn(
```



Input Variable

In [40]:

```
X = cancer_df.drop(['target'], axis = 1)
print(X.head(6))
```

```
   mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
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
5        12.45         15.70           82.57      477.1          0.12780

   mean compactness  mean concavity  mean concave points  mean symmetry  \
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
5           0.17000          0.1578              0.08089         0.2087

   mean fractal dimension  ...  worst radius  worst texture  worst perimeter
\
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
5                 0.07613  ...         15.47          23.75           103.40

   worst area  worst smoothness  worst compactness  worst concavity  \
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
5       741.6            0.1791             0.5249           0.5355

   worst concave points  worst symmetry  worst fractal dimension
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                0.1741          0.3985                  0.12440

[6 rows x 30 columns]
```

output variable

In [41]:

```
y = cancer_df['target']
print(y.head(6))
```

```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
5    0.0
Name: target, dtype: float64
```

split dataset into train and test

In [42]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state= 5)
```

Feature scaling

In [43]:

```
from sklearn.preprocessing import StandardScaler
```

In [44]:

```
sc = StandardScaler()
X_train_sc = sc.fit_transform(X_train)
X_test_sc = sc.transform(X_test)
```

In [45]:

```
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
```

In [47]:

```
print("Support vector classifier")
from sklearn.svm import SVC
svc_classifier = SVC()                        # Make an instance of the Model
svc_classifier.fit(X_train, y_train)          # Training the model on the entire data, stori
y_pred_scv = svc_classifier.predict(X_test)  # Make predictions on entire test data
print(accuracy_score(y_test, y_pred_scv))

# Train with Standard scaled Data
svc_classifier2 = SVC()
svc_classifier2.fit(X_train_sc, y_train)
y_pred_svc_sc = svc_classifier2.predict(X_test_sc)
print(accuracy_score(y_test, y_pred_svc_sc))

print("\n")
```

```
Support vector classifier
0.9649122807017544
0.9649122807017544
```

In [48]:

```
print("Logistic Regression")
from sklearn.linear_model import LogisticRegression
lr_classifier = LogisticRegression(random_state = 51, penalty = 'l2')
lr_classifier.fit(X_train, y_train)
y_pred_lr = lr_classifier.predict(X_test)
print(accuracy_score(y_test, y_pred_lr))

# Train with Standard scaled Data
lr_classifier2 = LogisticRegression(random_state = 51, penalty = 'l2')
lr_classifier2.fit(X_train_sc, y_train)
y_pred_lr_sc = lr_classifier.predict(X_test_sc)
print(accuracy_score(y_test, y_pred_lr_sc))

print("\n")
```

```
Logistic Regression
0.9649122807017544
0.631578947368421



C:\Users\sharm\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.p
y:762: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scik
it-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regre
ssion (https://scikit-learn.org/stable/modules/linear_model.html#logistic-re
gression)
  n_iter_i = _check_optimize_result(
```

In [49]:

```
print("K - Nearest Neighbor Classifier")
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
knn_classifier.fit(X_train, y_train)
y_pred_knn = knn_classifier.predict(X_test)
print(accuracy_score(y_test, y_pred_knn))

# Train with Standard scaled Data
knn_classifier2 = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
knn_classifier2.fit(X_train_sc, y_train)
y_pred_knn_sc = knn_classifier.predict(X_test_sc)
print(accuracy_score(y_test, y_pred_knn_sc))

print("\n")
```

```
K - Nearest Neighbor Classifier
0.9649122807017544
0.6666666666666666
```

In [50]:

```
print("Naive Bayes Classifier")
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred_nb = nb_classifier.predict(X_test)
print(accuracy_score(y_test, y_pred_nb))

# Train with Standard scaled Data
nb_classifier2 = GaussianNB()
nb_classifier2.fit(X_train_sc, y_train)
y_pred_nb_sc = nb_classifier2.predict(X_test_sc)
print(accuracy_score(y_test, y_pred_nb_sc))

print("\n")
```

```
Naive Bayes Classifier
0.9298245614035088
0.9122807017543859
```

In [51]:

```
print("Decision Tree Classifier")
from sklearn.tree import DecisionTreeClassifier
dt_classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 51)
dt_classifier.fit(X_train, y_train)
y_pred_dt = dt_classifier.predict(X_test)
print(accuracy_score(y_test, y_pred_dt))

# Train with Standard scaled Data
dt_classifier2 = DecisionTreeClassifier(criterion = 'entropy', random_state = 51)
dt_classifier2.fit(X_train_sc, y_train)
y_pred_dt_sc = dt_classifier.predict(X_test_sc)
print(accuracy_score(y_test, y_pred_dt_sc))

print("\n")
```

```
Decision Tree Classifier
0.9649122807017544
0.7894736842105263
```

In [52]:

```
print("Random Forest Classifier")
from sklearn.ensemble import RandomForestClassifier
rf_classifier = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', random_sta
rf_classifier.fit(X_train, y_train)
y_pred_rf = rf_classifier.predict(X_test)
print(accuracy_score(y_test, y_pred_rf))

# Train with Standard scaled Data
rf_classifier2 = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', random_st
rf_classifier2.fit(X_train_sc, y_train)
y_pred_rf_sc = rf_classifier.predict(X_test_sc)
print(accuracy_score(y_test, y_pred_rf_sc))

print("\n")
```

```
Random Forest Classifier
0.9824561403508771
0.8070175438596491
```

In [53]:

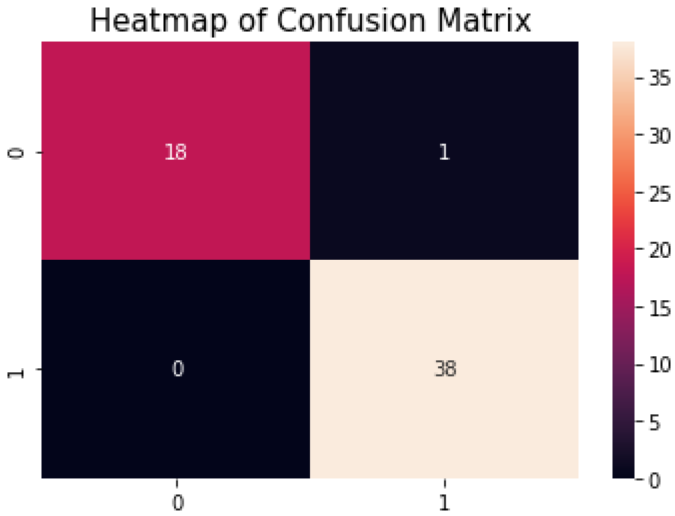
```
# XGBoost Classifier
print("XGB Classifier")
import xgboost as xgb
from xgboost import XGBClassifier
xgb_classifier = XGBClassifier()
xgb_classifier.fit(X_train, y_train, )
y_pred_xgb = xgb_classifier.predict(X_test)
print(accuracy_score(y_test, y_pred_xgb))


# Train with Standard scaled Data
xgb_classifier2 = XGBClassifier()
xgb_classifier2.fit(X_train_sc, y_train)
y_pred_xgb_sc = xgb_classifier2.predict(X_test_sc)
print(accuracy_score(y_test, y_pred_xgb_sc))
```

```
XGB Classifier

C:\Users\sharm\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarni
ng: The use of label encoder in XGBClassifier is deprecated and will be remo
ved in a future release. To remove this warning, do the following: 1) Pass o
ption use_label_encoder=False when constructing XGBClassifier object; and 2)
Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_
class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[14:43:24] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
0.9824561403508771
[14:43:24] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
0.9824561403508771
```

Confusion Matrix

In [54]:

```
cm = confusion_matrix(y_test, y_pred_xgb)
plt.title('Heatmap of Confusion Matrix', fontsize = 15)
sns.heatmap(cm, annot = True)
plt.show()


print("\n")
```



In [55]:

```
print("Support Vector Machine");
print(classification_report(y_test, y_pred_svc_sc))
print("\n")
```

```
Support Vector Machine
              precision    recall  f1-score   support

         0.0       0.95      0.95      0.95        19
         1.0       0.97      0.97      0.97        38

    accuracy                           0.96        57
   macro avg       0.96      0.96      0.96        57
weighted avg       0.96      0.96      0.96        57
```

In [56]:

```
print("Logistic Regression")
print(classification_report(y_test, y_pred_lr_sc))
print("\n")
```

```
Logistic Regression
              precision    recall  f1-score   support

         0.0       0.45      0.47      0.46        19
         1.0       0.73      0.71      0.72        38

    accuracy                           0.63        57
   macro avg       0.59      0.59      0.59        57
weighted avg       0.64      0.63      0.63        57
```

In [57]:

```
print("KNN- Classifier")
print(classification_report(y_test, y_pred_knn_sc))
print("\n")
```

```
KNN- Classifier
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        19
         1.0       0.67      1.00      0.80        38

    accuracy                           0.67        57
   macro avg       0.33      0.50      0.40        57
weighted avg       0.44      0.67      0.53        57



C:\Users\sharm\anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [58]:

```
print("Decision Tree")
print(classification_report(y_test, y_pred_dt_sc))
print("\n")
```

```
Decision Tree
              precision    recall  f1-score   support

         0.0       0.89      0.42      0.57        19
         1.0       0.77      0.97      0.86        38

    accuracy                           0.79        57
   macro avg       0.83      0.70      0.72        57
weighted avg       0.81      0.79      0.76        57
```

In [59]:

```
print("Random Forest")
print(classification_report(y_test, y_pred_rf_sc))
print("\n")
```

```
Random Forest
              precision    recall  f1-score   support

         0.0       1.00      0.42      0.59        19
         1.0       0.78      1.00      0.87        38

    accuracy                           0.81        57
   macro avg       0.89      0.71      0.73        57
weighted avg       0.85      0.81      0.78        57
```

In [60]:

```
print("XGboost Classifier")
print(classification_report(y_test, y_pred_xgb_sc))
print("\n")
```

```
XGboost Classifier
              precision    recall  f1-score   support

         0.0       1.00      0.95      0.97        19
         1.0       0.97      1.00      0.99        38

    accuracy                           0.98        57
   macro avg       0.99      0.97      0.98        57
weighted avg       0.98      0.98      0.98        57
```

Cross validation

In [61]:

```
from sklearn.model_selection import cross_val_score
cross_validation = cross_val_score(estimator = xgb_classifier, X = X_train_sc,y = y_train,
print("Cross validation accuracy of XGBoost model = ", cross_validation)
print("\nCross validation mean accuracy of XGBoost model = ", cross_validation.mean())
```

```
C:\Users\sharm\anaconda3\lib\site-packages\xgboost\sklearn.py:888: UserWarni
ng: The use of label encoder in XGBClassifier is deprecated and will be remo
ved in a future release. To remove this warning, do the following: 1) Pass o
ption use_label_encoder=False when constructing XGBClassifier object; and 2)
Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_
class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[14:46:08] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[14:46:08] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[14:46:08] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[14:46:08] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[14:46:09] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[14:46:09] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[14:46:09] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[14:46:09] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[14:46:09] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
[14:46:09] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_
```

```
1.3.0/src/learner.cc:1061: Starting in XGBoost 1.3.0, the default evaluation
metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behav
ior.
Cross validation accuracy of XGBoost model =  [0.98076923 0.96153846 0.98039
216 0.98039216 0.94117647 0.94117647
 0.98039216 1.         1.         0.8627451 ]

Cross validation mean accuracy of XGBoost model =  0.9628582202111614
```

```
In [ ]:
```