CASE STUDY: BREAST CANCER CLASSIFICATION

STEP #1: PROBLEM STATEMENT ¶

- SVM (Support vector model) - Great for separating classes of data via Max Margin Hyper Plane
- Predicting if the cancer diagnosis is benign or malignant based on several observations/features
- 30 features are used:

```
- 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)
```

- Datasets are linearly separable using all 30 input features
- Number of Instances: 569
- Class Distribution: 212 Malignant, 357 Benign
- Target class:

```
- Malignant
- Benign
```

STEP #2: IMPORTING DATA

In [3]:

```
# import libraries
import pandas as pd # Import Pandas for data manipulation using datafram
es
import numpy as np # Import Numpy for data statistical analysis
import matplotlib.pyplot as plt # Import matplotlib for data visualisati
on
import seaborn as sns # Statistical data visualization
# %matplotlib inline
```

In [4]:

```
# Import Cancer data drom the Sklearn library
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer() # Taking an instance of it
```

In [3]:

```
cancer
```

```
Out[3]: {'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601
e-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]),
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': 'Breast Cancer Wisconsin (Diagnostic) Database\n=============
=============================\n\nNotes\n-----\nData Set Characterist
ics:\n    :Number of Instances: 569\n\n    :Number of Attributes: 30 nu
meric, predictive attributes and the class\n\n    :Attribute Informatio
n:\n        - radius (mean of distances from center to points on the pe
rimeter)\n        - texture (standard deviation of gray-scale values)\n
        - perimeter\n        - area\n        - smoothness (local variat
ion in radius lengths)\n        - compactness (perimeter^2 / area - 1.
0)\n        - concavity (severity of concave portions of the contour)\n
        - concave points (number of concave portions of the contour)\n
       - symmetry \n        - fractal dimension ("coastline approximati
on" - 1)\n\n        The mean, standard error, and "worst" or largest (m
ean of the three\n        largest values) of these features were comput
ed for each image,\n        resulting in 30 features.  For instance, fi
eld 3 is Mean Radius, field\n        13 is Radius SE, field 23 is Worst
Radius.\n\n        - class:\n                - WDBC-Malignant\n
        - WDBC-Benign\n\n    :Summary Statistics:\n\n    ==============
====================== ====== ======\n
           Min    Max\n    =================================== ======
======\n    radius (mean):                        6.981  28.11\n    tex
ture (mean):                       9.71   39.28\n    perimeter (mean):
                    43.79  188.5\n    area (mean):
     143.5  2501.0\n    smoothness (mean):                    0.053  0.
163\n    compactness (mean):                    0.019  0.345\n    concav
ity (mean):                      0.0    0.427\n    concave points (mea
n):                  0.0    0.201\n    symmetry (mean):
    0.106  0.304\n    fractal dimension (mean):             0.05   0.09
7\n    radius (standard error):              0.112  2.873\n    texture
(standard error):              0.36   4.885\n    perimeter (standard err
or):           0.757  21.98\n    area (standard error):
6.802  542.2\n    smoothness (standard error):          0.002  0.031\n
   compactness (standard error):         0.002  0.135\n    concavity (s
tandard error):           0.0    0.396\n    concave points (standard er
ror):      0.0    0.053\n    symmetry (standard error):            0.00
8  0.079\n    fractal dimension (standard error):   0.001  0.03\n    ra
dius (worst):                        7.93   36.04\n    texture (worst):
                     12.02  49.54\n    perimeter (worst):
      50.41  251.2\n    area (worst):                         185.2  42
54.0\n    smoothness (worst):                   0.071  0.223\n    compa
ctness (worst):                  0.027  1.058\n    concavity (worst):
                  0.0    1.252\n    concave points (worst):
   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. Mangas
arian\n\n    :Donor: Nick Street\n\n    :Date: November, 1995\n\nThis i
s a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.\nhttp
```

```
s://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a
fine needle\naspirate (FNA) of a breast mass.  They describe\ncharacter
istics of the cell nuclei present in the image.\n\nSeparating plane des
cribed above was obtained using\nMultisurface Method-Tree (MSM-T) [K.
P. Bennett, "Decision Tree\nConstruction Via Linear Programming." Proce
edings of the 4th\nMidwest Artificial Intelligence and Cognitive Scienc
e Society,\npp. 97-101, 1992], a classification method which uses linea
r\nprogramming to construct a decision tree.  Relevant features\nwere s
elected using an exhaustive search in the space of 1-4\nfeatures and 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. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimin
ation of Two Linearly Inseparable Sets",\nOptimization Methods and Soft
ware 1, 1992, 23-34].\n\nThis database is also available through the UW
CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine
-learn/WDBC/\n\nReferences\n----------\n   - W.N. Street, W.H. Wolberg
and O.L. Mangasarian. Nuclear feature extraction \n     for breast tumo
r diagnosis. IS&T/SPIE 1993 International Symposium on \n     Electroni
c Imaging: Science and Technology, volume 1905, pages 861-870,\n     Sa
n Jose, CA, 1993.\n   - O.L. Mangasarian, W.N. Street and W.H. Wolberg.
Breast cancer diagnosis and \n     prognosis via linear programming. Op
erations Research, 43(4), pages 570-577, \n     July-August 1995.\n   -
W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techn
iques\n     to diagnose breast cancer from fine-needle aspirates. Cance
r Letters 77 (1994) \n     163-171.\n',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimete
r', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimensio
n',
        'radius error', 'texture error', 'perimeter error', 'area erro
r',
        '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')}
```

In [4]:
```
cancer.keys() # Titles of the data
```

Out[4]:
```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names'])
```

In [5]:
```
print(cancer['DESCR'])
```

```
Breast Cancer Wisconsin (Diagnostic) Database
=============================================

Notes
-----
Data Set Characteristics:
    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the cl
ass

    :Attribute Information:
        - radius (mean of distances from center to points on the perime
ter)
        - 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 t
hree
        largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 3 is Mean Radiu
s, field
        13 is Radius SE, field 23 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. Mangasar
ian

    :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/

References
----------
   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature ex
traction
```

```
     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 diag
nosis and
     prognosis via linear programming. Operations Research, 43(4), page
s 570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Lette
rs 77 (1994)
     163-171.
```

In [6]:
```python
print(cancer['target_names'])
```

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

In [7]:
```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 [8]:
```
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 [9]:
```
print(cancer['data'])
```

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

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

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

Out[13]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.172
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.175
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.159
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.239
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.158

5 rows × 31 columns

In [14]:
```
x = np.array([1,2,3])
x.shape
```

Out[14]: (3,)

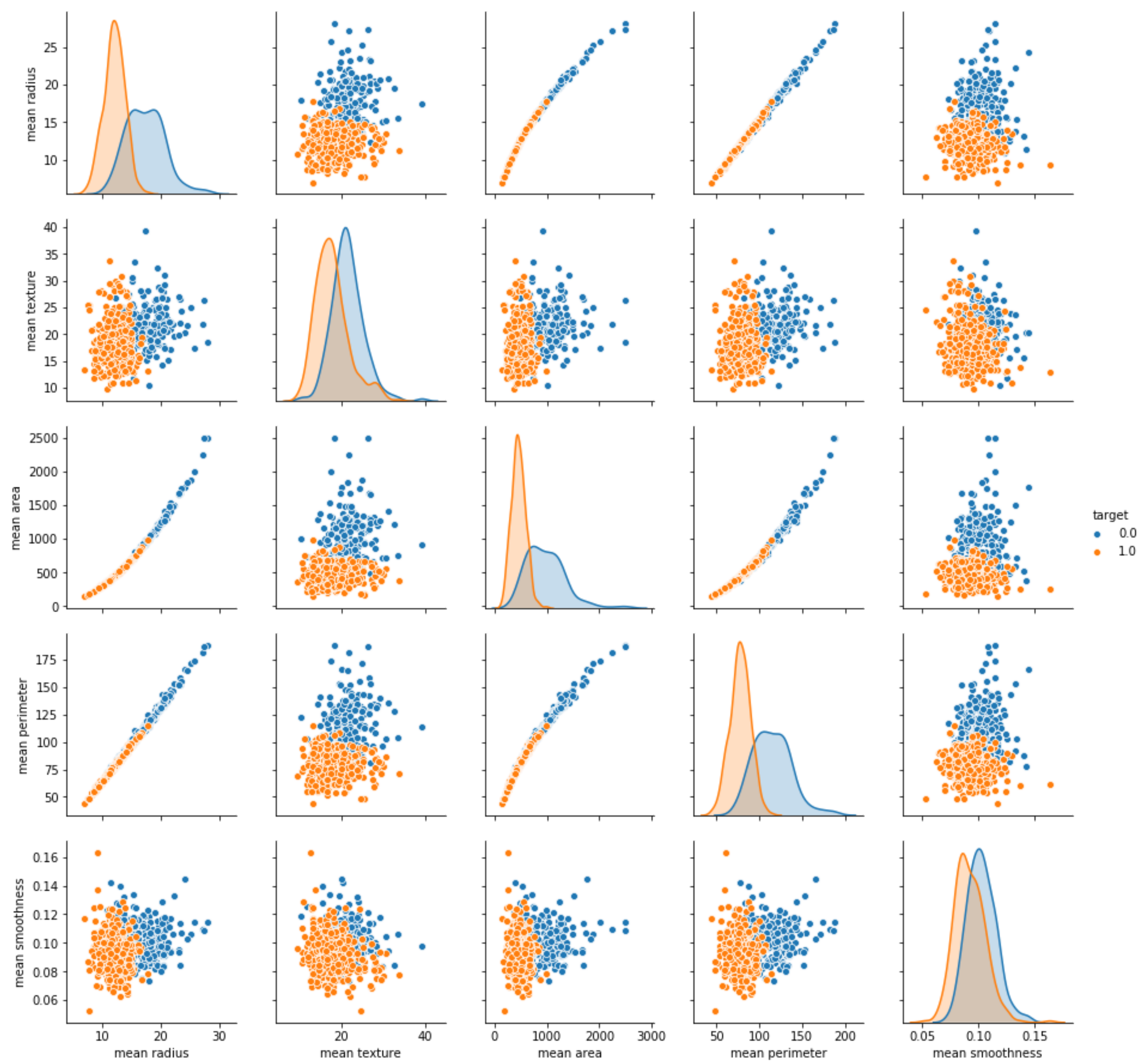
In [15]:
```
Example = np.c_[np.array([1,2,3]), np.array([4,5,6])]
Example.shape
```

Out[15]: (3, 2)

STEP #3: VISUALIZING THE DATA

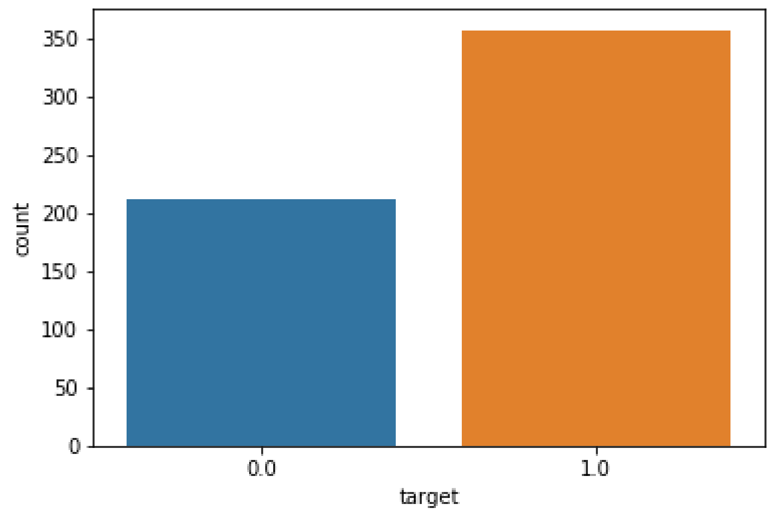
In [16]:
```
# Vars = variables to include
# Hue (specify the target class) = Dependent variable Y = Target (0,1)
sns.pairplot(df_cancer, hue = 'target', vars = ['mean radius', 'mean tex
ture', 'mean area', 'mean perimeter', 'mean smoothness'] )
```

Out[16]: <seaborn.axisgrid.PairGrid at 0x1c93d061b00>



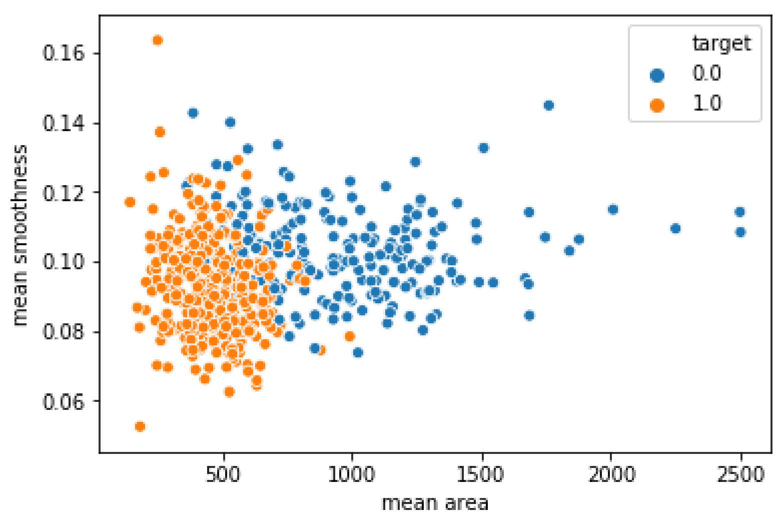
In [17]:
```
sns.countplot(df_cancer['target'], label = "Count")
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x1c93eebdeb8>



In [18]:
```
sns.scatterplot(x = 'mean area', y = 'mean smoothness', hue = 'target',
data = df_cancer)
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1c93ef31ac8>



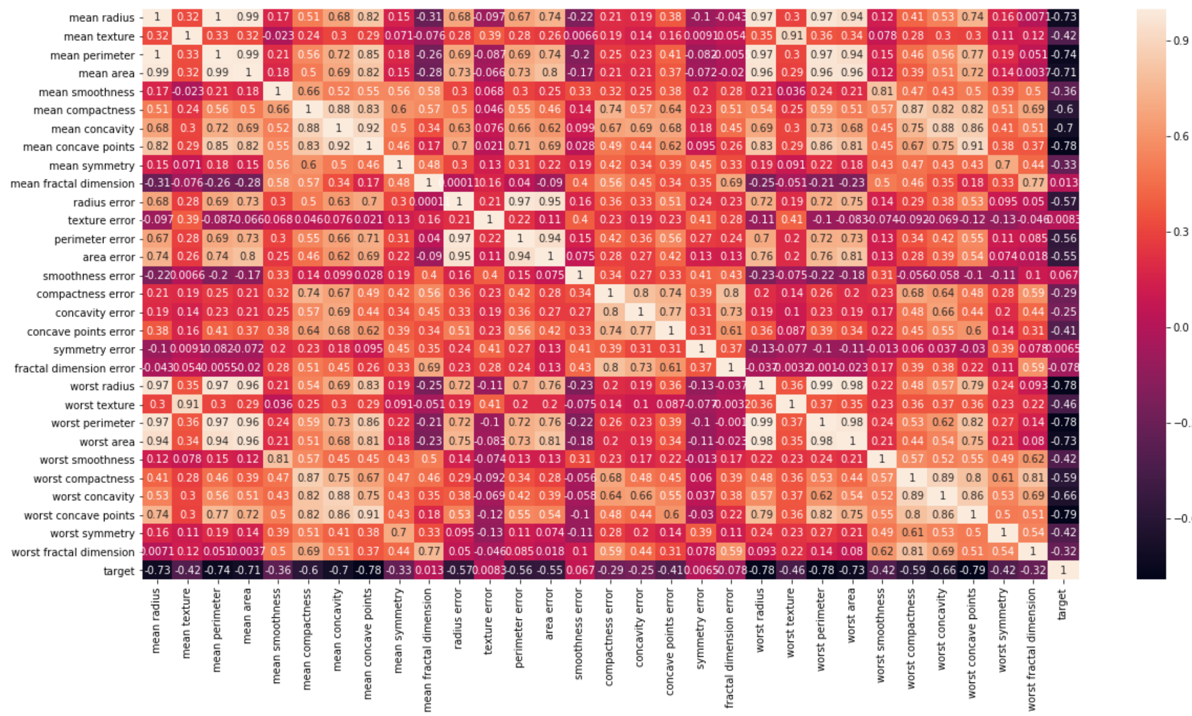
In [7]:
```
#sns.lmplot('mean area', 'mean smoothness', hue ='target', data = df_can
cer_all, fit_reg=False)
```

In [20]:

```
# Let's check the correlation between the variables
# Strong correlation between the mean radius and mean perimeter, mean ar
ea and mean primeter
plt.figure(figsize=(20,10))
sns.heatmap(df_cancer.corr(), annot=True)
# .corr method
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c93f754940>
```



STEP #4: MODEL TRAINING (FINDING A PROBLEM SOLUTION)

In [21]:

```
# Let's drop the target label coloumns
X = df_cancer.drop(['target'],axis=1)
# Axis = 1 which is the entire column
# Method drop to drop the column (target is the column name)
# Describe all inputs
# all columns are independent variables X with Target being the dependen
t variable Y
```

In [22]:

```
X
```

Out[22]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmet
0	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300100	0.147100	0.24
1	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900	0.070170	0.18
2	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197400	0.127900	0.20
3	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241400	0.105200	0.25
4	20.290	14.34	135.10	1297.0	0.10030	0.13280	0.198000	0.104300	0.18
5	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157800	0.080890	0.20
6	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112700	0.074000	0.17
7	13.710	20.83	90.20	577.9	0.11890	0.16450	0.093660	0.059850	0.21
8	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185900	0.093530	0.23
9	12.460	24.04	83.97	475.9	0.11860	0.23960	0.227300	0.085430	0.20
10	16.020	23.24	102.70	797.8	0.08206	0.06669	0.032990	0.033230	0.15
11	15.780	17.89	103.60	781.0	0.09710	0.12920	0.099540	0.066060	0.18
12	19.170	24.80	132.40	1123.0	0.09740	0.24580	0.206500	0.111800	0.23
13	15.850	23.95	103.70	782.7	0.08401	0.10020	0.099380	0.053640	0.18
14	13.730	22.61	93.60	578.3	0.11310	0.22930	0.212800	0.080250	0.20
15	14.540	27.54	96.73	658.8	0.11390	0.15950	0.163900	0.073640	0.23
16	14.680	20.13	94.74	684.5	0.09867	0.07200	0.073950	0.052590	0.15
17	16.130	20.68	108.10	798.8	0.11700	0.20220	0.172200	0.102800	0.21
18	19.810	22.15	130.00	1260.0	0.09831	0.10270	0.147900	0.094980	0.15
19	13.540	14.36	87.46	566.3	0.09779	0.08129	0.066640	0.047810	0.18
20	13.080	15.71	85.63	520.0	0.10750	0.12700	0.045680	0.031100	0.19
21	9.504	12.44	60.34	273.9	0.10240	0.06492	0.029560	0.020760	0.18
22	15.340	14.26	102.50	704.4	0.10730	0.21350	0.207700	0.097560	0.25
23	21.160	23.04	137.20	1404.0	0.09428	0.10220	0.109700	0.086320	0.17
24	16.650	21.38	110.00	904.6	0.11210	0.14570	0.152500	0.091700	0.19
25	17.140	16.40	116.00	912.7	0.11860	0.22760	0.222900	0.140100	0.30
26	14.580	21.53	97.41	644.8	0.10540	0.18680	0.142500	0.087830	0.22
27	18.610	20.25	122.10	1094.0	0.09440	0.10660	0.149000	0.077310	0.16
28	15.300	25.27	102.40	732.4	0.10820	0.16970	0.168300	0.087510	0.19
29	17.570	15.05	115.00	955.1	0.09847	0.11570	0.098750	0.079530	0.17
...	...	...	...	...	...	...	...	...	
539	7.691	25.44	48.34	170.4	0.08668	0.11990	0.092520	0.013640	0.20
540	11.540	14.44	74.65	402.9	0.09984	0.11200	0.067370	0.025940	0.18

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmet
541	14.470	24.99	95.81	656.4	0.08837	0.12300	0.100900	0.038900	0.187
542	14.740	25.42	94.70	668.6	0.08275	0.07214	0.041050	0.030270	0.184
543	13.210	28.06	84.88	538.4	0.08671	0.06877	0.029870	0.032750	0.162
544	13.870	20.70	89.77	584.8	0.09578	0.10180	0.036880	0.023690	0.162
545	13.620	23.23	87.19	573.2	0.09246	0.06747	0.029740	0.024430	0.166
546	10.320	16.35	65.31	324.9	0.09434	0.04994	0.010120	0.005495	0.188
547	10.260	16.58	65.85	320.8	0.08877	0.08066	0.043580	0.024380	0.166
548	9.683	19.34	61.05	285.7	0.08491	0.05030	0.023370	0.009615	0.158
549	10.820	24.21	68.89	361.6	0.08192	0.06602	0.015480	0.008160	0.197
550	10.860	21.48	68.51	360.5	0.07431	0.04227	0.000000	0.000000	0.166
551	11.130	22.44	71.49	378.4	0.09566	0.08194	0.048240	0.022570	0.203
552	12.770	29.43	81.35	507.9	0.08276	0.04234	0.019970	0.014990	0.153
553	9.333	21.94	59.01	264.0	0.09240	0.05605	0.039960	0.012820	0.169
554	12.880	28.92	82.50	514.3	0.08123	0.05824	0.061950	0.023430	0.156
555	10.290	27.61	65.67	321.4	0.09030	0.07658	0.059990	0.027380	0.159
556	10.160	19.59	64.73	311.7	0.10030	0.07504	0.005025	0.011160	0.179
557	9.423	27.88	59.26	271.3	0.08123	0.04971	0.000000	0.000000	0.174
558	14.590	22.68	96.39	657.1	0.08473	0.13300	0.102900	0.037360	0.145
559	11.510	23.93	74.52	403.5	0.09261	0.10210	0.111200	0.041050	0.138
560	14.050	27.15	91.38	600.4	0.09929	0.11260	0.044620	0.043040	0.153
561	11.200	29.37	70.67	386.0	0.07449	0.03558	0.000000	0.000000	0.106
562	15.220	30.62	103.40	716.9	0.10480	0.20870	0.255000	0.094290	0.212
563	20.920	25.09	143.00	1347.0	0.10990	0.22360	0.317400	0.147400	0.214
564	21.560	22.39	142.00	1479.0	0.11100	0.11590	0.243900	0.138900	0.172
565	20.130	28.25	131.20	1261.0	0.09780	0.10340	0.144000	0.097910	0.175
566	16.600	28.08	108.30	858.1	0.08455	0.10230	0.092510	0.053020	0.159
567	20.600	29.33	140.10	1265.0	0.11780	0.27700	0.351400	0.152000	0.239
568	7.760	24.54	47.92	181.0	0.05263	0.04362	0.000000	0.000000	0.158

569 rows × 30 columns

In [23]:

```
y = df_cancer['target']
y
# Define the output (Dependent variable)
```

Out[23]:
```
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0
5      0.0
6      0.0
7      0.0
8      0.0
9      0.0
10     0.0
11     0.0
12     0.0
13     0.0
14     0.0
15     0.0
16     0.0
17     0.0
18     0.0
19     1.0
20     1.0
21     1.0
22     0.0
23     0.0
24     0.0
25     0.0
26     0.0
27     0.0
28     0.0
29     0.0
      ...
539    1.0
540    1.0
541    1.0
542    1.0
543    1.0
544    1.0
545    1.0
546    1.0
547    1.0
548    1.0
549    1.0
550    1.0
551    1.0
552    1.0
553    1.0
554    1.0
555    1.0
556    1.0
557    1.0
558    1.0
559    1.0
560    1.0
561    1.0
562    0.0
563    0.0
564    0.0
```

```
565    0.0
566    0.0
567    0.0
568    1.0
Name: target, Length: 569, dtype: float64
```

In [24]:
```
from sklearn.model_selection import train_test_split # method we are goi
ng to be using
# Testing data set = model has not seen before
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
20, random_state=5)
```

In [25]:
```
X_train.shape
```

Out[25]:
```
(455, 30)
```

In [26]:
```
X_test.shape
```

Out[26]:
```
(114, 30)
```

In [27]:
```
y_train.shape
```

Out[27]:
```
(455,)
```

In [28]:
```
y_test.shape
```

Out[28]:
```
(114,)
```

In [29]:
```
from sklearn.svm import SVC # Class SVC
from sklearn.metrics import classification_report, confusion_matrix

svc_model = SVC() # Object, creating an object from the class
svc_model.fit(X_train, y_train) # Method on the object to train the mode
l
```

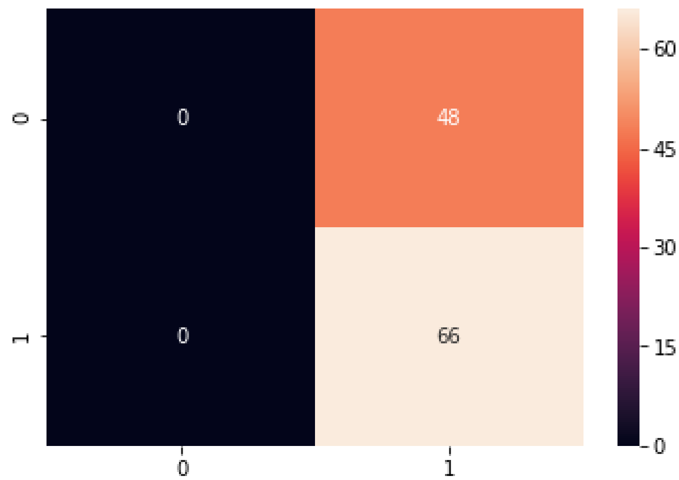
Out[29]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

STEP #5: EVALUATING THE MODEL

In [30]:
```
# Using testing data that the model has not seen before
# We want the model to be "generalized"
y_predict = svc_model.predict(X_test)
cm = confusion_matrix(y_test, y_predict)
```

In [31]:
```
sns.heatmap(cm, annot=True)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1c93face2b0>



In [32]:
```
print(classification_report(y_test, y_predict))
```

```
             precision    recall  f1-score   support

        0.0       0.00      0.00      0.00        48
        1.0       0.58      1.00      0.73        66

avg / total       0.34      0.58      0.42       114


C:\Users\Dr. Ryan\Anaconda3\lib\site-packages\sklearn\metrics\classific
ation.py:1135: UndefinedMetricWarning: Precision and F-score are ill-de
fined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

STEP #6: IMPROVING THE MODEL

In [33]:
```
min_train = X_train.min()
min_train
```

Out[33]:
```
mean radius                  6.981000
mean texture                 9.710000
mean perimeter              43.790000
mean area                  143.500000
mean smoothness              0.052630
mean compactness             0.019380
mean concavity               0.000000
mean concave points          0.000000
mean symmetry                0.106000
mean fractal dimension       0.049960
radius error                 0.111500
texture error                0.362100
perimeter error              0.757000
area error                   6.802000
smoothness error             0.001713
compactness error            0.002252
concavity error              0.000000
concave points error         0.000000
symmetry error               0.007882
fractal dimension error      0.000950
worst radius                 7.930000
worst texture               12.020000
worst perimeter             50.410000
worst area                 185.200000
worst smoothness             0.071170
worst compactness            0.027290
worst concavity              0.000000
worst concave points         0.000000
worst symmetry               0.156500
worst fractal dimension      0.055040
dtype: float64
```

In [34]:
```
range_train = (X_train - min_train).max()
range_train
```

Out[34]:
```
mean radius                   21.129000
mean texture                  29.570000
mean perimeter               144.710000
mean area                   2355.500000
mean smoothness                0.110770
mean compactness               0.326020
mean concavity                 0.426800
mean concave points            0.201200
mean symmetry                  0.198000
mean fractal dimension         0.045790
radius error                   2.761500
texture error                  4.522900
perimeter error               21.223000
area error                   518.798000
smoothness error               0.029417
compactness error              0.133148
concavity error                0.396000
concave points error           0.052790
symmetry error                 0.071068
fractal dimension error        0.028890
worst radius                  25.190000
worst texture                 37.520000
worst perimeter              170.390000
worst area                  3246.800000
worst smoothness               0.129430
worst compactness              1.030710
worst concavity                1.105000
worst concave points           0.291000
worst symmetry                 0.420900
worst fractal dimension        0.152460
dtype: float64
```

In [35]:
```
X_train_scaled = (X_train - min_train)/range_train
```

In [36]:
```
X_train_scaled
```

Out[36]:

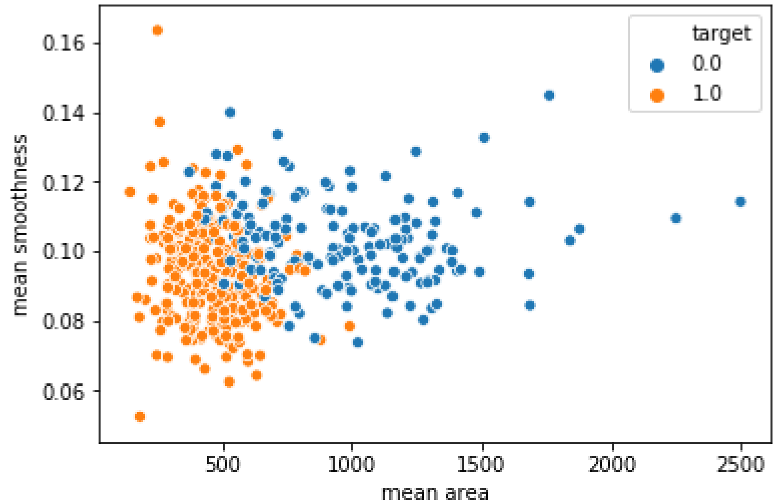
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
306	0.294335	0.206628	0.278350	0.167183	0.293220	0.101620	0.003423	0.016208
410	0.207251	0.265810	0.198328	0.108809	0.324546	0.103521	0.065206	0.104374
197	0.525297	0.410213	0.508673	0.373806	0.190304	0.205632	0.258435	0.287177
376	0.169861	0.355428	0.182157	0.082700	0.343956	0.449727	0.534208	0.295278
244	0.587770	0.466351	0.589524	0.429421	0.452018	0.418441	0.480084	0.441650
299	0.167022	0.452486	0.159353	0.080959	0.441184	0.149040	0.058458	0.093191
312	0.273510	0.123774	0.266049	0.153089	0.318769	0.184345	0.094939	0.126640
331	0.283923	0.326006	0.281459	0.157291	0.389636	0.285627	0.166518	0.146620
317	0.531923	0.309773	0.517656	0.375080	0.404712	0.283173	0.264761	0.395129
341	0.124237	0.241123	0.123350	0.058162	0.290512	0.223606	0.197329	0.113917
156	0.506366	0.373013	0.508673	0.348206	0.531462	0.451261	0.434630	0.523857
71	0.090255	0.166723	0.103656	0.042666	0.408053	0.410159	0.201640	0.142744
218	0.606702	0.400744	0.593670	0.461261	0.371942	0.341145	0.298032	0.431958
344	0.223816	0.194116	0.215880	0.117512	0.563059	0.163886	0.093861	0.161531
247	0.279663	0.148799	0.284431	0.156527	0.315699	0.353414	0.321931	0.197813
212	1.000000	0.296246	1.000000	1.000000	0.555836	0.405558	0.750000	0.792744
559	0.214350	0.480893	0.212356	0.110380	0.360928	0.253727	0.260544	0.204026
176	0.138341	0.282381	0.143805	0.067459	0.400469	0.337464	0.306232	0.184692
422	0.219083	0.213392	0.218851	0.112375	0.507087	0.298816	0.166284	0.223509
248	0.173648	0.524518	0.167369	0.086394	0.396678	0.162444	0.055740	0.080268
232	0.200625	0.815015	0.186580	0.103290	0.227228	0.050181	0.011638	0.031978
444	0.522931	0.241461	0.509364	0.359372	0.332581	0.318447	0.255389	0.310835
383	0.255999	0.262766	0.254647	0.135598	0.465559	0.338384	0.138051	0.143141
279	0.325098	0.184985	0.312349	0.188453	0.383949	0.176370	0.104944	0.184443
494	0.292442	0.366250	0.278281	0.167778	0.187054	0.102356	0.042174	0.062425
316	0.246060	0.147785	0.231221	0.134961	0.223075	0.039077	0.026312	0.025104
523	0.318472	0.303348	0.310552	0.181490	0.420060	0.268757	0.126172	0.188022
90	0.361541	0.483936	0.350909	0.220420	0.335019	0.204527	0.072680	0.146968
469	0.219556	0.286439	0.225209	0.112630	0.585628	0.395436	0.238988	0.276541
373	0.646457	0.258370	0.628913	0.505837	0.377629	0.270597	0.357779	0.444384
...	...	...	...	...	...	...	...	...
539	0.033603	0.531958	0.031442	0.011420	0.307394	0.308325	0.216776	0.067793
110	0.132330	0.246195	0.129293	0.062280	0.461045	0.198331	0.101546	0.088370

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
5	0.258839	0.202570	0.267984	0.141626	0.678613	0.461996	0.369728	0.402038
144	0.178380	0.177883	0.169097	0.089917	0.228401	0.098184	0.052741	0.039140
103	0.137015	0.327697	0.139313	0.065719	0.432157	0.237992	0.144189	0.150547
210	0.643618	0.420358	0.628222	0.486733	0.345491	0.354027	0.384255	0.475199
446	0.509679	0.619547	0.507981	0.355806	0.427372	0.343599	0.397844	0.412177
41	0.187846	0.393642	0.194251	0.096625	0.632572	0.314153	0.244611	0.281759
362	0.273510	0.308759	0.263147	0.149904	0.398393	0.184467	0.062980	0.088519
377	0.306640	0.625634	0.290927	0.177712	0.203485	0.085516	0.029780	0.055517
254	0.590137	0.325330	0.571557	0.435364	0.459240	0.304951	0.323102	0.426988
146	0.228075	0.232330	0.243245	0.122479	0.509795	0.461996	0.388707	0.368539
86	0.354915	0.397362	0.348697	0.214264	0.377449	0.245660	0.282099	0.245427
542	0.367220	0.531282	0.351807	0.222925	0.271915	0.161831	0.096181	0.150447
431	0.256472	0.269530	0.260383	0.137678	0.476393	0.344212	0.181373	0.139115
65	0.369114	0.481231	0.370465	0.222798	0.582920	0.394209	0.296860	0.448757
205	0.385205	0.235712	0.380001	0.243303	0.326171	0.234648	0.176898	0.202734
44	0.292915	0.409199	0.287679	0.164721	0.401824	0.261702	0.193510	0.261034
27	0.550381	0.356442	0.541151	0.403524	0.377088	0.267530	0.349110	0.384245
80	0.211510	0.380791	0.207449	0.109531	0.519726	0.227716	0.107568	0.110984
437	0.334091	0.212039	0.317808	0.198557	0.288435	0.121373	0.082802	0.146322
113	0.167022	0.354413	0.171723	0.080959	0.537781	0.340225	0.151734	0.152485
204	0.259785	0.300643	0.257757	0.143664	0.424483	0.265076	0.187559	0.189911
519	0.273037	0.236388	0.267570	0.148716	0.540489	0.283173	0.090909	0.148857
411	0.192106	0.240785	0.187478	0.097516	0.497156	0.179928	0.071368	0.123260
8	0.284869	0.409537	0.302052	0.159754	0.674099	0.533157	0.435567	0.464861
73	0.322732	0.205614	0.322300	0.187052	0.433962	0.333170	0.182498	0.251938
400	0.517251	0.382482	0.557045	0.361070	0.635280	0.730691	0.747188	0.595427
118	0.416442	0.446398	0.427821	0.271322	0.567572	0.477946	0.499766	0.471123
206	0.137015	0.255665	0.132195	0.064487	0.507990	0.162383	0.041143	0.097018

455 rows × 30 columns

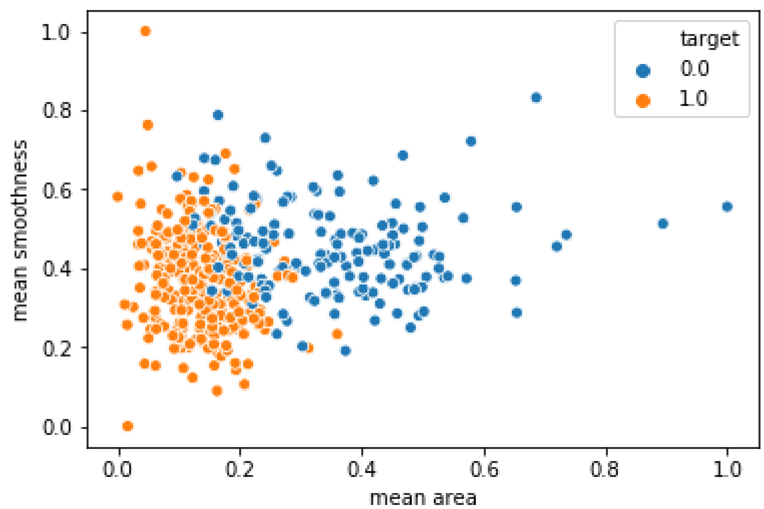
In [37]:
```
sns.scatterplot(x = X_train['mean area'], y = X_train['mean smoothness'], hue = y_train)
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x1c93fb7dac8>



In [38]:
```
sns.scatterplot(x = X_train_scaled['mean area'], y = X_train_scaled['mean smoothness'], hue = y_train)
```

Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x1c93fc637f0>



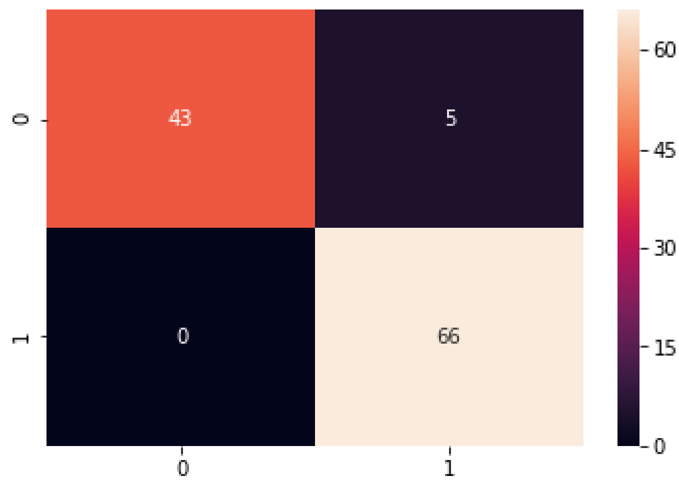
In [39]:
```
min_test = X_test.min()
range_test = (X_test - min_test).max()
X_test_scaled = (X_test - min_test)/range_test
```

In [40]:
```python
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

svc_model = SVC()
svc_model.fit(X_train_scaled, y_train) # Fitting the model to the traini
ng set
```

Out[40]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [41]:
```python
y_predict = svc_model.predict(X_test_scaled) # Using normalized data
cm = confusion_matrix(y_test, y_predict)

sns.heatmap(cm,annot=True,fmt="d")
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c94022fc50>
```



In [42]:
```python
print(classification_report(y_test,y_predict))
```

```
             precision    recall  f1-score   support

        0.0       1.00      0.90      0.95        48
        1.0       0.93      1.00      0.96        66

avg / total       0.96      0.96      0.96       114
```

IMPROVING THE MODEL - PART 2

In [43]:
```python
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'k
ernel': ['rbf']} # Specifying range of values
```

In [44]:
```python
from sklearn.model_selection import GridSearchCV
```

In [45]:

```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=4)
```

In [46]:
```
grid.fit(X_train_scaled,y_train)
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] C=0.1, gamma=1, kernel=rbf .......................................
[CV]  C=0.1, gamma=1, kernel=rbf, score=0.9671052631578947, total=   0.
0s
[CV] C=0.1, gamma=1, kernel=rbf .......................................
[CV]  C=0.1, gamma=1, kernel=rbf, score=0.9210526315789473, total=   0.
0s
[CV] C=0.1, gamma=1, kernel=rbf .......................................
[CV]  C=0.1, gamma=1, kernel=rbf, score=0.9470198675496688, total=   0.
0s
[CV] C=0.1, gamma=0.1, kernel=rbf .....................................
[CV]  C=0.1, gamma=0.1, kernel=rbf, score=0.9144736842105263, total=
0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf .....................................
[CV]  C=0.1, gamma=0.1, kernel=rbf, score=0.8881578947368421, total=
0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf .....................................
[CV]  C=0.1, gamma=0.1, kernel=rbf, score=0.8675496688741722, total=
0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ....................................
[CV]  C=0.1, gamma=0.01, kernel=rbf, score=0.6381578947368421, total=
0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ....................................
[CV]  C=0.1, gamma=0.01, kernel=rbf, score=0.6381578947368421, total=
0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ....................................
[CV]  C=0.1, gamma=0.01, kernel=rbf, score=0.6423841059602649, total=
0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf ...................................
[CV]  C=0.1, gamma=0.001, kernel=rbf, score=0.6381578947368421, total=
  0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf ...................................
[CV]  C=0.1, gamma=0.001, kernel=rbf, score=0.6381578947368421, total=
  0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf ...................................
[CV]  C=0.1, gamma=0.001, kernel=rbf, score=0.6423841059602649, total=
  0.0s
[CV] C=1, gamma=1, kernel=rbf .........................................
[CV]  C=1, gamma=1, kernel=rbf, score=0.993421052631579, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf .........................................
[CV]  C=1, gamma=1, kernel=rbf, score=0.9473684210526315, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf .........................................
[CV]  C=1, gamma=1, kernel=rbf, score=0.9801324503311258, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf .......................................
[CV]  C=1, gamma=0.1, kernel=rbf, score=0.9736842105263158, total=   0.
0s
[CV] C=1, gamma=0.1, kernel=rbf .......................................
[CV]  C=1, gamma=0.1, kernel=rbf, score=0.9276315789473685, total=   0.
0s
[CV] C=1, gamma=0.1, kernel=rbf .......................................
[CV]  C=1, gamma=0.1, kernel=rbf, score=0.9403973509933775, total=   0.
0s
[CV] C=1, gamma=0.01, kernel=rbf ......................................
[CV]  C=1, gamma=0.01, kernel=rbf, score=0.9144736842105263, total=
0.0s
[CV] C=1, gamma=0.01, kernel=rbf ......................................
```

```
[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=1, gamma=0.01, kernel=rbf, score=0.8947368421052632, total=
0.0s
[CV] C=1, gamma=0.01, kernel=rbf ....................................
[CV]  C=1, gamma=0.01, kernel=rbf, score=0.8675496688741722, total=
0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV]  C=1, gamma=0.001, kernel=rbf, score=0.6381578947368421, total=
0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV]  C=1, gamma=0.001, kernel=rbf, score=0.6381578947368421, total=
0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV]  C=1, gamma=0.001, kernel=rbf, score=0.6423841059602649, total=
0.0s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV]  C=10, gamma=1, kernel=rbf, score=0.993421052631579, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV]  C=10, gamma=1, kernel=rbf, score=0.9605263157894737, total=   0.0
s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV]  C=10, gamma=1, kernel=rbf, score=0.9735099337748344, total=   0.0
s
[CV] C=10, gamma=0.1, kernel=rbf ....................................
[CV]  C=10, gamma=0.1, kernel=rbf, score=0.993421052631579, total=   0.
0s
[CV] C=10, gamma=0.1, kernel=rbf ....................................
[CV]  C=10, gamma=0.1, kernel=rbf, score=0.9671052631578947, total=
0.0s
[CV] C=10, gamma=0.1, kernel=rbf ....................................
[CV]  C=10, gamma=0.1, kernel=rbf, score=0.9735099337748344, total=
0.0s
[CV] C=10, gamma=0.01, kernel=rbf ...................................
[CV]  C=10, gamma=0.01, kernel=rbf, score=0.9736842105263158, total=
0.0s
[CV] C=10, gamma=0.01, kernel=rbf ...................................
[CV]  C=10, gamma=0.01, kernel=rbf, score=0.9210526315789473, total=
0.0s
[CV] C=10, gamma=0.01, kernel=rbf ...................................
[CV]  C=10, gamma=0.01, kernel=rbf, score=0.9403973509933775, total=
0.0s
[CV] C=10, gamma=0.001, kernel=rbf ..................................
[CV]  C=10, gamma=0.001, kernel=rbf, score=0.9144736842105263, total=
0.0s
[CV] C=10, gamma=0.001, kernel=rbf ..................................
[CV]  C=10, gamma=0.001, kernel=rbf, score=0.8947368421052632, total=
0.0s
[CV] C=10, gamma=0.001, kernel=rbf ..................................
[CV]  C=10, gamma=0.001, kernel=rbf, score=0.8675496688741722, total=
0.0s
[CV] C=100, gamma=1, kernel=rbf .....................................
[CV]  C=100, gamma=1, kernel=rbf, score=0.9605263157894737, total=   0.
0s
[CV] C=100, gamma=1, kernel=rbf .....................................
[CV]  C=100, gamma=1, kernel=rbf, score=0.9539473684210527, total=   0.
0s
[CV] C=100, gamma=1, kernel=rbf .....................................
[CV]  C=100, gamma=1, kernel=rbf, score=0.9801324503311258, total=   0.
```

```
0s
[CV] C=100, gamma=0.1, kernel=rbf ....................................
[CV]  C=100, gamma=0.1, kernel=rbf, score=0.9868421052631579, total=
0.0s
[CV] C=100, gamma=0.1, kernel=rbf ....................................
[CV]  C=100, gamma=0.1, kernel=rbf, score=0.9539473684210527, total=
0.0s
[CV] C=100, gamma=0.1, kernel=rbf ....................................
[CV]  C=100, gamma=0.1, kernel=rbf, score=0.9801324503311258, total=
0.0s
[CV] C=100, gamma=0.01, kernel=rbf ...................................
[CV]  C=100, gamma=0.01, kernel=rbf, score=0.993421052631579, total=
0.0s
[CV] C=100, gamma=0.01, kernel=rbf ...................................
[CV]  C=100, gamma=0.01, kernel=rbf, score=0.9671052631578947, total=
0.0s
[CV] C=100, gamma=0.01, kernel=rbf ...................................
[CV]  C=100, gamma=0.01, kernel=rbf, score=0.9735099337748344, total=
0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV]  C=100, gamma=0.001, kernel=rbf, score=0.9736842105263158, total=
  0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV]  C=100, gamma=0.001, kernel=rbf, score=0.9210526315789473, total=
  0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV]  C=100, gamma=0.001, kernel=rbf, score=0.9403973509933775, total=
  0.0s

[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:    0.3s finished
```

Out[46]:
```
GridSearchCV(cv=None, error_score='raise',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.
0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.00
1], 'kernel': ['rbf']},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=4)
```

In [47]:
```
grid.best_params_
```

Out[47]:
```
{'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
```

In [48]:
```
grid.best_estimator_
```

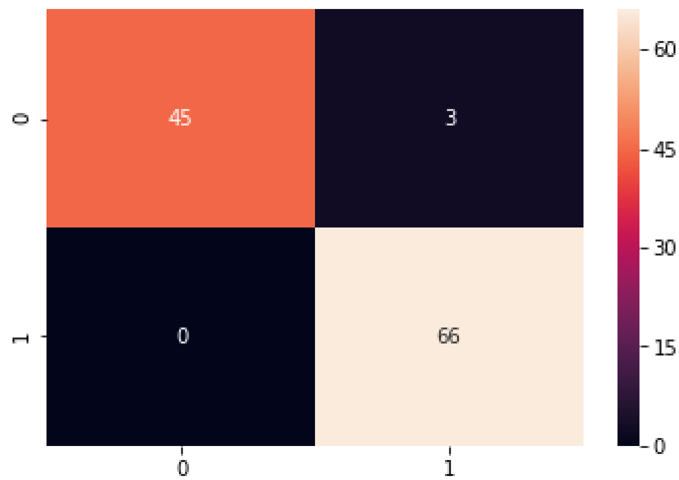
Out[48]:
```
SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [49]:
```
grid_predictions = grid.predict(X_test_scaled)
# Grid_predcitions replaced y_predict
```

In [50]:
```
cm = confusion_matrix(y_test, grid_predictions)
```

In [51]:
```
sns.heatmap(cm, annot=True)
```

Out[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1c94162d400>
```



In [52]:
```
print(classification_report(y_test,grid_predictions))
```

```
             precision    recall  f1-score   support

        0.0       1.00      0.94      0.97        48
        1.0       0.96      1.00      0.98        66

avg / total       0.97      0.97      0.97       114
```