CASE STUDY :: BREAST CANCER CLASSIFICATION

STEP #1:: PROBLEM STATEMENT

.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset

Data Set Characteristics:

```
:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class

:Attribute Information:
    - radius (mean of distances from center to points on the perimeter)
    - texture (standard deviation of gray-scale values)
    - perimeter
    - area
    - smoothness (local variation in radius lengths)
    - compactness (perimeter^2 / area - 1.0)
    - concavity (severity of concave portions of the contour)
    - concave points (number of concave portions of the contour)
    - symmetry
    - fractal dimension ("coastline approximation" - 1)

    The mean, standard error, and "worst" or largest (mean of the three
    worst/largest values) of these features were computed for each image,
    resulting in 30 features.  For instance, field 0 is Mean Radius, field
    10 is Radius SE, field 20 is Worst Radius.

    - class:
            - WDBC-Malignant
            - WDBC-Benign

:Summary Statistics:

===================================== ====== ======
                                       Min    Max
===================================== ====== ======
radius (mean):                        6.981  28.11
texture (mean):                       9.71   39.28
perimeter (mean):                     43.79  188.5
area (mean):                          143.5  2501.0
smoothness (mean):                    0.053  0.163
compactness (mean):                   0.019  0.345
concavity (mean):                     0.0    0.427
concave points (mean):                0.0    0.201
symmetry (mean):                      0.106  0.304
fractal dimension (mean):             0.05   0.097
radius (standard error):              0.112  2.873
texture (standard error):             0.36   4.885
perimeter (standard error):           0.757  21.98
area (standard error):                6.802  542.2
smoothness (standard error):          0.002  0.031
compactness (standard error):         0.002  0.135
concavity (standard error):           0.0    0.396
concave points (standard error):      0.0    0.053
symmetry (standard error):            0.008  0.079
```

```
fractal dimension (standard error):   0.001  0.03
radius (worst):                       7.93   36.04
texture (worst):                      12.02  49.54
perimeter (worst):                    50.41  251.2
area (worst):                         185.2  4254.0
smoothness (worst):                   0.071  0.223
compactness (worst):                  0.027  1.058
concavity (worst):                    0.0    1.252
concave points (worst):               0.0    0.291
symmetry (worst):                     0.156  0.664
fractal dimension (worst):            0.055  0.208
===================================== ====== ======

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995
```

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets. https://goo.gl/U2Uwz2 (https://goo.gl/U2Uwz2)

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

The actual linear program used to obtain the separating plane in the 3-dimensional space is that described in: [K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu cd math-prog/cpo-dataset/machine-learn/WDBC/

In [ ]:

```

```

STEP #2:: IMPORT LIBRARIES

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

LOAD ACTUAL CANCER DATA SETS

In [2]:
```
from sklearn.datasets import load_breast_cancer
```

CREATE INSTANCES OF LOADED DATA SETS

In [3]:
```
cancer = load_breast_cancer()
```

VIEW THE DATA

In [4]:
```
cancer
```

Out[4]:
```
{'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, 0, 1, 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 (diagnostic)
dataset\n--------------------------------------------\n\n**Data Set Character
istics:**\n\n    :Number of Instances: 569\n\n    :Number of Attributes: 30 n
umeric, 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        - c
ompactness (perimeter^2 / area - 1.0)\n        - concavity (severity of conca
ve portions of the contour)\n        - concave points (number of concave port
ions of the contour)\n        - symmetry\n        - fractal dimension ("coast
line approximation" - 1)\n\n        The mean, standard error, and "worst" or
largest (mean of the three\n        worst/largest values) of these features w
ere computed for each image,\n        resulting in 30 features.  For instanc
e, field 0 is Mean Radius, field\n        10 is Radius SE, field 20 is Worst
Radius.\n\n        - class:\n                - WDBC-Malignant\n
```

```
- WDBC-Benign\n\n    :Summary Statistics:\n\n    ============================
========= ====== ======\n                                                    Min    Ma
x\n    =================================== ====== ======\n    radius (mea
n):                        6.981  28.11\n    texture (mean):
9.71   39.28\n    perimeter (mean):                     43.79  188.5\n    are
a (mean):                        143.5  2501.0\n    smoothness (mean):
0.053  0.163\n    compactness (mean):                   0.019  0.345\n    con
cavity (mean):                   0.0    0.427\n    concave points (mean):
0.0    0.201\n    symmetry (mean):                      0.106  0.304\n    fra
ctal dimension (mean):           0.05   0.097\n    radius (standard error):
0.112  2.873\n    texture (standard error):             0.36   4.885\n    per
imeter (standard error):         0.757  21.98\n    area (standard error):
6.802  542.2\n    smoothness (standard error):          0.002  0.031\n    com
pactness (standard error):       0.002  0.135\n    concavity (standard erro
r):           0.0    0.396\n    concave points (standard error):      0.0
0.053\n    symmetry (standard error):            0.008  0.079\n    fractal di
mension (standard error):   0.001  0.03\n    radius (worst):
7.93   36.04\n    texture (worst):                      12.02  49.54\n    per
imeter (worst):                  50.41  251.2\n    area (worst):
185.2  4254.0\n    smoothness (worst):                   0.071  0.223\n    co
mpactness (worst):                0.027  1.058\n    concavity (worst):
0.0    1.252\n    concave points (worst):               0.0    0.291\n    sym
metry (worst):                   0.156  0.664\n    fractal dimension (wors
t):           0.055  0.208\n    ===================================== ======
======\n\n    :Missing Attribute Values: None\n\n    :Class Distribution: 212
- Malignant, 357 - Benign\n\n    :Creator:  Dr. William H. Wolberg, W. Nick S
treet, 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) dataset
s.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of
a fine needle\naspirate (FNA) of a breast mass.  They describe\ncharacteristi
cs of the cell nuclei present in the image.\n\nSeparating plane described abo
ve was obtained using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Deci
sion Tree\nConstruction Via Linear Programming." Proceedings of the 4th\nMidw
est Artificial Intelligence and Cognitive Science Society,\npp. 97-101, 199
2], a classification method which uses linear\nprogramming to construct a dec
ision tree.  Relevant features\nwere selected using an exhaustive search in t
he space of 1-4\nfeatures and 1-3 separating planes.\n\nThe actual linear pro
gram used to obtain the separating plane\nin the 3-dimensional space is that
described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramm
ing Discrimination of Two Linearly Inseparable Sets",\nOptimization Methods a
nd Software 1, 1992, 23-34].\n\nThis database is also available through the U
W CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-lea
rn/WDBC/\n\n.. topic:: References\n\n   - W.N. Street, W.H. Wolberg and O.L.
Mangasarian. Nuclear feature extraction \n     for breast tumor diagnosis. IS
&T/SPIE 1993 International Symposium on \n     Electronic Imaging: Science an
d Technology, volume 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 via linear programming. Operations Research, 43(4), pages 570-577,
\n     July-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mangasaria
n. Machine learning techniques\n     to diagnose breast cancer from fine-need
le aspirates. Cancer Letters 77 (1994) \n     163-171.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'me
an 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\\SUMAN DHUNGANA\\anaconda3\\lib\\site-packages\\sklea
rn\\datasets\\data\\breast_cancer.csv'}
```

In [5]:
```
cancer.keys()
```

Out[5]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_name
s', 'filename'])
```

In [6]:
```
print(cancer['DESCR'])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, fiel
d
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
    symmetry (standard error):            0.008  0.079
```

```
    fractal dimension (standard error):   0.001  0.03
    radius (worst):                       7.93   36.04
    texture (worst):                      12.02  49.54
    perimeter (worst):                    50.41  251.2
    area (worst):                         185.2  4254.0
    smoothness (worst):                   0.071  0.223
    compactness (worst):                  0.027  1.058
    concavity (worst):                    0.0    1.252
    concave points (worst):               0.0    0.291
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extracti
on
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
```

```
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis 
and
     prognosis via linear programming. Operations Research, 43(4), pages 570-
577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techni
ques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 
(1994)
     163-171.
```

In [7]:
```python
print(cancer['target_names'])
```

```
['malignant' 'benign']
```

In [8]:
```python
print(cancer['target'])
```

```
[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]
```

In [9]:
```python
print(cancer['feature_names'])
```

```
['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']
```

In [10]:
```python
cancer['data'].shape
```

Out[10]:
```
(569, 30)
```

CREATE DATAFRAMES

In [11]:
```
df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']], columns = np
.append(cancer['feature_names'], ['target']))
```

In [12]:
```
df_cancer.head()
```

Out[12]:

	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 rows × 31 columns

In [13]:
```
df_cancer.tail(2)
```

Out[13]:

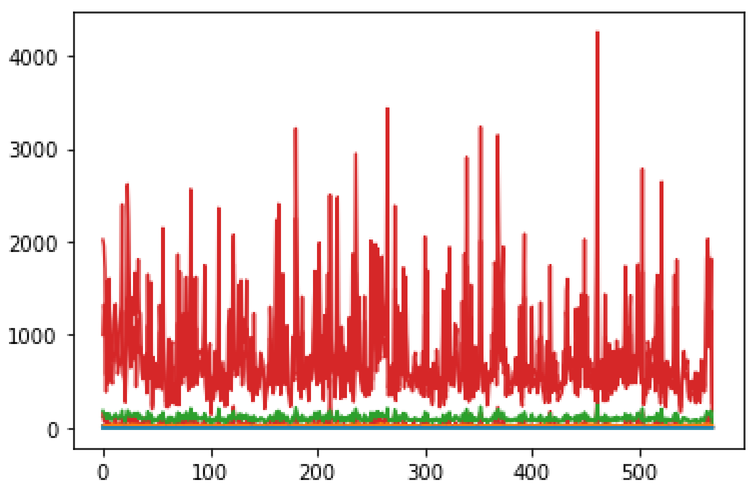
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmet
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.3514	0.152	0.23
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.0000	0.000	0.15

2 rows × 31 columns

STEP #3 :: VISUALIZING THE DATA

In [14]:
```
plt.plot(df_cancer)
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x24c3e887208>,
 <matplotlib.lines.Line2D at 0x24c3ef3ef48>,
 <matplotlib.lines.Line2D at 0x24c3ef4d148>,
 <matplotlib.lines.Line2D at 0x24c3ef4d308>,
 <matplotlib.lines.Line2D at 0x24c3ef4d5c8>,
 <matplotlib.lines.Line2D at 0x24c3ef4d888>,
 <matplotlib.lines.Line2D at 0x24c3ef4da88>,
 <matplotlib.lines.Line2D at 0x24c3ef4dd08>,
 <matplotlib.lines.Line2D at 0x24c3ef4d508>,
 <matplotlib.lines.Line2D at 0x24c3ef4d7c8>,
 <matplotlib.lines.Line2D at 0x24c3e87b808>,
 <matplotlib.lines.Line2D at 0x24c3ef50688>,
 <matplotlib.lines.Line2D at 0x24c3ef50908>,
 <matplotlib.lines.Line2D at 0x24c3ef50b88>,
 <matplotlib.lines.Line2D at 0x24c3ef50e08>,
 <matplotlib.lines.Line2D at 0x24c3ef520c8>,
 <matplotlib.lines.Line2D at 0x24c3ef52348>,
 <matplotlib.lines.Line2D at 0x24c3ef525c8>,
 <matplotlib.lines.Line2D at 0x24c3ef52848>,
 <matplotlib.lines.Line2D at 0x24c3ef52ac8>,
 <matplotlib.lines.Line2D at 0x24c3ef52d48>,
 <matplotlib.lines.Line2D at 0x24c3ef52fc8>,
 <matplotlib.lines.Line2D at 0x24c3ef55288>,
 <matplotlib.lines.Line2D at 0x24c3ef55508>,
 <matplotlib.lines.Line2D at 0x24c3ef55788>,
 <matplotlib.lines.Line2D at 0x24c3ef55a08>,
 <matplotlib.lines.Line2D at 0x24c3ef55c88>,
 <matplotlib.lines.Line2D at 0x24c3ef55f08>,
 <matplotlib.lines.Line2D at 0x24c3ef571c8>,
 <matplotlib.lines.Line2D at 0x24c3ef57448>,
 <matplotlib.lines.Line2D at 0x24c3ef576c8>]
```

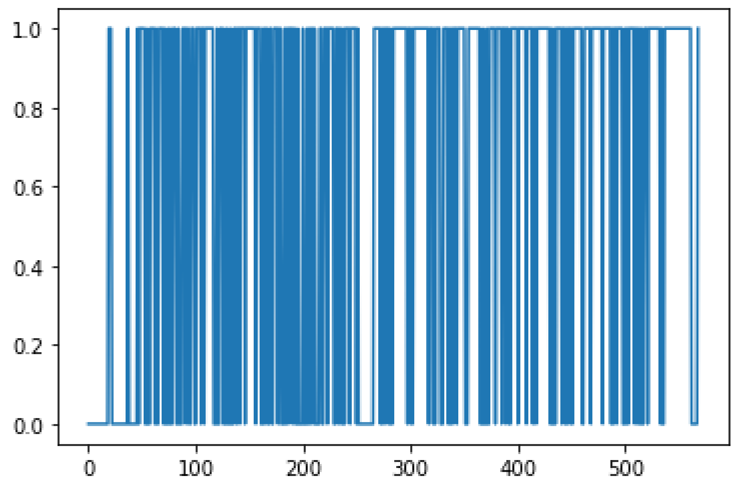


In [15]:
```
df_cancer.columns
```

Out[15]:
```
Index(['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',
       'target'],
      dtype='object')
```

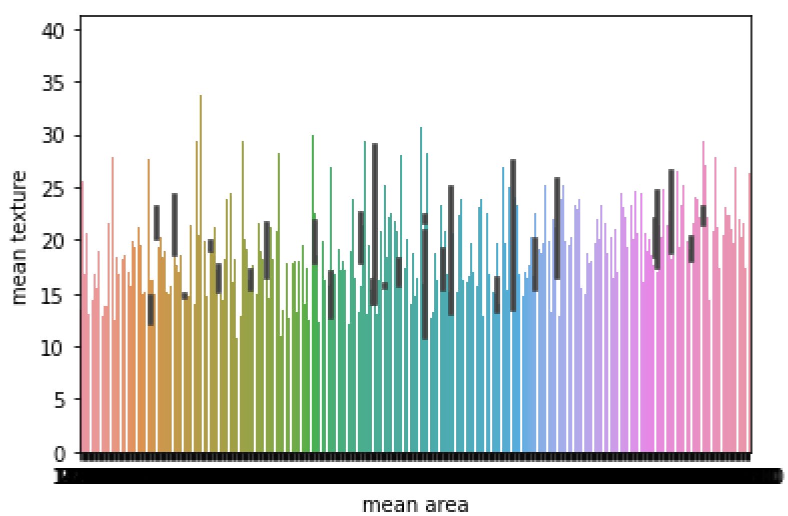
In [16]:
```
plt.plot(df_cancer['target'])
```

Out[16]:
```
[<matplotlib.lines.Line2D at 0x24c3f051d88>]
```



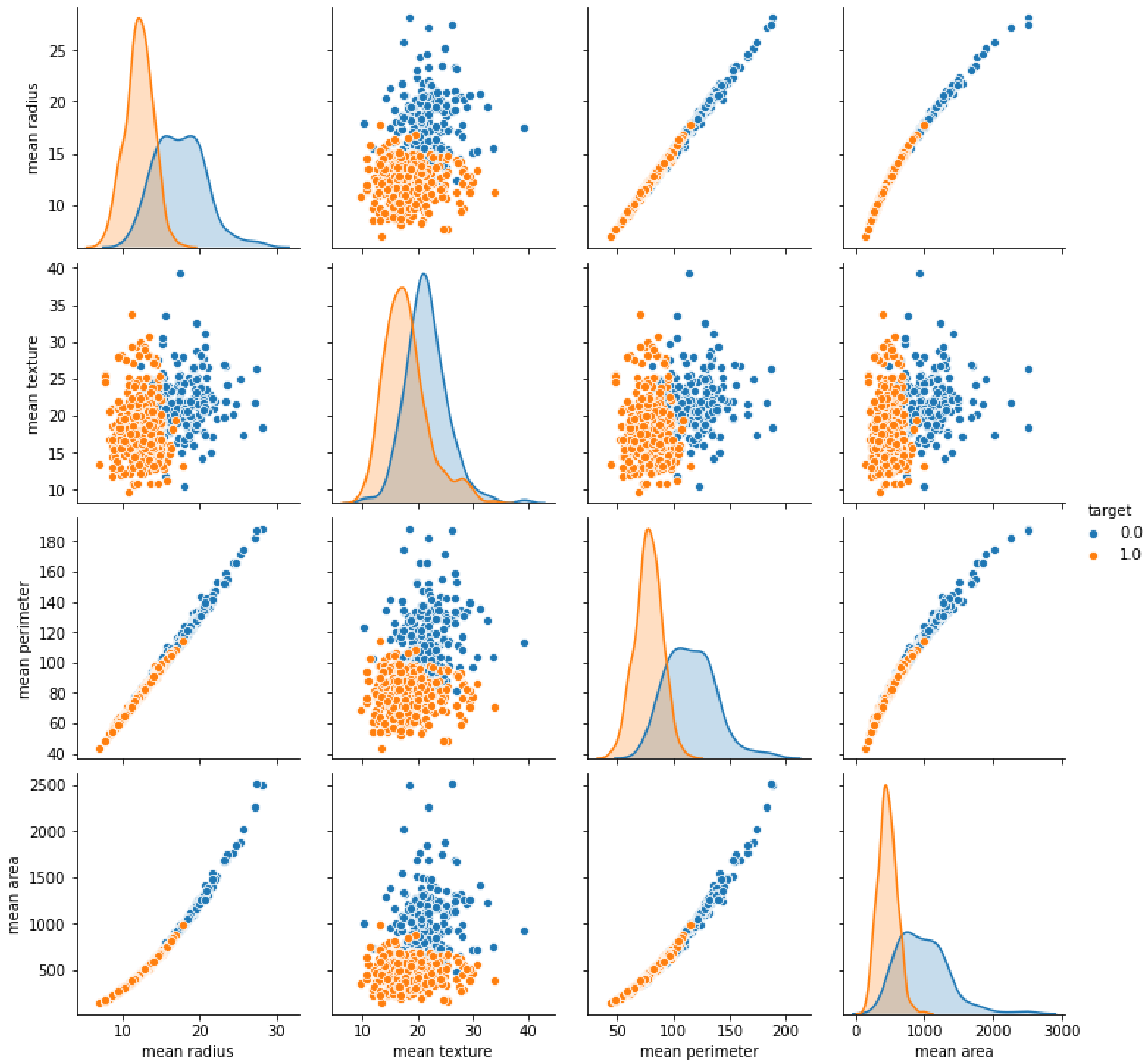
In [17]:
```
sns.barplot(x='mean area', y='mean texture', data = df_cancer)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x24c3f0bd688>
```



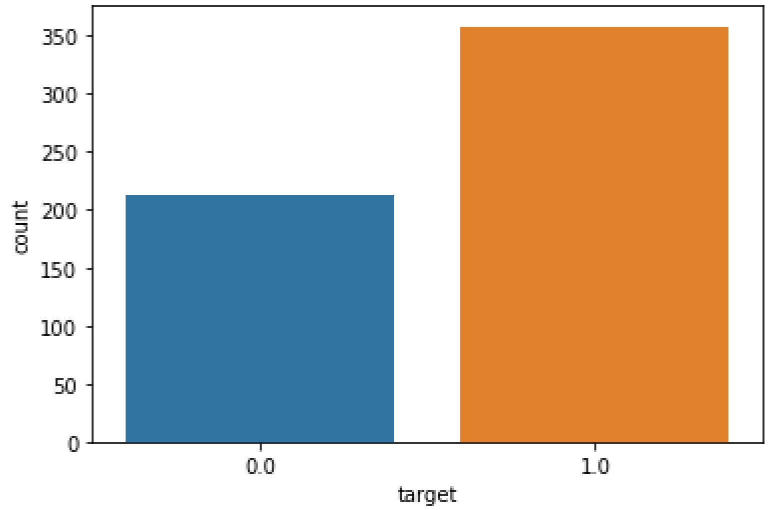
In [18]:
```
sns.pairplot(df_cancer, hue = 'target',vars = ['mean radius', 'mean texture',
'mean perimeter', 'mean area'])
```

Out[18]: <seaborn.axisgrid.PairGrid at 0x24c41352cc8>



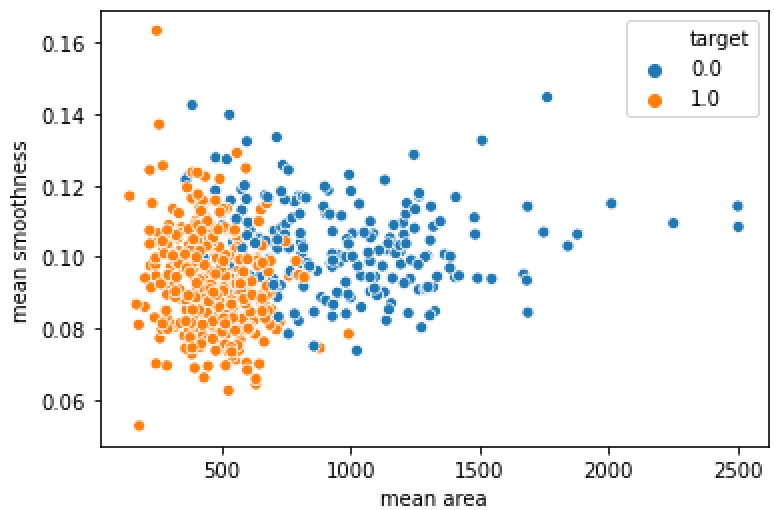
In [19]:
```
sns.countplot(df_cancer['target'])
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x24c41f1c648>



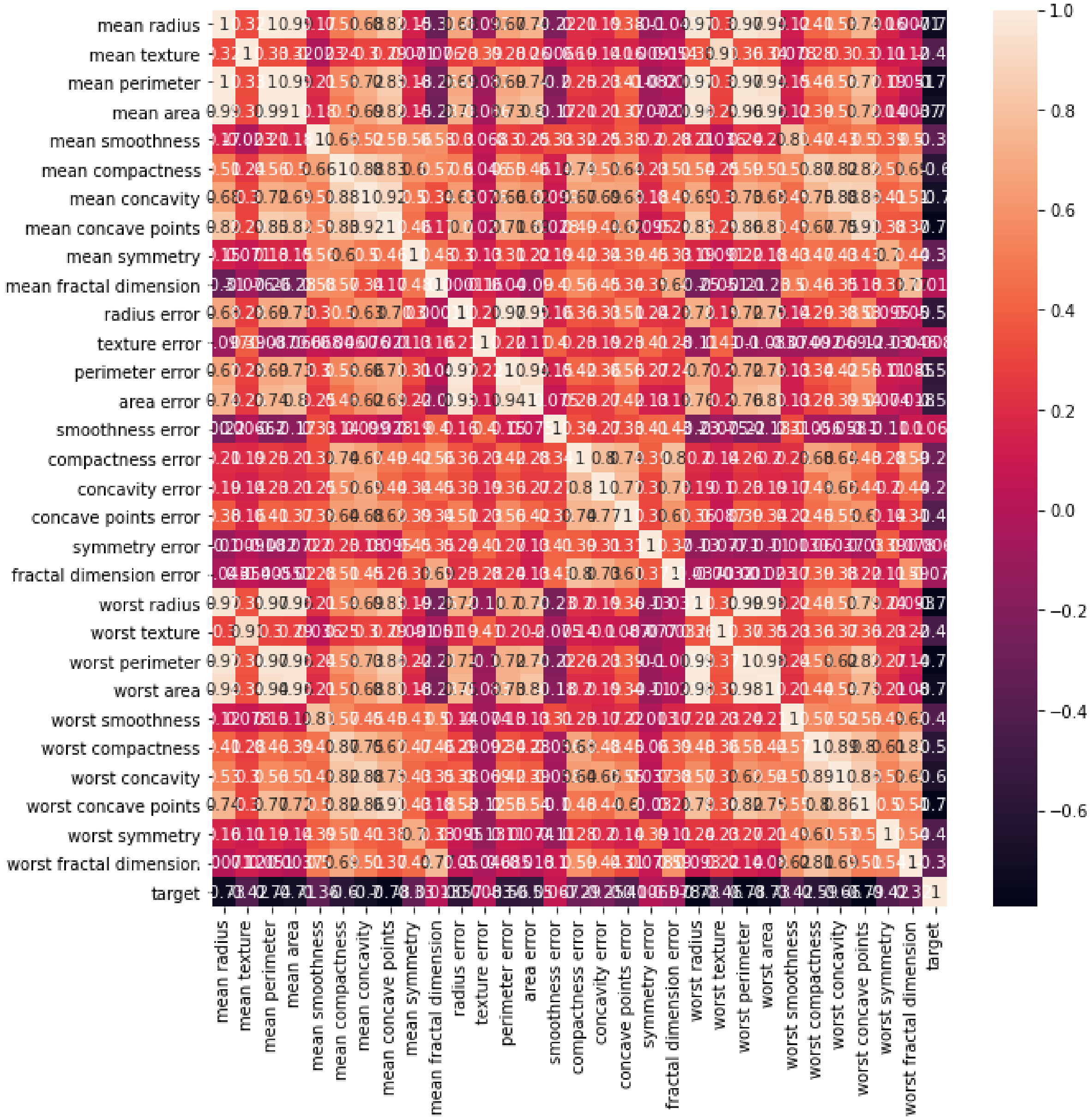
In [20]:
```
sns.scatterplot(x='mean area', y='mean smoothness', hue = 'target', data=df_ca
ncer)
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x24c4218d148>



In [21]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df_cancer.corr(), annot=True)
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x24c421e3808>



In [ ]:

STEP #4: Model Training (Finding a problem solution)

In [22]:
```
X = df_cancer.drop(['target'], axis=1)
y = df_cancer['target']
```

In [23]:
```
X.head()
```

Out[23]:

	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 rows × 30 columns

In [24]:
```
y.head()
```

Out[24]:
```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: target, dtype: float64
```

In [25]:
```
from sklearn.model_selection import train_test_split
```

In [26]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=5)
```

In [27]:
```
X_train
```

Out[27]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmet
306	13.200	15.82	84.07	537.3	0.08511	0.05251	0.001461	0.003261	0.16
410	11.360	17.57	72.49	399.8	0.08858	0.05313	0.027830	0.021000	0.16
197	18.080	21.84	117.40	1024.0	0.07371	0.08642	0.110300	0.057780	0.17
376	10.570	20.22	70.15	338.3	0.09073	0.16600	0.228000	0.059410	0.21
244	19.400	23.50	129.10	1155.0	0.10270	0.15580	0.204900	0.088860	0.19
...	...	...	...	...	...	...	...	...	
8	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185900	0.093530	0.23
73	13.800	15.79	90.43	584.1	0.10070	0.12800	0.077890	0.050690	0.16
400	17.910	21.02	124.40	994.0	0.12300	0.25760	0.318900	0.119800	0.21
118	15.780	22.91	105.70	782.6	0.11550	0.17520	0.213300	0.094790	0.20
206	9.876	17.27	62.92	295.4	0.10890	0.07232	0.017560	0.019520	0.19

455 rows × 30 columns

In [28]:
```
X_test
```

Out[28]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmet
28	15.30	25.27	102.40	732.4	0.10820	0.16970	0.16830	0.08751	0.19
163	12.34	22.22	79.85	464.5	0.10120	0.10150	0.05370	0.02822	0.15
123	14.50	10.89	94.28	640.7	0.11010	0.10990	0.08842	0.05778	0.18
361	13.30	21.57	85.24	546.1	0.08582	0.06373	0.03344	0.02424	0.18
549	10.82	24.21	68.89	361.6	0.08192	0.06602	0.01548	0.00816	0.19
...	...	...	...	...	...	...	...	...	
414	15.13	29.81	96.71	719.5	0.08320	0.04605	0.04686	0.02739	0.18
515	11.34	18.61	72.76	391.2	0.10490	0.08499	0.04302	0.02594	0.19
186	18.31	18.58	118.60	1041.0	0.08588	0.08468	0.08169	0.05814	0.16
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.25
261	17.35	23.06	111.00	933.1	0.08662	0.06290	0.02891	0.02837	0.15

114 rows × 30 columns

In [29]:
```
y_train.head()
```

Out[29]:
```
306    1.0
410    1.0
197    0.0
376    1.0
244    0.0
Name: target, dtype: float64
```

In [30]:
```
y_test.head()
```

Out[30]:
```
28     0.0
163    1.0
123    1.0
361    1.0
549    1.0
Name: target, dtype: float64
```

In [31]:
```
from sklearn.svm import SVC

# svm :: Support vector machine
# SVC :: Support Vector Classification
```

In [32]:
```
svc_model = SVC()
```

In [33]:
```
svc_model.fit(X_train, y_train)
```

Out[33]:
```
SVC()
```

In [ ]:
```

```

STEP #5: Evaluating the model

In [34]:
```
y_predict = svc_model.predict(X_test)
```

In [ ]:
```

```

In [35]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [36]:
```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.85      0.92        48
         1.0       0.90      1.00      0.95        66

    accuracy                           0.94       114
   macro avg       0.95      0.93      0.94       114
weighted avg       0.94      0.94      0.94       114
```

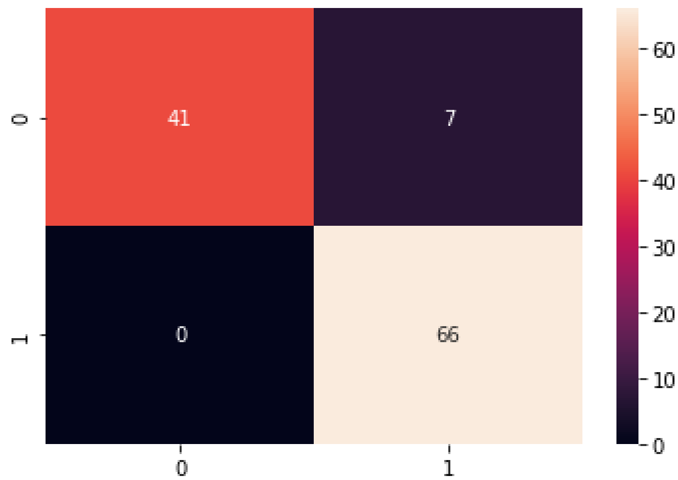
In [37]:
```
print(confusion_matrix(y_test,y_predict))
```

```
[[41  7]
 [ 0 66]]
```

In [38]:
```
cm = confusion_matrix(y_test,y_predict)
```

In [39]:
```
sns.heatmap(cm, annot=True)
```

Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x24c42ede7c8>



In [ ]:
```

```

STEP #6: Improving a model - PART 1

Improving before normalization

Need all the minimum value here

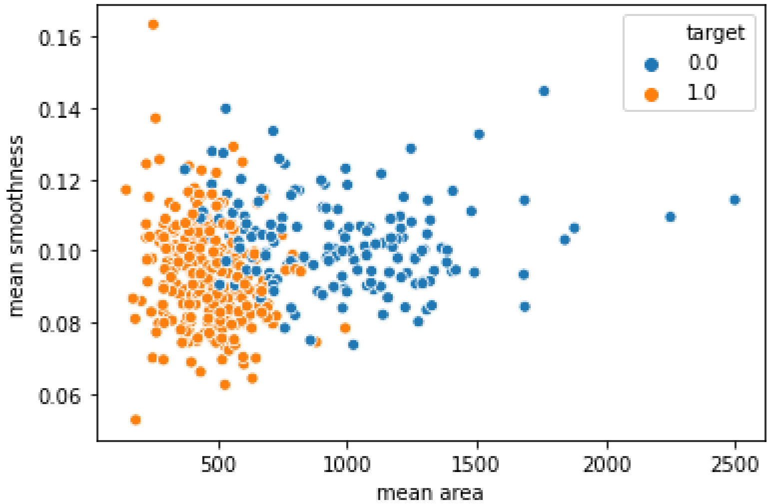
In [40]:
```
min_train = X_train.min()
```

In [41]:
```
range_train = (X_train-min_train).max()
```

In [42]:
```
X_train_scale = (X_train-min_train)/range_train
```

In [43]:
```
sns.scatterplot(x=X_train['mean area'], y = X_train['mean smoothness'], hue=y_
train)
```

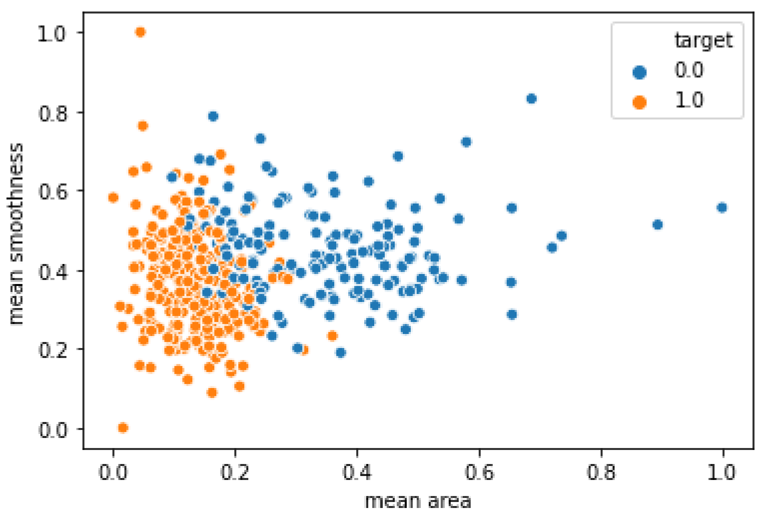
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x24c42f8ffc8>



In [44]:
```
sns.scatterplot(x=X_train_scale['mean area'], y = X_train_scale['mean smoothne
ss'], hue=y_train)

# all same but range is different here
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x24c42fedf88>



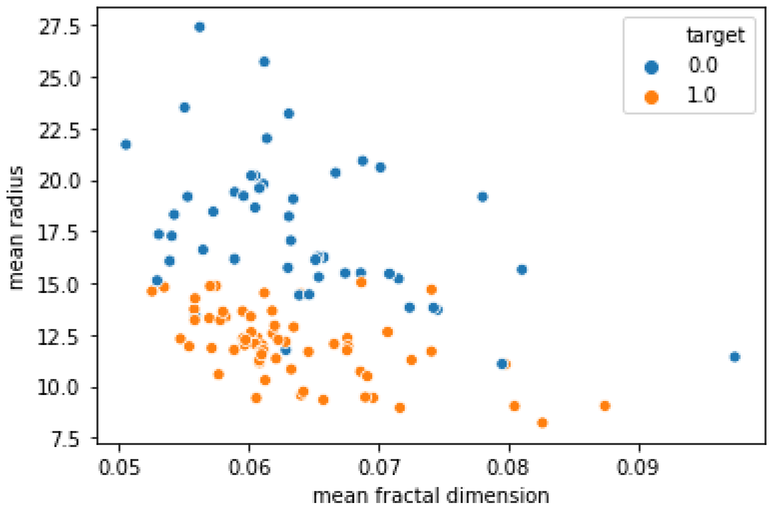
In [45]:
```
min_test = X_test.min()
```

In [46]:
```
range_test = (X_test-min_test).max()
```

In [47]:
```
X_test_scaled = (X_test-min_test)/range_test
```

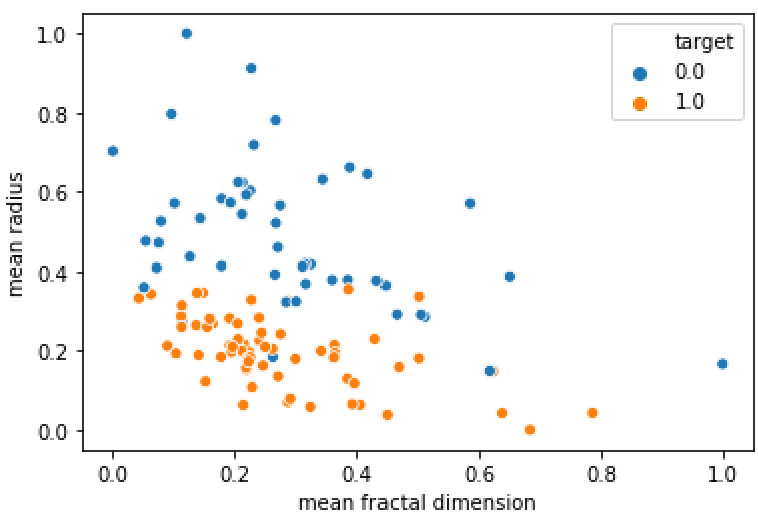
In [48]:
```
sns.scatterplot(x=X_test['mean fractal dimension'], y = X_test['mean radius'],
hue = y_test)
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x24c44056848>



In [49]:
```
sns.scatterplot(x=X_test_scaled['mean fractal dimension'], y = X_test_scaled[
'mean radius'], hue = y_test)
```

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x24c440c6e08>



In [50]:
```
svc_model.fit(X_train_scale, y_train)
```

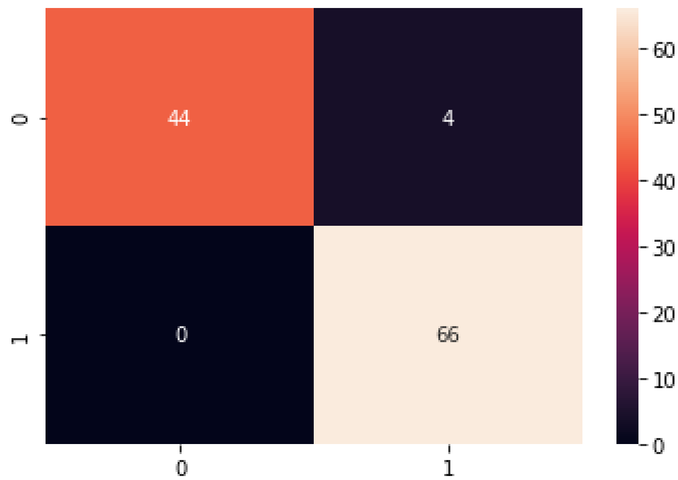
Out[50]: SVC()

In [51]:
```
y_scaled_predict = svc_model.predict(X_test_scaled)
```

In [52]:
```
cm = confusion_matrix(y_test, y_scaled_predict)
```

In [53]:
```
sns.heatmap(cm, annot=True)
```

Out[53]: <matplotlib.axes._subplots.AxesSubplot at 0x24c44137b48>



Classification report

Gives summary

In [54]:
```
print(classification_report(y_test, y_scaled_predict))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.92      0.96        48
         1.0       0.94      1.00      0.97        66

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

In [ ]:

Improving the model - PART II

After normalization::

- C Parameter
- Gamma Parameter

In [55]:
```
param_grid = {'C':[0.1,1,10,100], 'gamma':[1,0.1,0.01,0.001],'kernel':['rbf']}

# rbf:: Radial Basis Fuction
# We can only define range using dictionaries in C and gamma parameter
```

In [56]:
```
from sklearn.model_selection import GridSearchCV
```

In [57]:
```
grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=4)
```

In [58]:
```
grid
```

Out[58]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.00
1],
                         'kernel': ['rbf']},
             verbose=4)
```

In [59]:
```
grid.fit(X_train_scale, y_train)

# search the best value for gamma and c
# sklearn automatically search either gamma better or c
# We need not to worry about it
```

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] C=0.1, gamma=1, kernel=rbf ......................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=0.1, gamma=1, kernel=rbf ......................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=0.1, gamma=1, kernel=rbf ......................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.912, total=   0.0s
[CV] C=0.1, gamma=1, kernel=rbf ......................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=0.1, gamma=1, kernel=rbf ......................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.934, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ....................................
[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ....................................
[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.901, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ....................................
[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.890, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ....................................
[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.923, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ....................................
[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.868, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ...................................

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.
0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.
0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.
0s
```

```
[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.648, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ...................................
[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ...................................
[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ...................................
[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ...................................
[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.648, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf ........................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf ........................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf ........................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf ........................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf ........................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf ......................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.989, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf ......................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf ......................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.923, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf ......................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf ......................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.934, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf .....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.945, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf .....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.901, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf .....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.879, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf .....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.923, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf .....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.868, total=   0.0s
[CV] C=1, gamma=0.001, kernel=rbf ....................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.648, total=   0.0s
[CV] C=1, gamma=0.001, kernel=rbf ....................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=1, gamma=0.001, kernel=rbf ....................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=1, gamma=0.001, kernel=rbf ....................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
```

```
[CV] C=1, gamma=0.001, kernel=rbf ....................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf .......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf .......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf .......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf .......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf .......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf .....................................
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf .....................................
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf .....................................
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf .....................................
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.989, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf .....................................
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ........ C=10, gamma=0.01, kernel=rbf, score=0.989, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ........ C=10, gamma=0.01, kernel=rbf, score=0.945, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ........ C=10, gamma=0.01, kernel=rbf, score=0.923, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ........ C=10, gamma=0.01, kernel=rbf, score=0.967, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ........ C=10, gamma=0.01, kernel=rbf, score=0.934, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ....... C=10, gamma=0.001, kernel=rbf, score=0.945, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ....... C=10, gamma=0.001, kernel=rbf, score=0.901, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ....... C=10, gamma=0.001, kernel=rbf, score=0.879, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ....... C=10, gamma=0.001, kernel=rbf, score=0.923, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ....... C=10, gamma=0.001, kernel=rbf, score=0.879, total=   0.0s
[CV] C=100, gamma=1, kernel=rbf ......................................
[CV] .......... C=100, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=100, gamma=1, kernel=rbf ......................................
[CV] .......... C=100, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=100, gamma=1, kernel=rbf ......................................
[CV] .......... C=100, gamma=1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=100, gamma=1, kernel=rbf ......................................
[CV] .......... C=100, gamma=1, kernel=rbf, score=0.989, total=   0.0s
[CV] C=100, gamma=1, kernel=rbf ......................................
[CV] .......... C=100, gamma=1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=100, gamma=0.1, kernel=rbf ....................................
[CV] ........ C=100, gamma=0.1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=100, gamma=0.1, kernel=rbf ....................................
[CV] ........ C=100, gamma=0.1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=100, gamma=0.1, kernel=rbf ....................................
```

```
[CV] ........ C=100, gamma=0.1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=100, gamma=0.1, kernel=rbf ....................................
[CV] ........ C=100, gamma=0.1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=100, gamma=0.1, kernel=rbf ....................................
[CV] ........ C=100, gamma=0.1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=100, gamma=0.01, kernel=rbf ...................................
[CV] ....... C=100, gamma=0.01, kernel=rbf, score=1.000, total=   0.0s
[CV] C=100, gamma=0.01, kernel=rbf ...................................
[CV] ....... C=100, gamma=0.01, kernel=rbf, score=0.967, total=   0.0s
[CV] C=100, gamma=0.01, kernel=rbf ...................................
[CV] ....... C=100, gamma=0.01, kernel=rbf, score=0.967, total=   0.0s
[CV] C=100, gamma=0.01, kernel=rbf ...................................
[CV] ....... C=100, gamma=0.01, kernel=rbf, score=0.989, total=   0.0s
[CV] C=100, gamma=0.01, kernel=rbf ...................................
[CV] ....... C=100, gamma=0.01, kernel=rbf, score=0.945, total=   0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=100, gamma=0.001, kernel=rbf, score=0.989, total=   0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=100, gamma=0.001, kernel=rbf, score=0.945, total=   0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=100, gamma=0.001, kernel=rbf, score=0.923, total=   0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=100, gamma=0.001, kernel=rbf, score=0.967, total=   0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV] ...... C=100, gamma=0.001, kernel=rbf, score=0.934, total=   0.0s

[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:    1.2s finished
```

Out[59]:

```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.00
1],
                         'kernel': ['rbf']},
             verbose=4)
```

In [60]:

```
grid.best_params_

# it displays the best value here
```

Out[60]:

```
{'C': 1, 'gamma': 1, 'kernel': 'rbf'}
```

In [61]:

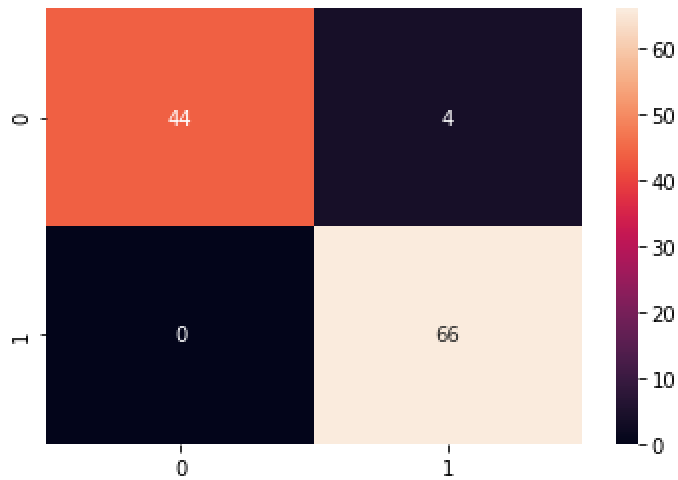
```
grid_predictions = grid.predict(X_test_scaled)
```

In [62]:

```
cm = confusion_matrix(y_test, grid_predictions)
```

In [63]:
```
sns.heatmap(cm, annot=True)
```

Out[63]: <matplotlib.axes._subplots.AxesSubplot at 0x24c441f2bc8>



In [64]:
```
print(classification_report(y_test, grid_predictions))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.92      0.96        48
         1.0       0.94      1.00      0.97        66

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