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

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

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

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

Out[4]:
```
{'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_dataset:\n\nBreast cancer wisconsin (diagn
ostic) dataset\n--------------------------------------------\n\n**Data
Set Characteristics:**\n\n    :Number of Instances: 569\n\n    :Number
of Attributes: 30 numeric, 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 g
ray-scale values)\n        - perimeter\n        - area\n        - smoot
hness (local variation in radius lengths)\n        - compactness (perim
eter^2 / area - 1.0)\n        - concavity (severity of concave portions
of the contour)\n        - concave points (number of concave portions o
f the contour)\n        - symmetry \n        - fractal dimension ("coas
tline approximation" - 1)\n\n        The mean, standard error, and "wor
st" or largest (mean of the three\n        largest values) of these fea
tures were computed for each image,\n        resulting in 30 features.
For instance, field 3 is Mean Radius, field\n        13 is Radius SE, f
ield 23 is Worst Radius.\n\n        - class:\n                - WDBC-Ma
lignant\n                - WDBC-Benign\n\n    :Summary Statistics:\n\n
   ===================================== ====== ======\n
                            Min    Max\n    ===========================
========== ====== ======\n    radius (mean):                        6.9
81  28.11\n    texture (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    concavity (mean):                     0.0    0.427\n    conc
ave points (mean):                0.0    0.201\n    symmetry (mean):
                   0.106  0.304\n    fractal dimension (mean):
    0.05   0.097\n    radius (standard error):              0.112  2.87
3\n    texture (standard error):             0.36   4.885\n    perimete
r (standard error):           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 (standard error):           0.0    0.396\n    concave poin
ts (standard error):      0.0    0.053\n    symmetry (standard error):
           0.008  0.079\n    fractal dimension (standard error):   0.00
1  0.03\n    radius (worst):                       7.93   36.04\n    te
```

```
xture (worst):                      12.02  49.54\n    perimeter (wors
t):                     50.41  251.2\n    area (worst):
        185.2  4254.0\n    smoothness (worst):                  0.071
0.223\n    compactness (worst):                  0.027  1.058\n    conc
avity (worst):                    0.0    1.252\n    concave points (wor
st):                 0.0    0.291\n    symmetry (worst):
    0.156  0.664\n    fractal dimension (worst):            0.055  0.20
8\n    ===================================== ====== ======\n\n    :Miss
ing Attribute Values: None\n\n    :Class Distribution: 212 - Malignant,
357 - Benign\n\n    :Creator:  Dr. William H. Wolberg, W. Nick Street,
Olvi L. Mangasarian\n\n    :Donor: Nick Street\n\n    :Date: November,
1995\n\nThis is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) d
atasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitiz
ed image of a fine needle\naspirate (FNA) of a breast mass.  They descr
ibe\ncharacteristics of the cell nuclei present in the image.\n\nSepara
ting plane described above was obtained using\nMultisurface Method-Tree
(MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Program
ming." Proceedings of the 4th\nMidwest Artificial Intelligence and Cogn
itive Science Society,\npp. 97-101, 1992], a classification method whic
h uses linear\nprogramming to construct a decision tree.  Relevant feat
ures\nwere selected using an exhaustive search in the space of 1-4\nfea
tures and 1-3 separating planes.\n\nThe actual linear program used to o
btain the separating plane\nin the 3-dimensional space is that describe
d in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgrammin
g Discrimination of Two Linearly Inseparable Sets",\nOptimization Metho
ds and Software 1, 1992, 23-34].\n\nThis database is also available thr
ough the UW CS ftp server:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dat
aset/machine-learn/WDBC/\n\n.. topic:: References\n\n   - W.N. Street,
W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction \n     fo
r breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on \n
    Electronic Imaging: Science and 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 p
rogramming. Operations Research, 43(4), pages 570-577, \n     July-Augu
st 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine
learning techniques\n     to diagnose breast cancer from fine-needle as
pirates. Cancer Letters 77 (1994) \n     163-171.',
 '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'),
 'filename': 'C:\\Users\\XYZ\\Anaconda4\\lib\\site-packages\\sklearn\\d
atasets\\data\\breast_cancer.csv'}
```

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

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

In [6]:
```
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 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 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]
```

Features in the Dataset

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

```
(569, 30)
```

In [9]:
```
df_cancer=pd.DataFrame(np.c_[cancer['data'],cancer['target']] ,columns=
np.append(cancer['feature_names'],['target']))
```

In [10]:
```
df_cancer.head()
```

Out[10]:

	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 [11]:
```
df_cancer.tail()
```

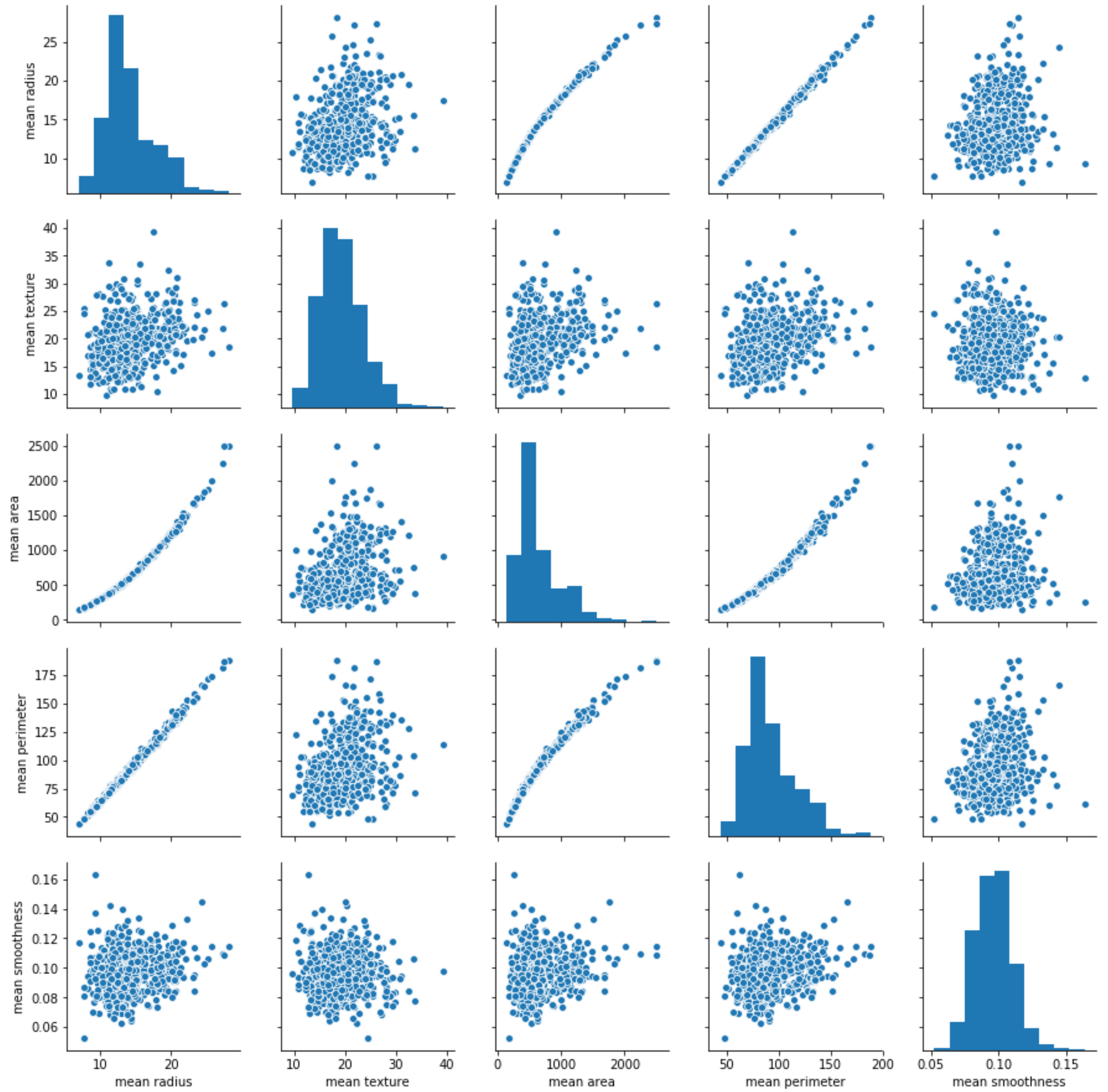
Out[11]:

	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 [12]:
```
sns.pairplot(df_cancer, vars=['mean radius','mean texture','mean area',
'mean perimeter','mean smoothness'])
```

Out[12]: <seaborn.axisgrid.PairGrid at 0x11e18d7d748>

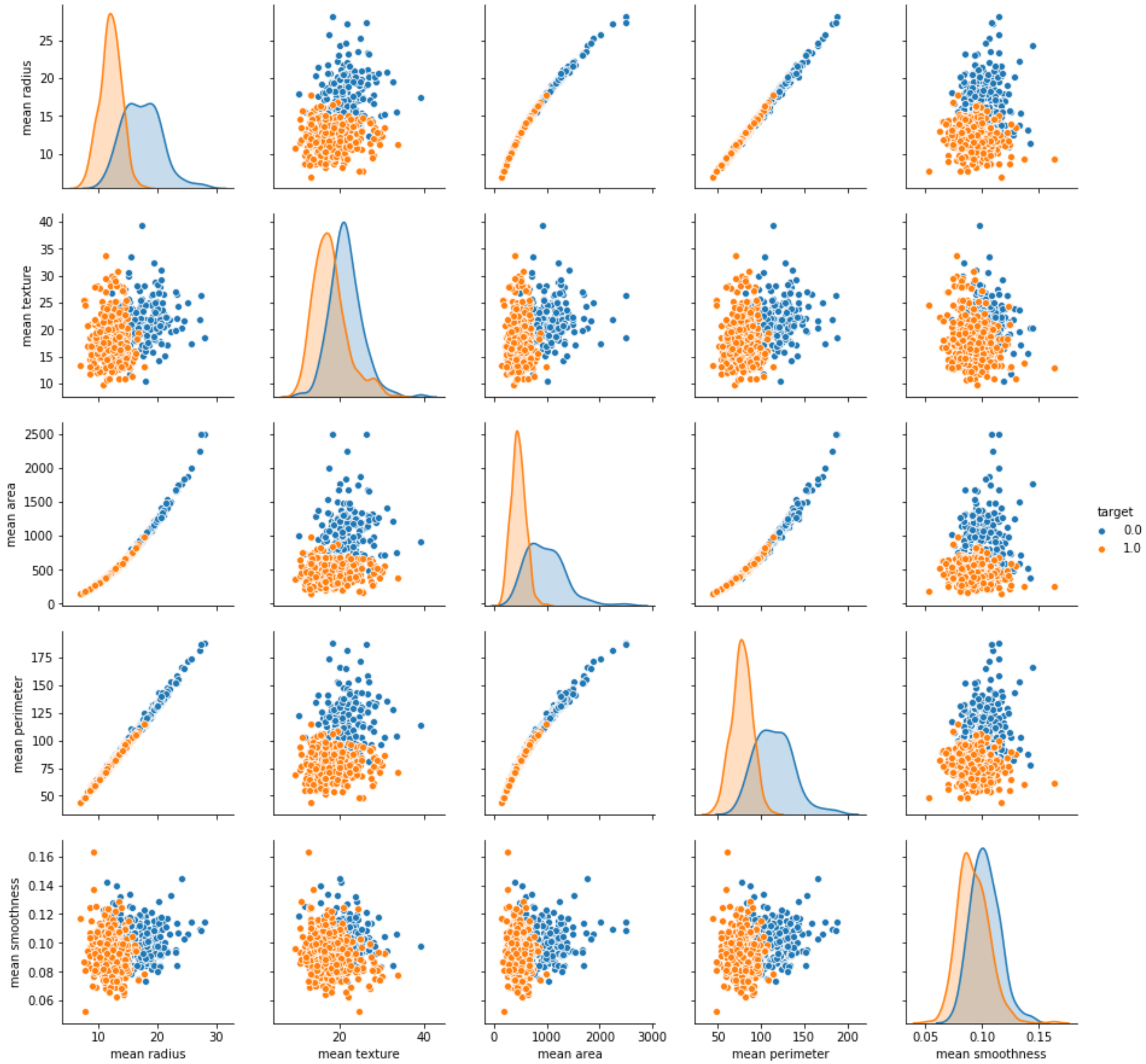


In [13]:
```
sns.pairplot(df_cancer, hue='target' ,vars=['mean radius','mean texture','mean area','mean perimeter','mean smoothness'])
```

```
C:\Users\XYZ\Anaconda4\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

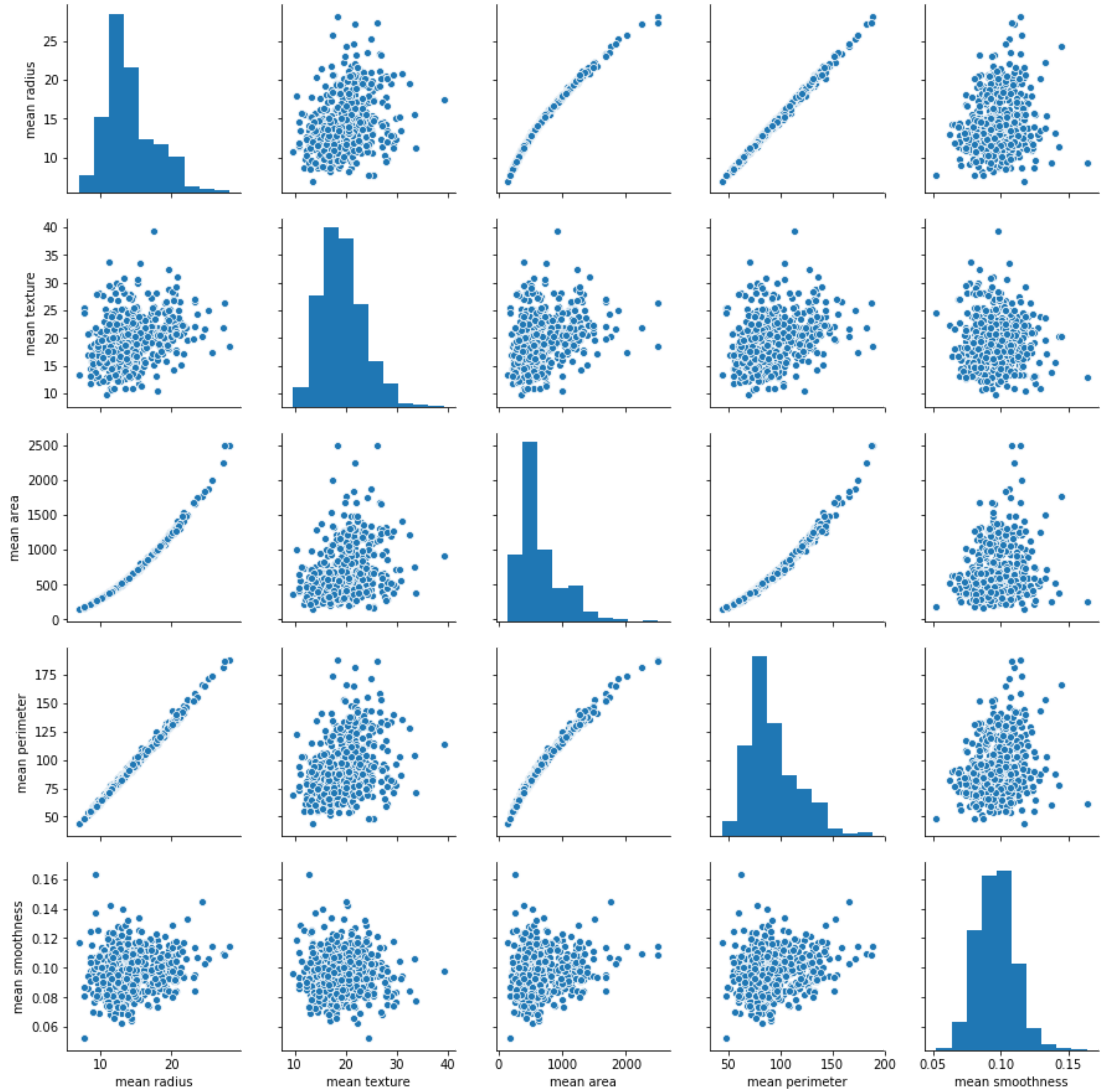
Out[13]:

```
<seaborn.axisgrid.PairGrid at 0x11e1b8e22b0>
```



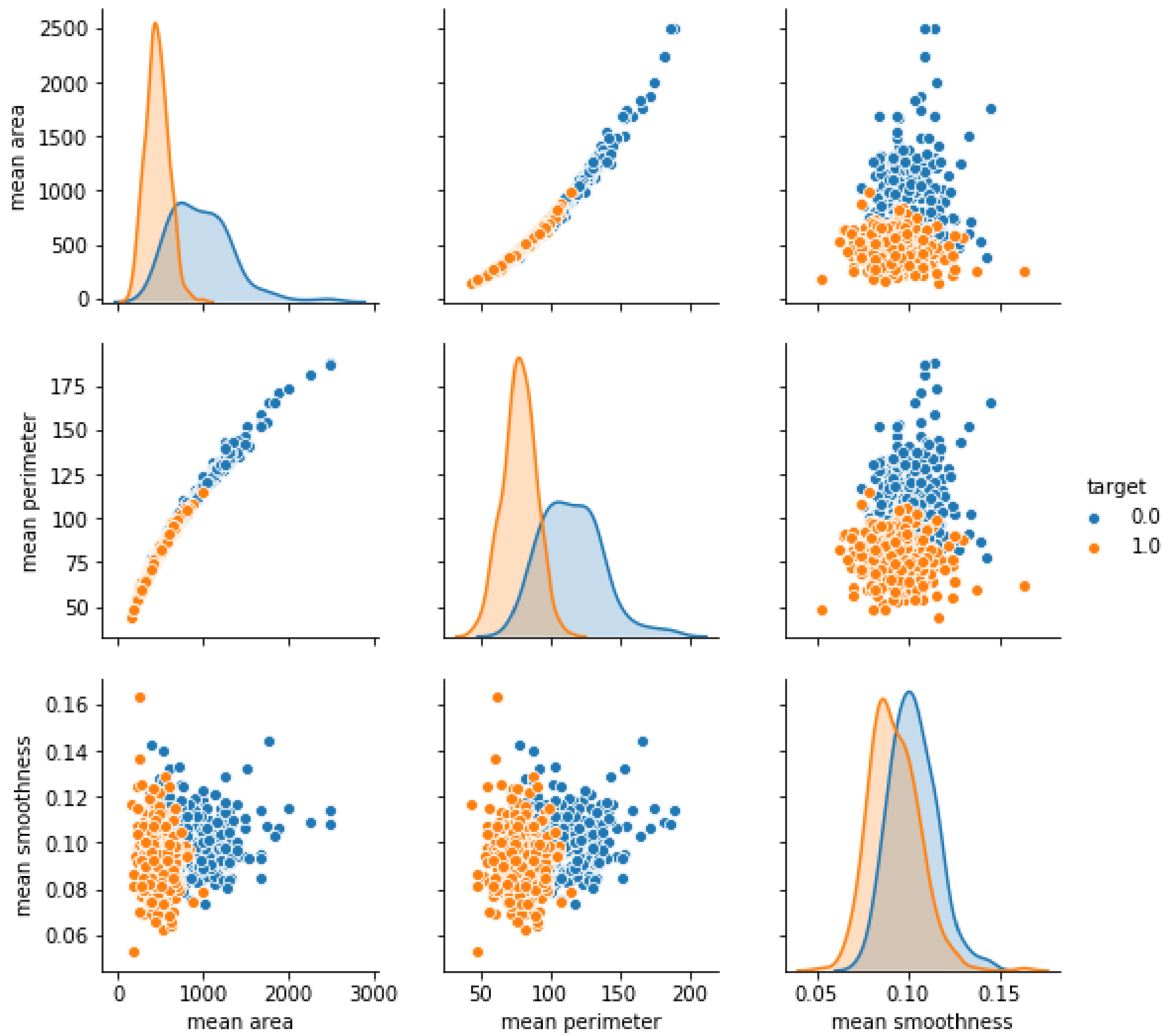
In [14]:
```
sns.pairplot(df_cancer, vars=['mean radius','mean texture','mean area',
'mean perimeter','mean smoothness'])
```

Out[14]: <seaborn.axisgrid.PairGrid at 0x11e1c9ab080>



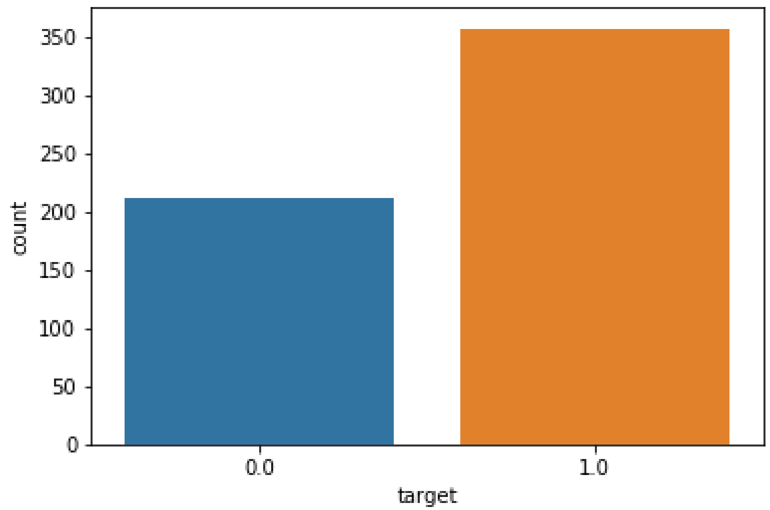
In [15]:
```
sns.pairplot(df_cancer, hue='target' ,vars=['mean area','mean perimete
r','mean smoothness'])
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x11e1ed31898>



In [16]:
```
sns.countplot(df_cancer['target'])
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x11e1f7cbdd8>

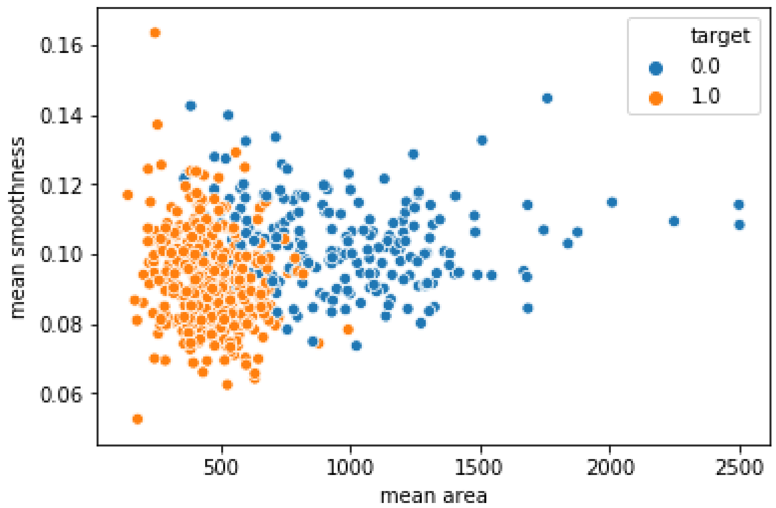


Plot of figures

1==> Benign cancer 0==>malignant cancer

In [17]:
```
sns.scatterplot(x='mean area' ,y='mean smoothness',hue='target', data=d
f_cancer)
```

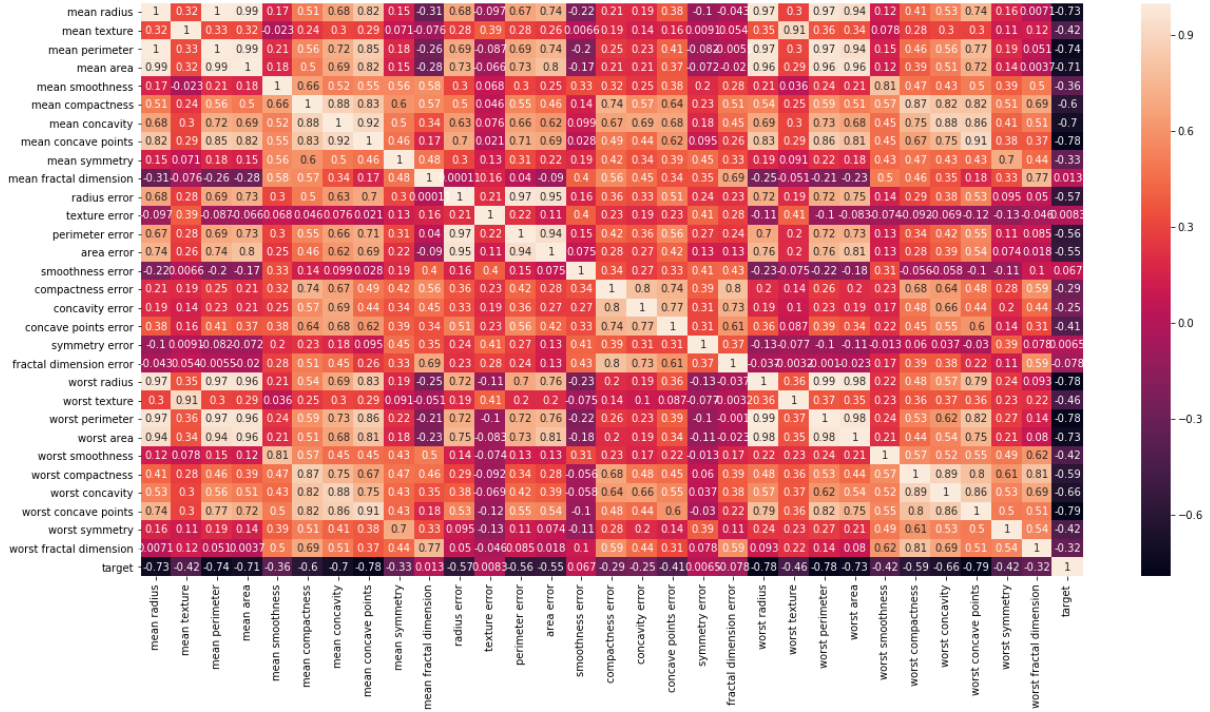
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x11e1f99e828>



Corelation Between Features of the Dataset

In [18]:
```
plt.figure(figsize=(20,10))
sns.heatmap(df_cancer.corr(),annot=True)
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e1f7efcf8>
```



In [19]:
```
X=df_cancer.drop(['target'],axis=1)
```

In [20]:
```
X
```

Out[20]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
0	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300100	0.147100	0.241
1	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900	0.070170	0.181
2	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197400	0.127900	0.206
3	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241400	0.105200	0.259
4	20.290	14.34	135.10	1297.0	0.10030	0.13280	0.198000	0.104300	0.180

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
5	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157800	0.080890	0.208
6	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112700	0.074000	0.179
7	13.710	20.83	90.20	577.9	0.11890	0.16450	0.093660	0.059850	0.219
8	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185900	0.093530	0.235
9	12.460	24.04	83.97	475.9	0.11860	0.23960	0.227300	0.085430	0.203
10	16.020	23.24	102.70	797.8	0.08206	0.06669	0.032990	0.033230	0.152
11	15.780	17.89	103.60	781.0	0.09710	0.12920	0.099540	0.066060	0.184
12	19.170	24.80	132.40	1123.0	0.09740	0.24580	0.206500	0.111800	0.239
13	15.850	23.95	103.70	782.7	0.08401	0.10020	0.099380	0.053640	0.184
14	13.730	22.61	93.60	578.3	0.11310	0.22930	0.212800	0.080250	0.206
15	14.540	27.54	96.73	658.8	0.11390	0.15950	0.163900	0.073640	0.230
16	14.680	20.13	94.74	684.5	0.09867	0.07200	0.073950	0.052590	0.158
17	16.130	20.68	108.10	798.8	0.11700	0.20220	0.172200	0.102800	0.216
18	19.810	22.15	130.00	1260.0	0.09831	0.10270	0.147900	0.094980	0.158
19	13.540	14.36	87.46	566.3	0.09779	0.08129	0.066640	0.047810	0.188
20	13.080	15.71	85.63	520.0	0.10750	0.12700	0.045680	0.031100	0.196
21	9.504	12.44	60.34	273.9	0.10240	0.06492	0.029560	0.020760	0.181
22	15.340	14.26	102.50	704.4	0.10730	0.21350	0.207700	0.097560	0.252
23	21.160	23.04	137.20	1404.0	0.09428	0.10220	0.109700	0.086320	0.176
24	16.650	21.38	110.00	904.6	0.11210	0.14570	0.152500	0.091700	0.199
25	17.140	16.40	116.00	912.7	0.11860	0.22760	0.222900	0.140100	0.304
26	14.580	21.53	97.41	644.8	0.10540	0.18680	0.142500	0.087830	0.225
27	18.610	20.25	122.10	1094.0	0.09440	0.10660	0.149000	0.077310	0.169
28	15.300	25.27	102.40	732.4	0.10820	0.16970	0.168300	0.087510	0.192

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
29	17.570	15.05	115.00	955.1	0.09847	0.11570	0.098750	0.079530	0.173
...	...	...	...	...	...	...	...	...	.
539	7.691	25.44	48.34	170.4	0.08668	0.11990	0.092520	0.013640	0.203
540	11.540	14.44	74.65	402.9	0.09984	0.11200	0.067370	0.025940	0.181
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

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
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 [21]:
```
y=df_cancer['target']
```

Class Labeled Tuple

In [22]:
```
y
```

Out[22]:
```
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
```

Training of Svm classifier

In [23]:
```
from sklearn.model_selection import train_test_split
```

In [24]:
```
train_test_split
```

Out[24]:
```
<function sklearn.model_selection._split.train_test_split(*arrays, **op
tions)>
```

In [25]:
```
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.20, random_state=5)
```

In [26]:
```
X_train
```

Out[26]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
306	13.200	15.82	84.07	537.3	0.08511	0.05251	0.001461	0.003261	0.163
410	11.360	17.57	72.49	399.8	0.08858	0.05313	0.027830	0.021000	0.160
197	18.080	21.84	117.40	1024.0	0.07371	0.08642	0.110300	0.057780	0.177
376	10.570	20.22	70.15	338.3	0.09073	0.16600	0.228000	0.059410	0.218

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
244	19.400	23.50	129.10	1155.0	0.10270	0.15580	0.204900	0.088860	0.197
299	10.510	23.09	66.85	334.2	0.10150	0.06797	0.024950	0.018750	0.169
312	12.760	13.37	82.29	504.1	0.08794	0.07948	0.040520	0.025480	0.160
331	12.980	19.35	84.52	514.0	0.09579	0.11250	0.071070	0.029500	0.176
317	18.220	18.87	118.70	1027.0	0.09746	0.11170	0.113000	0.079500	0.180
341	9.606	16.84	61.64	280.5	0.08481	0.09228	0.084220	0.022920	0.203
156	17.680	20.74	117.40	963.7	0.11150	0.16650	0.185500	0.105400	0.197
71	8.888	14.64	58.79	244.0	0.09783	0.15310	0.086060	0.028720	0.190
218	19.800	21.56	129.70	1230.0	0.09383	0.13060	0.127200	0.086910	0.209
344	11.710	15.45	75.03	420.3	0.11500	0.07281	0.040060	0.032500	0.200
247	12.890	14.11	84.95	512.2	0.08760	0.13460	0.137400	0.039800	0.159
212	28.110	18.47	188.50	2499.0	0.11420	0.15160	0.320100	0.159500	0.164
559	11.510	23.93	74.52	403.5	0.09261	0.10210	0.111200	0.041050	0.138
176	9.904	18.06	64.60	302.4	0.09699	0.12940	0.130700	0.037160	0.166
422	11.610	16.02	75.46	408.2	0.10880	0.11680	0.070970	0.044970	0.188
248	10.650	25.22	68.01	347.0	0.09657	0.07234	0.023790	0.016150	0.189
232	11.220	33.81	70.79	386.8	0.07780	0.03574	0.004967	0.006434	0.184
444	18.030	16.85	117.50	990.0	0.08947	0.12320	0.109000	0.062540	0.172
383	12.390	17.48	80.64	462.9	0.10420	0.12970	0.058920	0.028800	0.177
279	13.850	15.18	88.99	587.4	0.09516	0.07688	0.044790	0.037110	0.211
494	13.160	20.54	84.06	538.7	0.07335	0.05275	0.018000	0.012560	0.171
316	12.180	14.08	77.25	461.4	0.07734	0.03212	0.011230	0.005051	0.167
523	13.710	18.68	88.73	571.0	0.09916	0.10700	0.053850	0.037830	0.171
90	14.620	24.02	94.57	662.7	0.08974	0.08606	0.031020	0.029570	0.168

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
469	11.620	18.18	76.38	408.8	0.11750	0.14830	0.102000	0.055640	0.195
373	20.640	17.35	134.80	1335.0	0.09446	0.10760	0.152700	0.089410	0.157
...	...	...	...	...	...	...	...	...	.
539	7.691	25.44	48.34	170.4	0.08668	0.11990	0.092520	0.013640	0.203
110	9.777	16.99	62.50	290.2	0.10370	0.08404	0.043340	0.017780	0.158
5	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157800	0.080890	0.208
144	10.750	14.97	68.26	355.3	0.07793	0.05139	0.022510	0.007875	0.139
103	9.876	19.40	63.95	298.3	0.10050	0.09697	0.061540	0.030290	0.194
210	20.580	22.14	134.70	1290.0	0.09090	0.13480	0.164000	0.095610	0.176
446	17.750	28.03	117.30	981.6	0.09997	0.13140	0.169800	0.082930	0.171
41	10.950	21.35	71.90	371.1	0.12270	0.12180	0.104400	0.056690	0.189
362	12.760	18.84	81.87	496.6	0.09676	0.07952	0.026880	0.017810	0.175
377	13.460	28.21	85.89	562.1	0.07517	0.04726	0.012710	0.011170	0.142
254	19.450	19.33	126.50	1169.0	0.10350	0.11880	0.137900	0.085910	0.177
146	11.800	16.58	78.99	432.0	0.10910	0.17000	0.165900	0.074150	0.267
86	14.480	21.46	94.25	648.2	0.09444	0.09947	0.120400	0.049380	0.207
542	14.740	25.42	94.70	668.6	0.08275	0.07214	0.041050	0.030270	0.184
431	12.400	17.68	81.47	467.8	0.10540	0.13160	0.077410	0.027990	0.181
65	14.780	23.94	97.40	668.3	0.11720	0.14790	0.126700	0.090290	0.195
205	15.120	16.68	98.78	716.6	0.08876	0.09588	0.075500	0.040790	0.159
44	13.170	21.81	85.42	531.5	0.09714	0.10470	0.082590	0.052520	0.174
27	18.610	20.25	122.10	1094.0	0.09440	0.10660	0.149000	0.077310	0.169
80	11.450	20.97	73.81	401.5	0.11020	0.09362	0.045910	0.022330	0.184
437	14.040	15.98	89.78	611.2	0.08458	0.05895	0.035340	0.029440	0.171

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
113	10.510	20.19	68.64	334.2	0.11220	0.13030	0.064760	0.030680	0.192
204	12.470	18.60	81.09	481.9	0.09965	0.10580	0.080050	0.038210	0.192
519	12.750	16.70	82.51	493.8	0.11250	0.11170	0.038800	0.029950	0.212
411	11.040	16.83	70.92	373.2	0.10770	0.07804	0.030460	0.024800	0.171
8	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185900	0.093530	0.235
73	13.800	15.79	90.43	584.1	0.10070	0.12800	0.077890	0.050690	0.166
400	17.910	21.02	124.40	994.0	0.12300	0.25760	0.318900	0.119800	0.211
118	15.780	22.91	105.70	782.6	0.11550	0.17520	0.213300	0.094790	0.209
206	9.876	17.27	62.92	295.4	0.10890	0.07232	0.017560	0.019520	0.193

455 rows × 30 columns

In [27]:

```
y_train
```

Out[27]:

```
306    1.0
410    1.0
197    0.0
376    1.0
244    0.0
299    1.0
312    1.0
331    1.0
317    0.0
341    1.0
156    0.0
71     1.0
218    0.0
344    1.0
247    1.0
212    0.0
```

```
559    1.0
176    1.0
422    1.0
248    1.0
232    1.0
444    0.0
383    1.0
279    1.0
494    1.0
316    1.0
523    1.0
90     1.0
469    1.0
373    0.0
      ...
539    1.0
110    1.0
5      0.0
144    1.0
103    1.0
210    0.0
446    0.0
41     0.0
362    1.0
377    1.0
254    0.0
146    0.0
86     0.0
542    1.0
431    1.0
65     0.0
205    0.0
44     0.0
27     0.0
80     1.0
437    1.0
113    1.0
204    1.0
519    1.0
```

```
411    1.0
8      0.0
73     0.0
400    0.0
118    0.0
206    1.0
Name: target, Length: 455, dtype: float64
```

20% Test Sample

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

Out[28]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
28	15.300	25.27	102.40	732.4	0.10820	0.16970	0.168300	0.087510	0.192
163	12.340	22.22	79.85	464.5	0.10120	0.10150	0.053700	0.028220	0.155
123	14.500	10.89	94.28	640.7	0.11010	0.10990	0.088420	0.057780	0.185
361	13.300	21.57	85.24	546.1	0.08582	0.06373	0.033440	0.024240	0.181
549	10.820	24.21	68.89	361.6	0.08192	0.06602	0.015480	0.008160	0.197
339	23.510	24.27	155.10	1747.0	0.10690	0.12830	0.230800	0.141000	0.179
286	11.940	20.76	77.87	441.0	0.08605	0.10110	0.065740	0.037910	0.158
354	11.140	14.07	71.24	384.6	0.07274	0.06064	0.045050	0.014710	0.169
421	14.690	13.98	98.22	656.1	0.10310	0.18360	0.145000	0.063000	0.208
124	13.370	16.39	86.10	553.5	0.07115	0.07325	0.080920	0.028000	0.142
543	13.210	28.06	84.88	538.4	0.08671	0.06877	0.029870	0.032750	0.162
537	11.690	24.44	76.37	406.4	0.12360	0.15520	0.045150	0.045310	0.213
567	20.600	29.33	140.10	1265.0	0.11780	0.27700	0.351400	0.152000	0.239
555	10.290	27.61	65.67	321.4	0.09030	0.07658	0.059990	0.027380	0.159

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
511	14.810	14.70	94.66	680.7	0.08472	0.05016	0.034160	0.025410	0.165
333	11.250	14.78	71.38	390.0	0.08306	0.04458	0.000974	0.002941	0.177
68	9.029	17.33	58.79	250.5	0.10660	0.14130	0.313000	0.043750	0.211
189	12.300	15.90	78.83	463.7	0.08080	0.07253	0.038440	0.016540	0.166
557	9.423	27.88	59.26	271.3	0.08123	0.04971	0.000000	0.000000	0.174
436	12.870	19.54	82.67	509.2	0.09136	0.07883	0.017970	0.020900	0.186
479	16.250	19.51	109.80	815.8	0.10260	0.18930	0.223600	0.091940	0.215
52	11.940	18.24	75.71	437.6	0.08261	0.04751	0.019720	0.013490	0.186
401	11.930	10.91	76.14	442.7	0.08872	0.05242	0.026060	0.017960	0.160
355	12.560	19.07	81.92	485.8	0.08760	0.10380	0.103000	0.043910	0.153
318	9.042	18.90	60.07	244.5	0.09968	0.19720	0.197500	0.049080	0.233
359	9.436	18.32	59.82	278.6	0.10090	0.05956	0.027100	0.014060	0.150
40	13.440	21.58	86.18	563.0	0.08162	0.06031	0.031100	0.020310	0.178
323	20.340	21.51	135.90	1264.0	0.11700	0.18750	0.256500	0.150400	0.256
495	14.870	20.21	96.12	680.9	0.09587	0.08345	0.068240	0.049510	0.148
45	18.650	17.60	123.70	1076.0	0.10990	0.16860	0.197400	0.100900	0.190
...	...	...	...	...	...	...	...	...	.
7	13.710	20.83	90.20	577.9	0.11890	0.16450	0.093660	0.059850	0.219
155	12.250	17.94	78.27	460.3	0.08654	0.06679	0.038850	0.023310	0.197
56	19.210	18.57	125.50	1152.0	0.10530	0.12670	0.132300	0.089940	0.191
151	8.219	20.70	53.27	203.9	0.09405	0.13050	0.132100	0.021680	0.222
203	13.810	23.75	91.56	597.8	0.13230	0.17680	0.155800	0.091760	0.225
34	16.130	17.88	107.00	807.2	0.10400	0.15590	0.135400	0.077520	0.199
417	15.500	21.08	102.90	803.1	0.11200	0.15710	0.152200	0.084810	0.208

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mea symmetr
42	19.070	24.81	128.30	1104.0	0.09081	0.21900	0.210700	0.099610	0.231
453	14.530	13.98	93.86	644.2	0.10990	0.09242	0.068950	0.064950	0.165
500	15.040	16.74	98.73	689.4	0.09883	0.13640	0.077210	0.061420	0.166
258	15.660	23.20	110.20	773.5	0.11090	0.31140	0.317600	0.137700	0.249
369	22.010	21.90	147.20	1482.0	0.10630	0.19540	0.244800	0.150100	0.182
313	11.540	10.72	73.73	409.1	0.08597	0.05969	0.013670	0.008907	0.183
426	10.480	14.98	67.49	333.6	0.09816	0.10130	0.063350	0.022180	0.192
140	9.738	11.97	61.24	288.5	0.09250	0.04102	0.000000	0.000000	0.190
388	11.270	15.50	73.38	392.0	0.08365	0.11140	0.100700	0.027570	0.181
116	8.950	15.76	58.74	245.2	0.09462	0.12430	0.092630	0.023080	0.130
198	19.180	22.49	127.50	1148.0	0.08523	0.14280	0.111400	0.067720	0.176
490	12.250	22.44	78.18	466.5	0.08192	0.05200	0.017140	0.012610	0.154
50	11.760	21.60	74.72	427.9	0.08637	0.04966	0.016570	0.011150	0.149
199	14.450	20.22	94.49	642.7	0.09872	0.12060	0.118000	0.059800	0.195
366	20.200	26.83	133.70	1234.0	0.09905	0.16690	0.164100	0.126500	0.187
455	13.380	30.72	86.34	557.2	0.09245	0.07426	0.028190	0.032640	0.137
162	19.590	18.15	130.70	1214.0	0.11200	0.16660	0.250800	0.128600	0.202
403	12.940	16.17	83.18	507.6	0.09879	0.08836	0.032960	0.023900	0.173
414	15.130	29.81	96.71	719.5	0.08320	0.04605	0.046860	0.027390	0.185
515	11.340	18.61	72.76	391.2	0.10490	0.08499	0.043020	0.025940	0.192
186	18.310	18.58	118.60	1041.0	0.08588	0.08468	0.081690	0.058140	0.162
3	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241400	0.105200	0.259
261	17.350	23.06	111.00	933.1	0.08662	0.06290	0.028910	0.028370	0.156

114 rows × 30 columns

CLASSIFIER SVM IS CODED UNDER BELOW

In [29]:
```
from sklearn.svm import SVC
```

In [30]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [31]:
```
svc_model=SVC()
```

In [32]:
```
svc_model.fit(X_train,y_train)
```

Out[32]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=
0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rb
f',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Evaluating the model code is under Below

In [33]:
```
y_predict=svc_model.predict(X_test)
```

In [34]:
```
y_predict
```

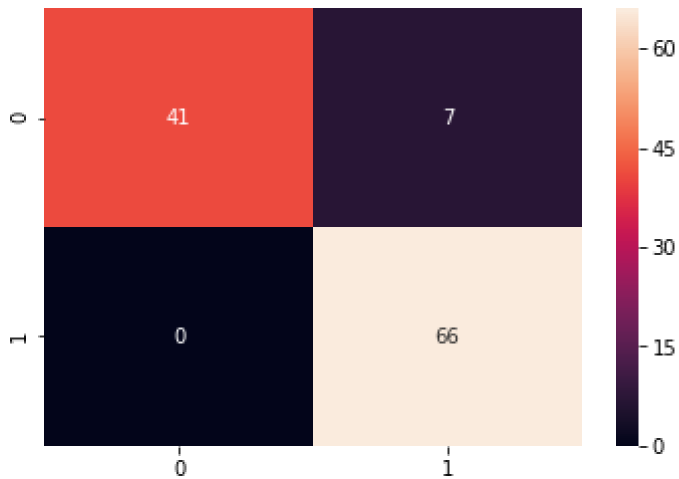
Out[34]:
```
array([0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1.,
1.,
       1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0., 1., 0., 0., 0., 1.,
0.,
       1., 1., 0., 1., 1., 0., 1., 1., 1., 0., 1., 1., 0., 0., 1., 0.,
1.,
       1., 1., 1., 1., 0., 1., 0., 1., 0., 0., 1., 1., 1., 1., 1., 1.,
```

```
                    1.,
                          1., 0., 1., 0., 1., 1., 1., 1., 1., 1., 0., 0., 0., 1., 0., 0.,
                    0.,
                          1., 0., 1., 0., 0., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1.,
                    0.,
                          1., 1., 0., 0., 1., 0., 1., 0., 1., 0., 1., 0.])
```

In [35]:
```
cm=confusion_matrix(y_test,y_predict)
```

In [36]:
```
sns.heatmap(cm,annot=True)
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e1fc40518>
```



In [37]:
```
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
```

Normalisation of breast cancer data

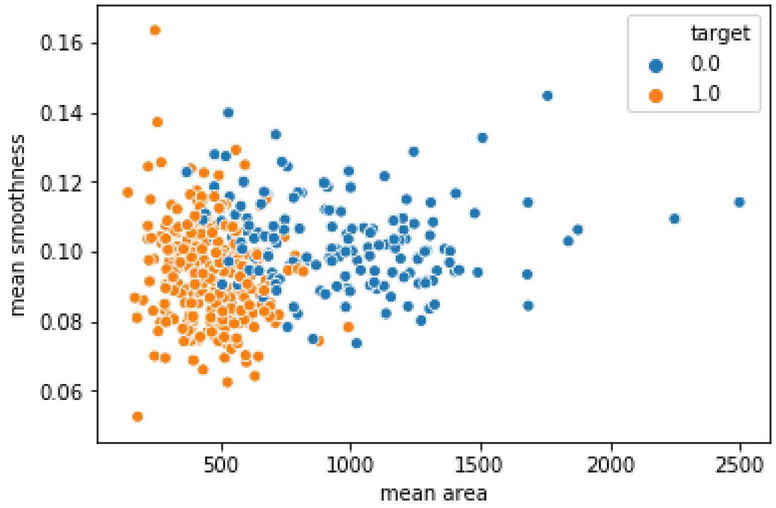
In [38]:
```
min_train=X_train.min()
```

In [39]:
```
range_train=(X_train-min_train).max()
```

In [40]:
```
X_train_scaled=(X_train-min_train)/range_train
```

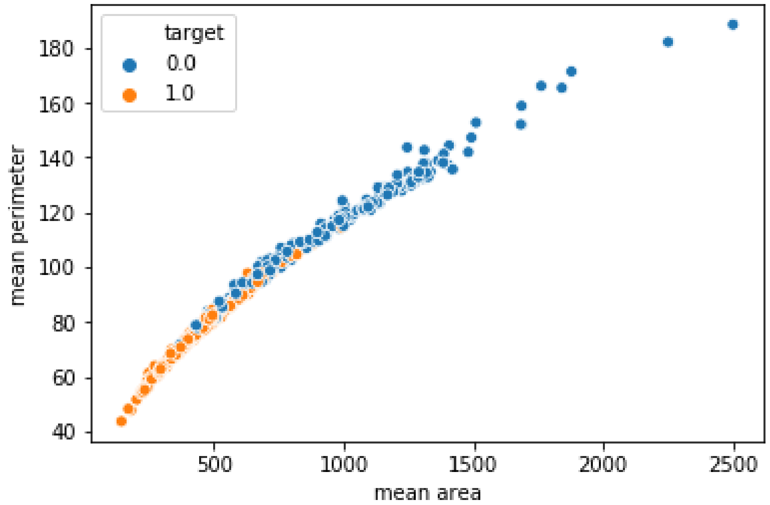
In [41]:
```
sns.scatterplot(x=X_train['mean area'],y=X_train['mean smoothness'],hue
=y_train)
```

Out[41]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e1fccdf60>
```



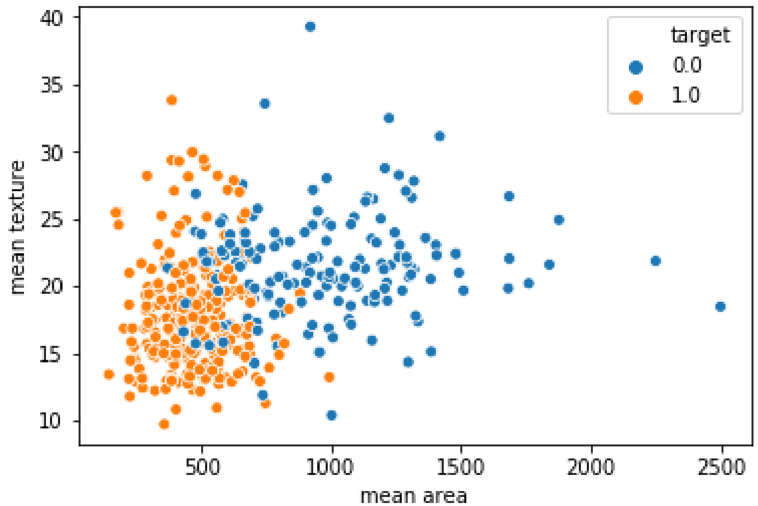
In [163]:
```
sns.scatterplot(x=X_train['mean area'],y=X_train['mean perimeter'],hue=
y_train)
```

Out[163]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e23fe41d0>
```



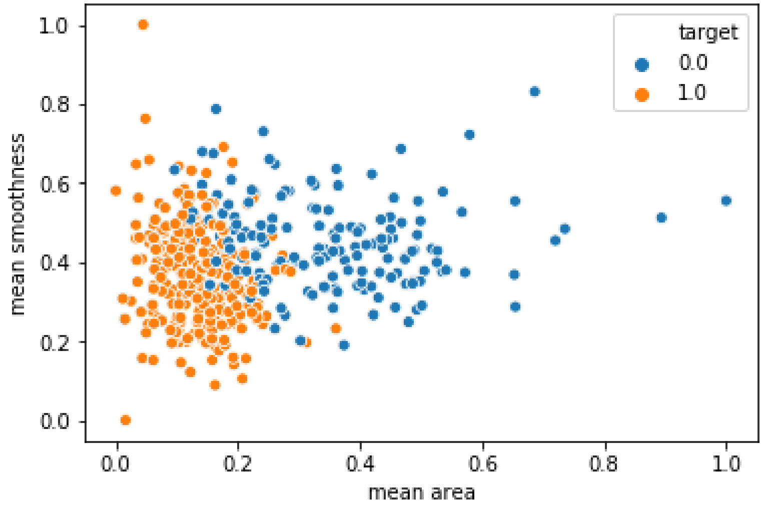
In [164]:
```
sns.scatterplot(x=X_train['mean area'],y=X_train['mean texture'],hue=y_
train)
```

Out[164]: <matplotlib.axes._subplots.AxesSubplot at 0x11e24055160>



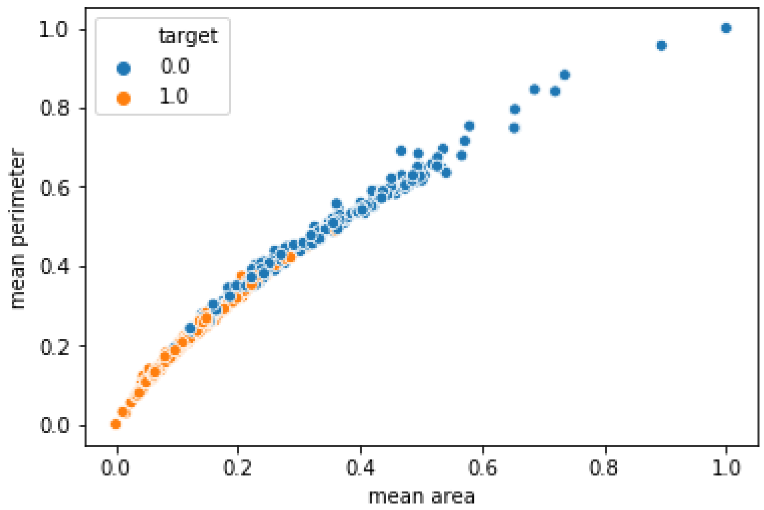
In [42]:
```
sns.scatterplot(x=X_train_scaled['mean area'],y=X_train_scaled['mean smoothness'],hue=y_train)
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x11e1ff76438>



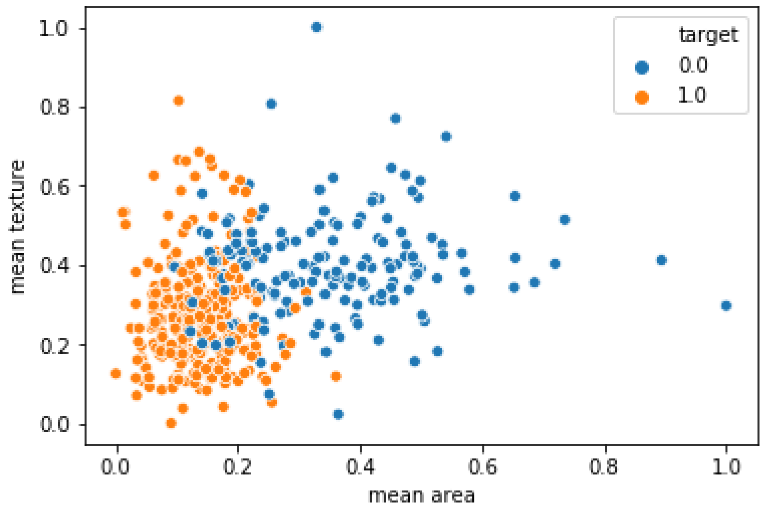
In [162]:
```
sns.scatterplot(x=X_train_scaled['mean area'],y=X_train_scaled['mean perimeter'],hue=y_train)
```

Out[162]: <matplotlib.axes._subplots.AxesSubplot at 0x11e23f7aba8>



In [165]:
```
sns.scatterplot(x=X_train_scaled['mean area'],y=X_train_scaled['mean te
xture'],hue=y_train)
```

Out[165]: <matplotlib.axes._subplots.AxesSubplot at 0x11e240b7710>



Normalized Training Data

In [168]:

```
X_train_scaled
```

Out[168]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sy
306	0.294335	0.206628	0.278350	0.167183	0.293220	0.101620	0.003423	0.016208	0
410	0.207251	0.265810	0.198328	0.108809	0.324546	0.103521	0.065206	0.104374	0
197	0.525297	0.410213	0.508673	0.373806	0.190304	0.205632	0.258435	0.287177	0
376	0.169861	0.355428	0.182157	0.082700	0.343956	0.449727	0.534208	0.295278	0
244	0.587770	0.466351	0.589524	0.429421	0.452018	0.418441	0.480084	0.441650	0
299	0.167022	0.452486	0.159353	0.080959	0.441184	0.149040	0.058458	0.093191	0
312	0.273510	0.123774	0.266049	0.153089	0.318769	0.184345	0.094939	0.126640	0
331	0.283923	0.326006	0.281459	0.157291	0.389636	0.285627	0.166518	0.146620	0
317	0.531923	0.309773	0.517656	0.375080	0.404712	0.283173	0.264761	0.395129	0
341	0.124237	0.241123	0.123350	0.058162	0.290512	0.223606	0.197329	0.113917	0
156	0.506366	0.373013	0.508673	0.348206	0.531462	0.451261	0.434630	0.523857	0
71	0.090255	0.166723	0.103656	0.042666	0.408053	0.410159	0.201640	0.142744	0
218	0.606702	0.400744	0.593670	0.461261	0.371942	0.341145	0.298032	0.431958	0
344	0.223816	0.194116	0.215880	0.117512	0.563059	0.163886	0.093861	0.161531	0
247	0.279663	0.148799	0.284431	0.156527	0.315699	0.353414	0.321931	0.197813	0
212	1.000000	0.296246	1.000000	1.000000	0.555836	0.405558	0.750000	0.792744	0
559	0.214350	0.480893	0.212356	0.110380	0.360928	0.253727	0.260544	0.204026	0
176	0.138341	0.282381	0.143805	0.067459	0.400469	0.337464	0.306232	0.184692	0
422	0.219083	0.213392	0.218851	0.112375	0.507087	0.298816	0.166284	0.223509	0
248	0.173648	0.524518	0.167369	0.086394	0.396678	0.162444	0.055740	0.080268	0

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sy
232	0.200625	0.815015	0.186580	0.103290	0.227228	0.050181	0.011638	0.031978	0
444	0.522931	0.241461	0.509364	0.359372	0.332581	0.318447	0.255389	0.310835	0
383	0.255999	0.262766	0.254647	0.135598	0.465559	0.338384	0.138051	0.143141	0
279	0.325098	0.184985	0.312349	0.188453	0.383949	0.176370	0.104944	0.184443	0
494	0.292442	0.366250	0.278281	0.167778	0.187054	0.102356	0.042174	0.062425	0
316	0.246060	0.147785	0.231221	0.134961	0.223075	0.039077	0.026312	0.025104	0
523	0.318472	0.303348	0.310552	0.181490	0.420060	0.268757	0.126172	0.188022	0
90	0.361541	0.483936	0.350909	0.220420	0.335019	0.204527	0.072680	0.146968	0
469	0.219556	0.286439	0.225209	0.112630	0.585628	0.395436	0.238988	0.276541	0
373	0.646457	0.258370	0.628913	0.505837	0.377629	0.270597	0.357779	0.444384	0
...	...	...	...	...	...	...	...	...	
539	0.033603	0.531958	0.031442	0.011420	0.307394	0.308325	0.216776	0.067793	0
110	0.132330	0.246195	0.129293	0.062280	0.461045	0.198331	0.101546	0.088370	0
5	0.258839	0.202570	0.267984	0.141626	0.678613	0.461996	0.369728	0.402038	0
144	0.178380	0.177883	0.169097	0.089917	0.228401	0.098184	0.052741	0.039140	0
103	0.137015	0.327697	0.139313	0.065719	0.432157	0.237992	0.144189	0.150547	0
210	0.643618	0.420358	0.628222	0.486733	0.345491	0.354027	0.384255	0.475199	0
446	0.509679	0.619547	0.507981	0.355806	0.427372	0.343599	0.397844	0.412177	0
41	0.187846	0.393642	0.194251	0.096625	0.632572	0.314153	0.244611	0.281759	0
362	0.273510	0.308759	0.263147	0.149904	0.398393	0.184467	0.062980	0.088519	0
377	0.306640	0.625634	0.290927	0.177712	0.203485	0.085516	0.029780	0.055517	0
254	0.590137	0.325330	0.571557	0.435364	0.459240	0.304951	0.323102	0.426988	0
146	0.228075	0.232330	0.243245	0.122479	0.509795	0.461996	0.388707	0.368539	0
86	0.354915	0.397362	0.348697	0.214264	0.377449	0.245660	0.282099	0.245427	0

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sy
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

normalizing the test data

In [43]:
```
min_test=X_test.min()
```

In [44]:
```
range_test=(X_test-min_test).max()
```

In [45]:
```
X_test_scaled=(X_test-min_test)/range_test
```

In [171]:
```
X_test_scaled
```

Out[171]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sym
28	0.368783	0.7275	0.367657	0.230073	0.553036	0.487059	0.462999	0.457449	0.5
163	0.214624	0.5750	0.198907	0.113447	0.461819	0.240181	0.147730	0.147517	0.2
123	0.327118	0.0085	0.306892	0.190153	0.577795	0.270588	0.243246	0.302039	0.4
361	0.264622	0.5425	0.239243	0.148970	0.261402	0.103457	0.091994	0.126712	0.4
549	0.135462	0.6745	0.116890	0.068652	0.210581	0.111747	0.042586	0.042656	0.5
339	0.796365	0.6775	0.762029	0.671760	0.536096	0.337195	0.634938	0.737062	0.4
286	0.193792	0.5020	0.184090	0.103217	0.264399	0.238733	0.180853	0.198170	0.2
354	0.152127	0.1675	0.134476	0.078664	0.090956	0.092271	0.123934	0.076895	0.3
421	0.337014	0.1630	0.336377	0.196857	0.486578	0.537376	0.398900	0.329326	0.6
124	0.268267	0.2835	0.245678	0.152192	0.070237	0.137919	0.222613	0.146367	0.1
543	0.259934	0.8670	0.236549	0.145618	0.273000	0.121701	0.082173	0.171197	0.3
537	0.180772	0.6860	0.172865	0.088155	0.753714	0.434570	0.124209	0.236853	0.6
567	0.644810	0.9305	0.649779	0.461930	0.678134	0.875475	0.966713	0.794564	0.8
555	0.107859	0.8445	0.092794	0.051151	0.319781	0.149973	0.165034	0.143126	0.2
511	0.343263	0.1990	0.309736	0.207566	0.247068	0.054335	0.093975	0.132828	0.3
333	0.157856	0.2030	0.135523	0.081015	0.225437	0.034136	0.002679	0.015374	0.4
68	0.042185	0.3305	0.041308	0.020286	0.532187	0.384253	0.861073	0.228698	0.6
189	0.212541	0.2590	0.191274	0.113099	0.195986	0.135312	0.105750	0.086461	0.3
557	0.062705	0.8580	0.044825	0.029341	0.201590	0.052706	0.000000	0.000000	0.3

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sym
436	0.242227	0.4410	0.220010	0.132907	0.333594	0.158118	0.049436	0.109252	0.4
479	0.418259	0.4395	0.423034	0.266379	0.480063	0.558009	0.615131	0.480606	0.6
52	0.193792	0.3760	0.167926	0.101737	0.219573	0.044742	0.054250	0.070518	0.4
401	0.193271	0.0095	0.171144	0.103957	0.299192	0.062516	0.071692	0.093884	0.2
355	0.226082	0.4175	0.214398	0.122720	0.284597	0.248507	0.283356	0.229535	0.2
318	0.042862	0.4090	0.050887	0.017674	0.442012	0.586606	0.543329	0.256560	0.8
359	0.063382	0.3800	0.049016	0.032519	0.457910	0.088362	0.074553	0.073497	0.2
40	0.271913	0.5430	0.246277	0.156328	0.206672	0.091077	0.085557	0.106168	0.4
323	0.631269	0.5395	0.618349	0.461495	0.667709	0.551493	0.705640	0.786200	0.9
495	0.346388	0.4745	0.320662	0.207653	0.392364	0.174842	0.187730	0.258808	0.2
45	0.543253	0.3440	0.527052	0.379653	0.575189	0.483077	0.543054	0.527444	0.5
...	...	...	...	...	...	...	...	...	
7	0.285975	0.5055	0.276360	0.162814	0.692468	0.468235	0.257662	0.312859	0.7
155	0.209937	0.3610	0.187084	0.111619	0.270784	0.114534	0.106878	0.121850	0.5
56	0.572418	0.3925	0.540522	0.412738	0.515246	0.331403	0.363961	0.470152	0.5
151	0.000000	0.4990	0.000000	0.000000	0.368647	0.345158	0.363411	0.113330	0.7
203	0.291183	0.6515	0.286537	0.171477	0.867084	0.512760	0.428611	0.479665	0.7
34	0.412010	0.3580	0.402080	0.262635	0.498306	0.437104	0.372490	0.405227	0.5
417	0.379199	0.5180	0.371399	0.260851	0.602554	0.441448	0.418707	0.443335	0.6
42	0.565127	0.7045	0.561476	0.391842	0.326427	0.665520	0.579642	0.520700	0.7
453	0.328681	0.1630	0.303749	0.191676	0.575189	0.207312	0.189684	0.339519	0.3
500	0.355242	0.3010	0.340193	0.211353	0.430936	0.366516	0.212407	0.321066	0.3
258	0.387532	0.6240	0.426027	0.247965	0.588220	1.000000	0.873728	0.719812	0.9
369	0.718244	0.5590	0.702911	0.556397	0.528277	0.580090	0.673453	0.784631	0.4

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	sym
313	0.172960	0.0000	0.153109	0.089330	0.263357	0.088833	0.037607	0.046560	0.4
426	0.117754	0.2130	0.106413	0.056462	0.422205	0.239457	0.174278	0.115944	0.5
140	0.079110	0.0625	0.059642	0.036829	0.348449	0.021249	0.000000	0.000000	0.5
388	0.158898	0.2390	0.150490	0.081886	0.233125	0.276018	0.277029	0.144119	0.4
116	0.038071	0.2520	0.040934	0.017979	0.376075	0.322715	0.254828	0.120648	0.0
198	0.570856	0.5885	0.555489	0.410996	0.253714	0.389683	0.306465	0.353999	0.4
490	0.209937	0.5860	0.186410	0.114318	0.210581	0.060995	0.047153	0.065917	0.2
50	0.184417	0.5440	0.160518	0.097514	0.268569	0.052525	0.045585	0.058285	0.2
199	0.324514	0.4750	0.308464	0.191023	0.429502	0.309321	0.324622	0.312598	0.5
366	0.623978	0.8055	0.601886	0.448435	0.433802	0.476923	0.451444	0.661265	0.4
455	0.268788	1.0000	0.247474	0.153803	0.347798	0.141575	0.077552	0.170622	0.1
162	0.592209	0.3715	0.579436	0.439728	0.602554	0.475837	0.689959	0.672243	0.5
403	0.245873	0.2725	0.223827	0.132210	0.430414	0.192615	0.090674	0.124935	0.3
414	0.359929	0.9545	0.325077	0.224457	0.227261	0.039457	0.128913	0.143178	0.4
515	0.162544	0.3945	0.145850	0.081538	0.510034	0.180416	0.118349	0.135599	0.5
186	0.525546	0.3930	0.488887	0.364416	0.262184	0.179294	0.224732	0.303921	0.2
3	0.166710	0.4830	0.181920	0.079317	1.000000	0.900452	0.664099	0.549922	1.0
261	0.475548	0.6170	0.432014	0.317444	0.271827	0.100452	0.079532	0.148301	0.2

114 rows × 30 columns

Retraining the svm classifier

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

```
Out[46]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=
0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rb
f',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
In [47]: y_predict=svc_model.predict(X_test_scaled)
```

```
In [48]: cm=confusion_matrix(y_test,y_predict)
```

Svm hyperparameters optimizations

```
In [51]: from sklearn.model_selection import GridSearchCV
```

```
In [52]: param_grid = { 'C': [0.1,1,10,100], 'gamma': [1,0.1,0.01,0.001],'kerne
l' :['rbf'] }
```

```
In [53]: from sklearn.model_selection import GridSearchCV
```

```
In [54]: grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=4)
```

```
In [55]: grid.fit(X_train_scaled,y_train)
```

```
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 ...................................
[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.648, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ...................................
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent
workers.
[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=0.637, 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:    0.5s finished
```

Out[55]:

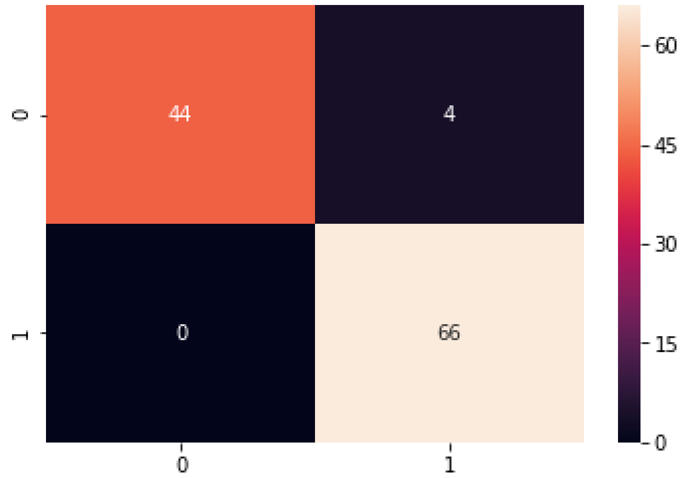
```
GridSearchCV(cv=None, error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='scale', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrink
ing=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.0
1, 0.001],
                         'kernel': ['rbf']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fa
lse,
             scoring=None, verbose=4)
```

In [56]:
```
y_predict=grid.predict(X_test_scaled)
```

In [57]:
```
cm=confusion_matrix(y_test,y_predict)
```

In [58]:
```
sns.heatmap(cm,annot=True)
```

Out[58]: <matplotlib.axes._subplots.AxesSubplot at 0x11e1fe9ec50>

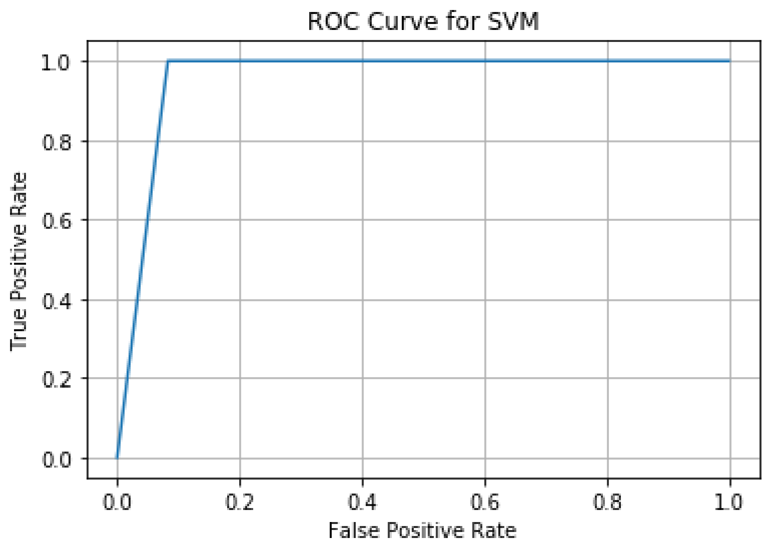


In [59]:
```
print(classification_report(y_test,y_predict))
```

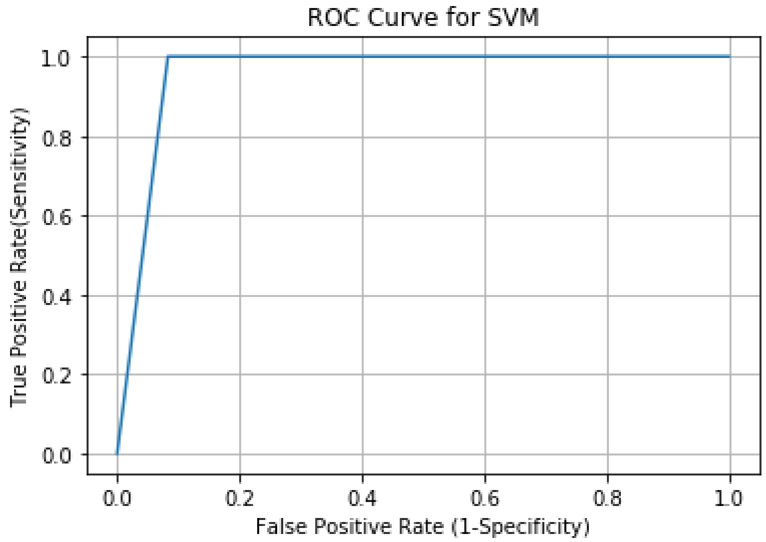
```
              precision    recall  f1-score   support

         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 [167]:
```
grid.score(X_test_scaled,y_test)
```
Out[167]: 0.9649122807017544

In [60]:
```
from sklearn import metrics
fpr, tpr, thresholds=metrics.roc_curve(y_test,y_predict)
plt.plot(fpr,tpr)
plt.title("ROC Curve for SVM")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.grid(True)
```



In [61]:
```
from sklearn import metrics
fpr, tpr, thresholds=metrics.roc_curve(y_test,y_predict)
plt.plot(fpr,tpr)
plt.title("ROC Curve for SVM")
plt.xlabel("False Positive Rate (1-Specificity)")
plt.ylabel("True Positive Rate(Sensitivity)")
plt.grid(True)
```



Knn classifier code

In [62]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [63]:
```
knn=KNeighborsClassifier(n_neighbors=1)
```

In [64]:
```
knn.fit(X_train_scaled,y_train)
```

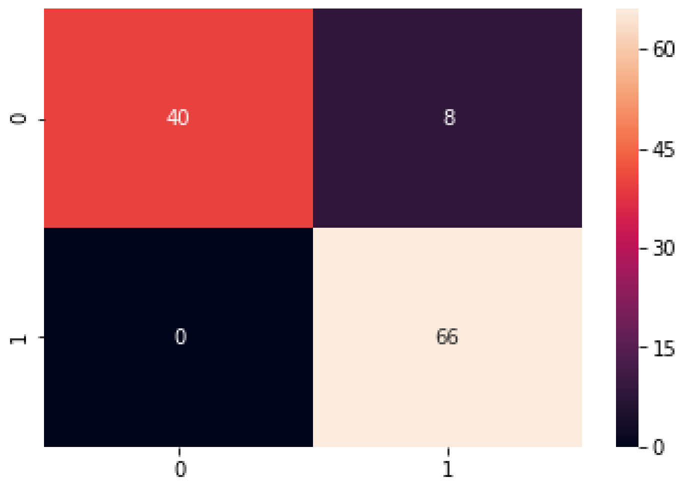
Out[64]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=
2,
                     weights='uniform')
```

In [65]:
```
y_predict1=knn.predict(X_test_scaled)
```

In [66]:
```
cm1=confusion_matrix(y_test,y_predict1)
```

In [67]:
```python
sns.heatmap(cm1,annot=True)
```

Out[67]: <matplotlib.axes._subplots.AxesSubplot at 0x11e21bdd4e0>



In [68]:
```python
print(classification_report(y_test,y_predict1))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.83      0.91        48
         1.0       0.89      1.00      0.94        66

    accuracy                           0.93       114
   macro avg       0.95      0.92      0.93       114
weighted avg       0.94      0.93      0.93       114
```

In [69]:
```python
error_rate = []

for i in range(1,40):

    knn=KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train_scaled,y_train)
    pred_i=knn.predict(X_test_scaled)
```

```
    error_rate.append(np.mean(pred_i !=y_test))
```

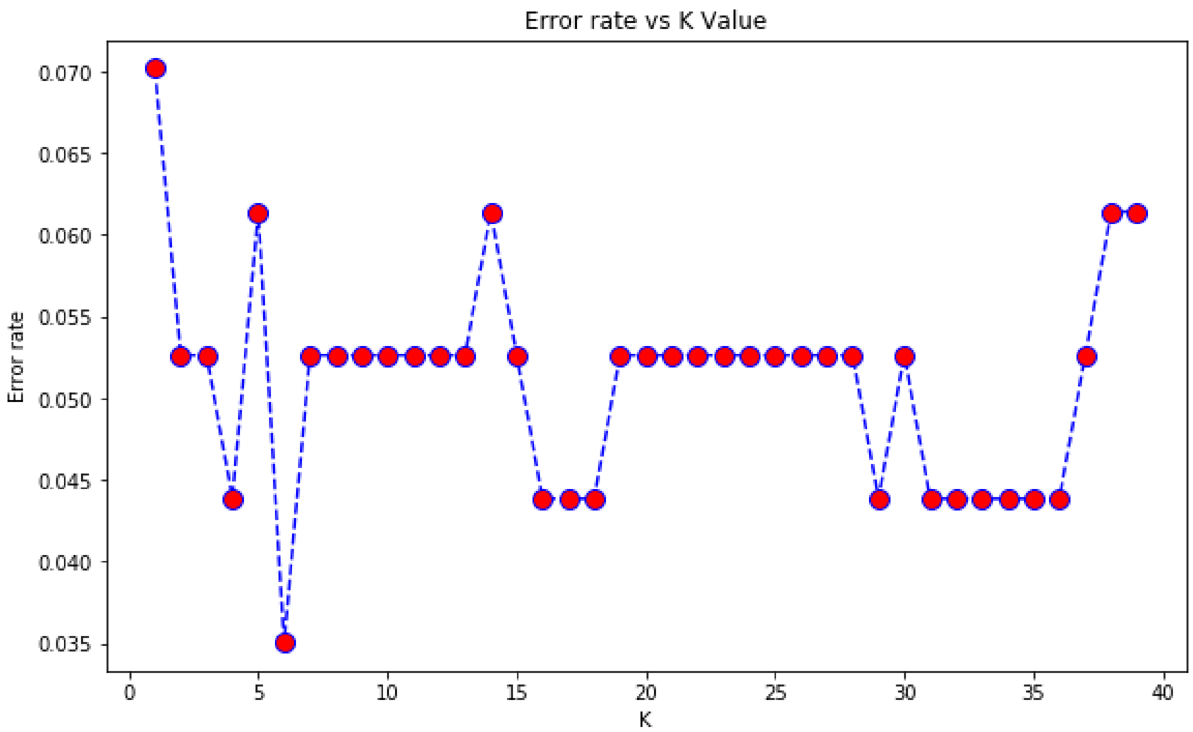
In [70]:
```
error_rate
```

Out[70]:
```
[0.07017543859649122,
 0.05263157894736842,
 0.05263157894736842,
 0.043859649122807015,
 0.06140350877192982,
 0.03508771929824561,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.06140350877192982,
 0.05263157894736842,
 0.043859649122807015,
 0.043859649122807015,
 0.043859649122807015,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.05263157894736842,
 0.043859649122807015,
 0.05263157894736842,
 0.043859649122807015,
 0.043859649122807015,
 0.043859649122807015,
 0.043859649122807015,
```

```
0.043859649122807015,
0.043859649122807015,
0.05263157894736842,
0.06140350877192982,
0.06140350877192982]
```

In [71]:
```
plt.figure(figsize=(10,6))
plt.plot(range(1,40),error_rate,color='blue',linestyle='dashed',marker=
'o',markerfacecolor='red' ,markersize=10 )
plt.title('Error rate vs K Value')
plt.xlabel('K')
plt.ylabel('Error rate')
```

Out[71]:
```
Text(0, 0.5, 'Error rate')
```



K Parameter is evaluated For the classfier

In [72]:
```
knn=KNeighborsClassifier(n_neighbors=6)
```

In [159]:
```
knn1=KNeighborsClassifier(n_neighbors=5)
```

In [160]:
```
knn1.fit(X_train_scaled,y_train)
```

Out[160]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=
2,
                     weights='uniform')
```

In [76]:
```
knn.fit(X_train_scaled,y_train)
```

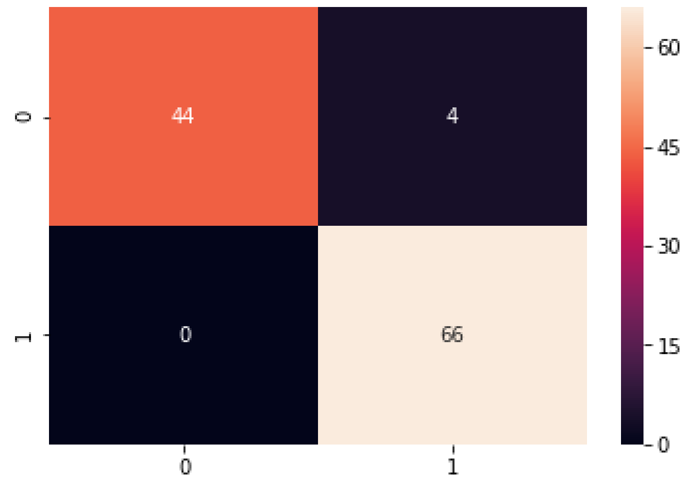
Out[76]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=
2,
                     weights='uniform')
```

In [77]:
```
y_predict2=knn.predict(X_test_scaled)
```

In [78]:
```
cm=confusion_matrix(y_test,y_predict2)
```

In [79]:
```
sns.heatmap(cm,annot=True)
```

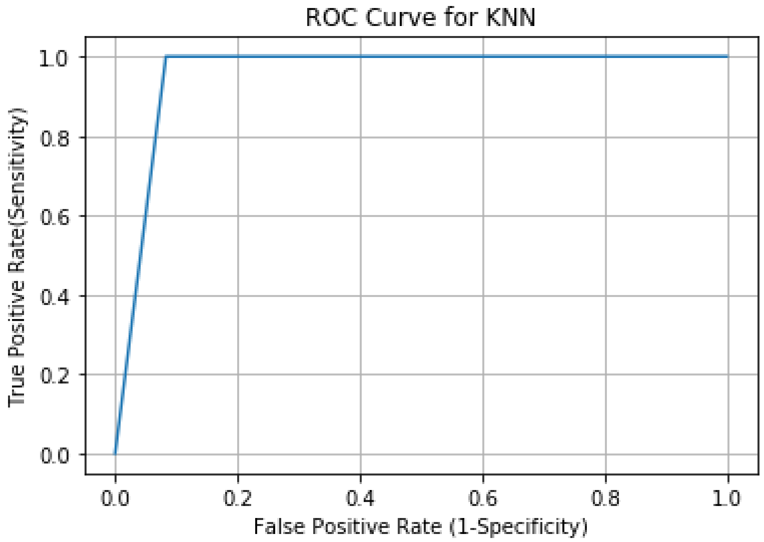
Out[79]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e21d2c4e0>
```



In [80]:
```
print(classification_report(y_test,y_predict2))
```

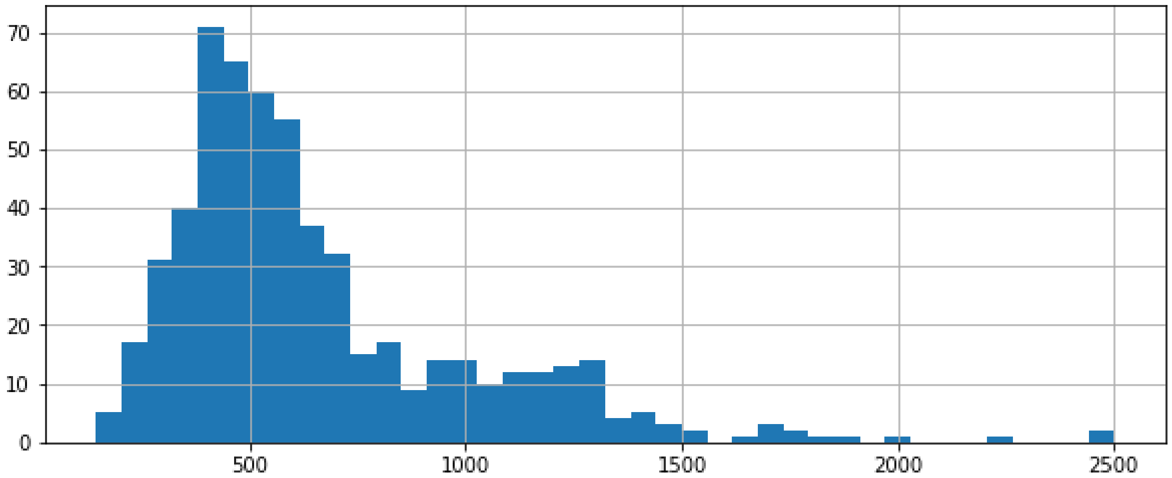
```
              precision    recall  f1-score   support

         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 [81]:
```
from sklearn import metrics
fpr, tpr, thresholds=metrics.roc_curve(y_test,y_predict2)
plt.plot(fpr,tpr)
plt.title("ROC Curve for KNN")
plt.xlabel("False Positive Rate (1-Specificity)")
plt.ylabel("True Positive Rate(Sensitivity)")
plt.grid(True)
```



Logistic regression classifier

In [82]:
```
X['mean area'].hist(bins=40,figsize=(10,4))
```
Out[82]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e21e47d30>
```



In [83]:
```
from sklearn.linear_model import LogisticRegression
```

In [84]:
```
logmodel=LogisticRegression()
```

In [85]:
```
logmodel.fit(X_train_scaled,y_train)
```

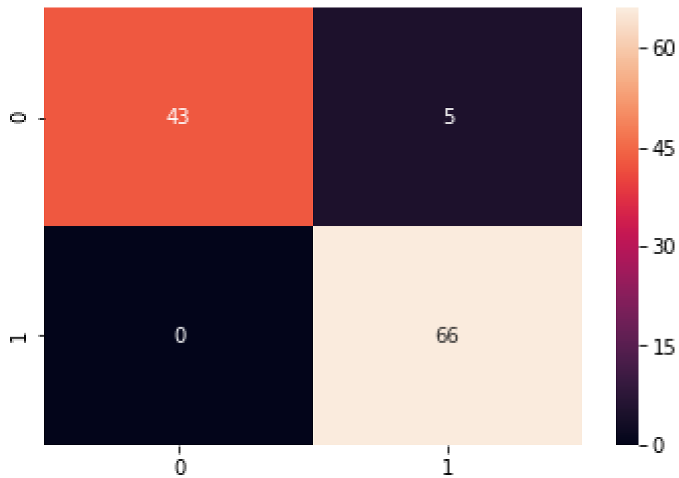
Out[85]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=
True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbo
se=0,
                   warm_start=False)
```

In [86]:
```
y_predict3=logmodel.predict(X_test_scaled)
```

In [87]:
```
cm3=confusion_matrix(y_test,y_predict3)
```

In [88]:
```
sns.heatmap(cm3,annot=True)
```

Out[88]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e21c449e8>
```



In [89]:
```
print(classification_report(y_test,y_predict3))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.90      0.95        48
         1.0       0.93      1.00      0.96        66

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

Naive Bayes classifier

In [90]:
```
from sklearn.preprocessing import StandardScaler
```

In [91]:
```
sc = StandardScaler()
X_train1 = sc.fit_transform(X_train_scaled)
X_test1 = sc.transform(X_test_scaled)
```

In [92]:
```
from sklearn.naive_bayes import GaussianNB
```

In [93]:
```
nb=GaussianNB()
```

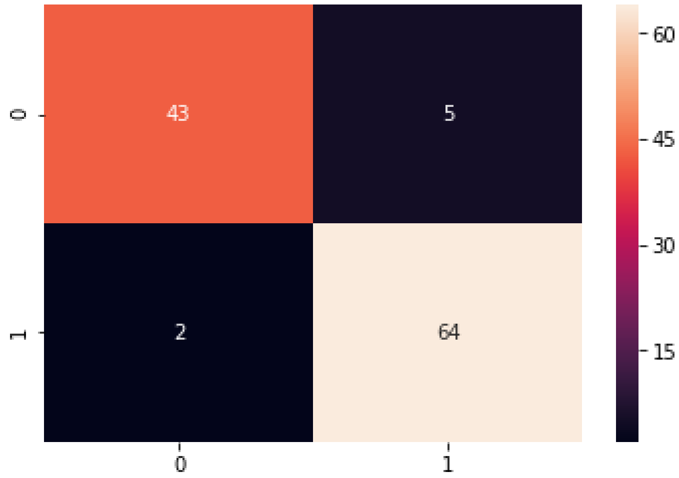
In [94]:
```
nb.fit(X_train_scaled,y_train)
```

Out[94]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

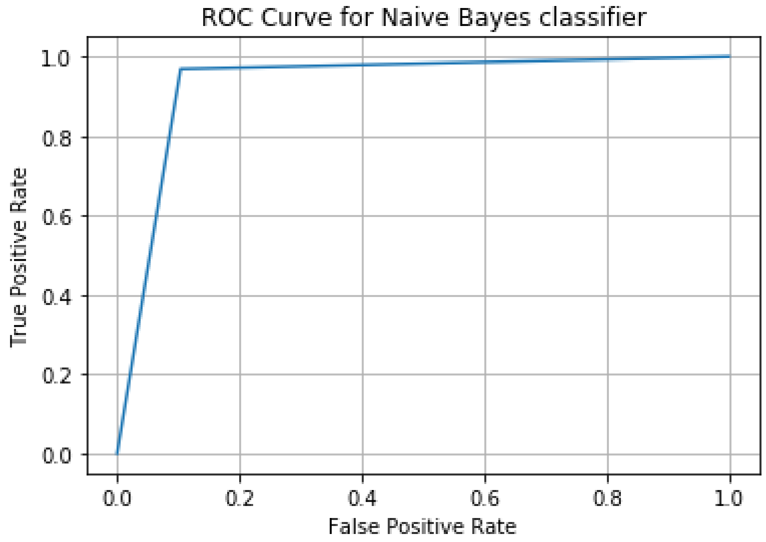
In [95]:
```
y_predict4=nb.predict(X_test_scaled)
```

In [96]:
```
cm=confusion_matrix(y_test,y_predict4)
sns.heatmap(cm,annot=True)
```

Out[96]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e21f7b2e8>
```



In [97]:
```
from sklearn import metrics
fpr, tpr, thresholds=metrics.roc_curve(y_test,y_predict4)
plt.plot(fpr,tpr)
plt.title("ROC Curve for Naive Bayes classifier")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.grid(True)
```



In [98]:
```
print(classification_report(y_test,y_predict4))
```

```
              precision    recall  f1-score   support

         0.0       0.96      0.90      0.92        48
         1.0       0.93      0.97      0.95        66

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

In [99]:
```
r1=classification_report(y_test,y_predict4)
```

In [100]:
```
s1=X_train_scaled.loc[44]
```

In [101]:
```
from sklearn.preprocessing import StandardScaler
```

In [102]:
```
sc=StandardScaler()
```

In [103]:
```
data_np = np.asarray(s1, dtype = float)
```

In [104]:
```
data_np = data_np.reshape(1,-1)
```

In [105]:
```
data_np
```

Out[105]:
```
array([[0.29291495, 0.40919851, 0.28767881, 0.16472087, 0.4018236 ,
        0.26170174, 0.19350984, 0.2610338 , 0.34646465, 0.25791658,
        0.02980264, 0.05531849, 0.02718749, 0.01481887, 0.05564809,
        0.08703097, 0.03666667, 0.12981625, 0.04570271, 0.02664608,
        0.32949583, 0.47627932, 0.3233171 , 0.17109154, 0.61137294,
        0.35229114, 0.33737557, 0.55223368, 0.50558327, 0.26984127]])
```

In [106]:
```
r1
```

Out[106]:
```
'              precision    recall  f1-score   support\n\n         0.0
      0.96      0.90      0.92        48\n         1.0       0.93
0.97      0.95        66\n\n    accuracy                           0.94
       114\n   macro avg       0.94      0.93      0.94       114\nweig
hted avg       0.94      0.94      0.94       114\n'
```

Decision tree classfier

In [107]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [108]:
```
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state
 = 0)
```

In [109]:
```
classifier.fit(X_train_scaled,y_train)
```

Out[109]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='ent
ropy',
                       max_depth=None, max_features=None, max_leaf_node
s=None,
                       min_impurity_decrease=0.0, min_impurity_split=No
```

```
ne,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecate
d',
                                random_state=0, splitter='best')
```

In [110]:
```
y_predict5=classifier.predict(X_test)
```

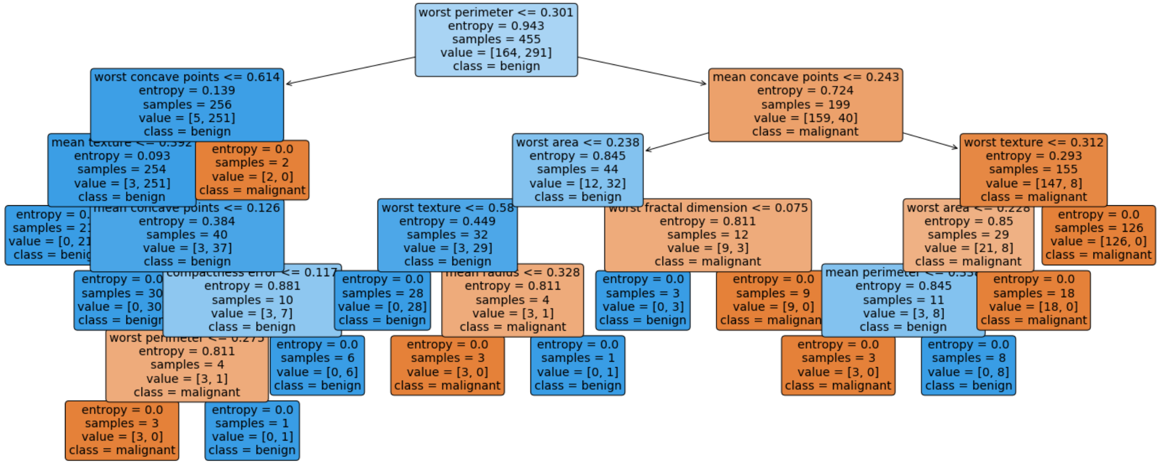
In [172]:
```
classifier.get_depth()
```

Out[172]:
```
6
```

In [173]:
```
classifier.decision_path(X_train_scaled)
```

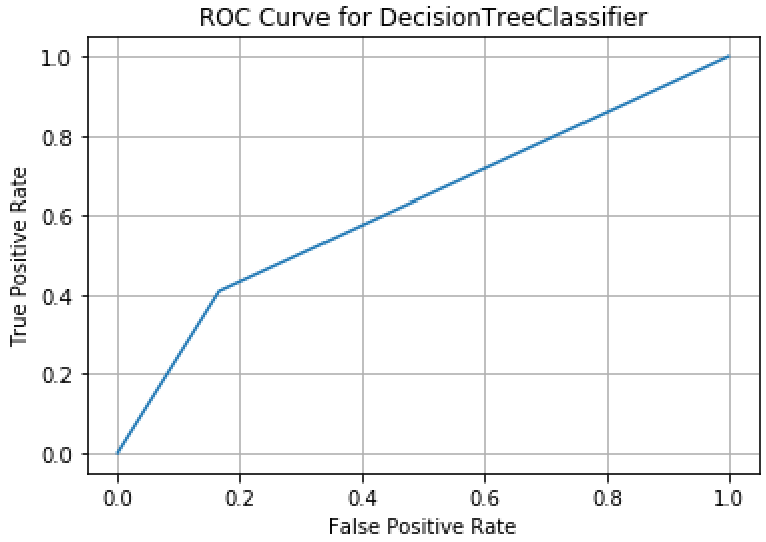
Out[173]:
```
<455x29 sparse matrix of type '<class 'numpy.int64'>'
	with 1960 stored elements in Compressed Sparse Row format>
```

In [183]:
```
from sklearn.tree import plot_tree
plt.figure(figsize=(25,10))
a = plot_tree(classifier,
              feature_names=cancer.feature_names,
              class_names=cancer.target_names,
              filled=True,
              rounded=True,
              fontsize=14)
```



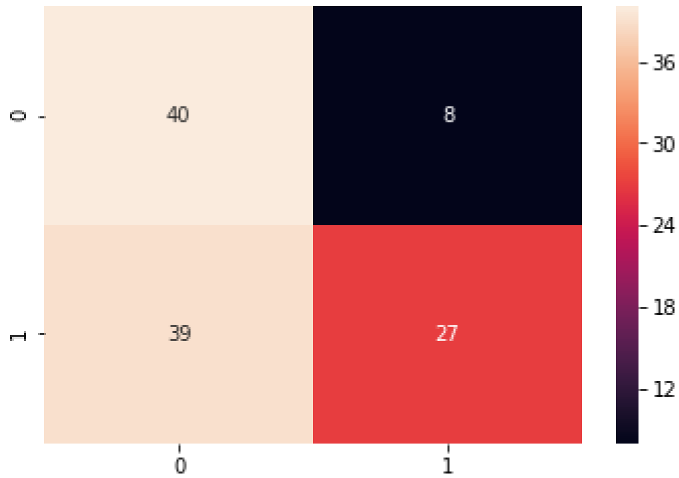
In [111]:
```
cm=confusion_matrix(y_test,y_predict5)
```

In [112]:
```
from sklearn import metrics
fpr, tpr, thresholds=metrics.roc_curve(y_test,y_predict5)
plt.plot(fpr,tpr)
plt.title("ROC Curve for DecisionTreeClassifier")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.grid(True)
```



In [113]:
```
sns.heatmap(cm,annot=True)
```

Out[113]: <matplotlib.axes._subplots.AxesSubplot at 0x11e2210ab70>



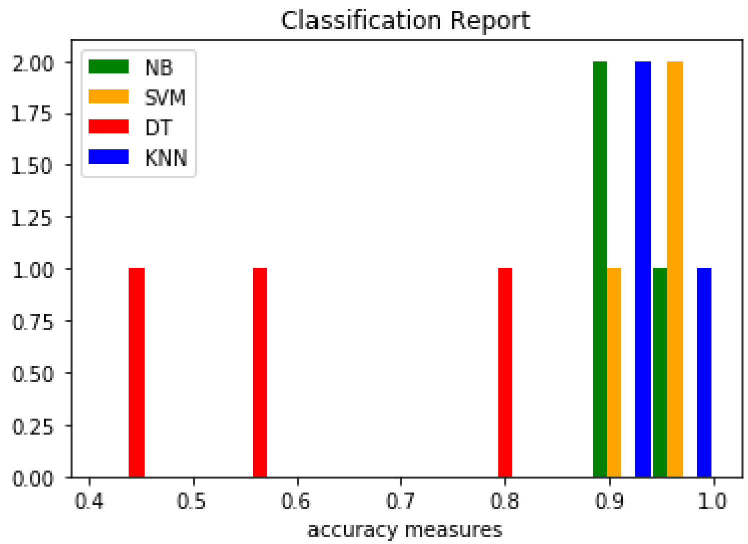
In [114]:
```
print(classification_report(y_test,y_predict5))
```

```
              precision    recall  f1-score   support

         0.0       0.51      0.83      0.63        48
         1.0       0.77      0.41      0.53        66

    accuracy                           0.59       114
   macro avg       0.64      0.62      0.58       114
weighted avg       0.66      0.59      0.57       114
```

In [115]:

```
svm=[0.940, 1.00,0.970]
nb=[0.960,0.900,0.920]
dt=[0.770,0.410,0.530]
knn=[0.890,1.00,0.940]
plt.xlabel("accuracy measures")
plt.title("Classification Report")
plt.hist([nb,svm,dt,knn], rwidth=0.95, color=['green','orange','red','b
lue'], label=['NB','SVM','DT','KNN']
        )
plt.legend()
```

Out[115]: <matplotlib.legend.Legend at 0x11e221cdc88>

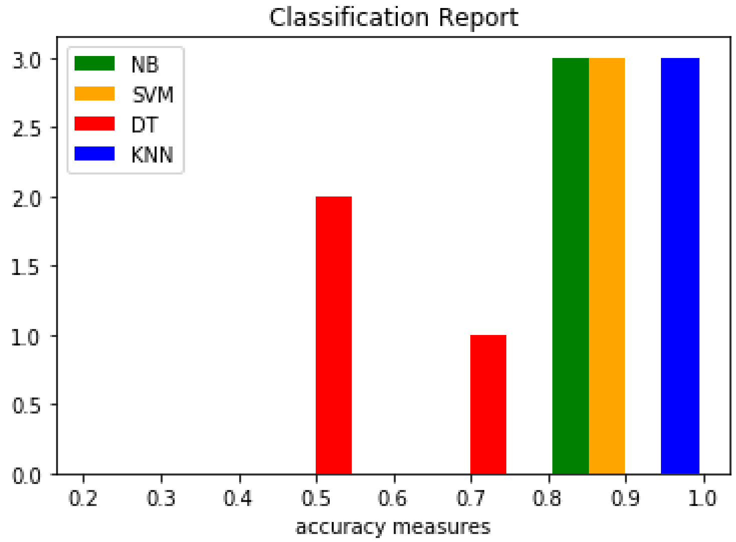


In [116]:

```
svm=[0.940, 1.00,0.970]
nb=[0.960,0.900,0.920]
dt=[0.770,0.410,0.530]
knn=[0.890,1.00,0.940]
s1=[0.940,0.960,0.770,0.890]



plt.xlabel("accuracy measures")
plt.title("Classification Report")
plt.hist([nb,svm,dt,knn],bins=[0.2,0.4,0.6,0.8,1.0], rwidth=0.95, color
=['green','orange','red','blue'], label=['NB','SVM','DT','KNN']
        )
plt.legend()
```

Out[116]:

```
<matplotlib.legend.Legend at 0x11e2229e4e0>
```

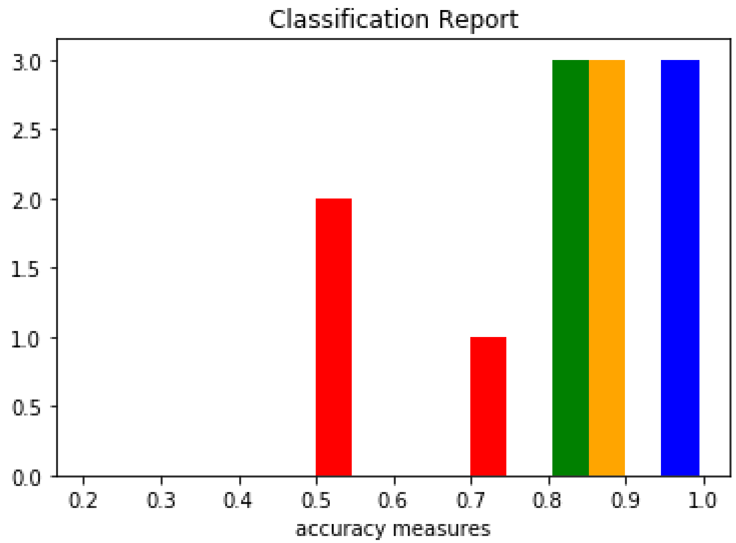


In [117]:

```
svm=[0.940, 1.00,0.970]
nb=[0.960,0.900,0.920]
dt=[0.770,0.410,0.530]
knn=[0.890,1.00,0.940]

pets= "NB", "SVM", "KNN","DT"



plt.xlabel("accuracy measures")
plt.title("Classification Report")
plt.hist([nb,svm,dt,knn],bins=[0.2,0.4,0.6,0.8,1.0], rwidth=0.95, color
=['green','orange','red','blue'], label=['NB','SVM','DT','KNN']
        )
# plt.legend(b,pets,fontsize=20)
```

Out[117]:

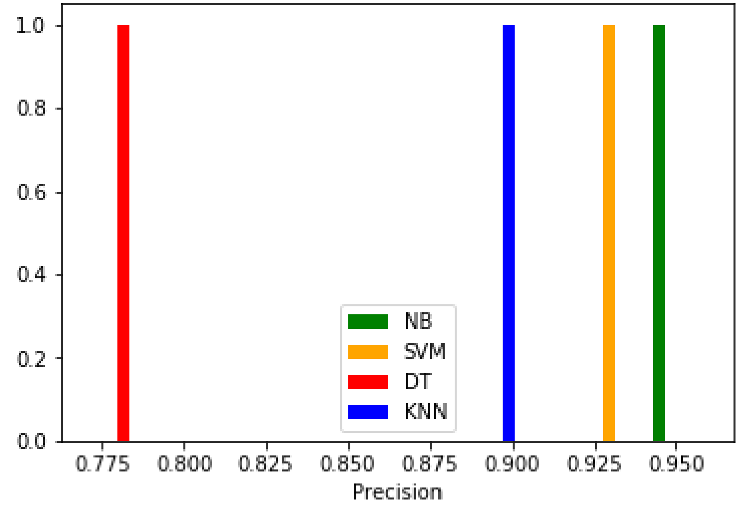
```
([array([0., 0., 0., 3.]),
  array([0., 0., 0., 3.]),
  array([0., 2., 1., 0.]),
  array([0., 0., 0., 3.])],
 array([0.2, 0.4, 0.6, 0.8, 1. ]),
 <a list of 4 Lists of Patches objects>)
```



In [118]:

```
s1=[0.960]
s2=[0.940]
s3=[0.770]
s4=[0.890]
plt.xlabel('Precision')
plt.hist([s1,s2,s3,s4],color=['green','orange','red','blue'], label=['N
B','SVM','DT','KNN'])
plt.legend()
```

Out[118]: <matplotlib.legend.Legend at 0x11e221967f0>

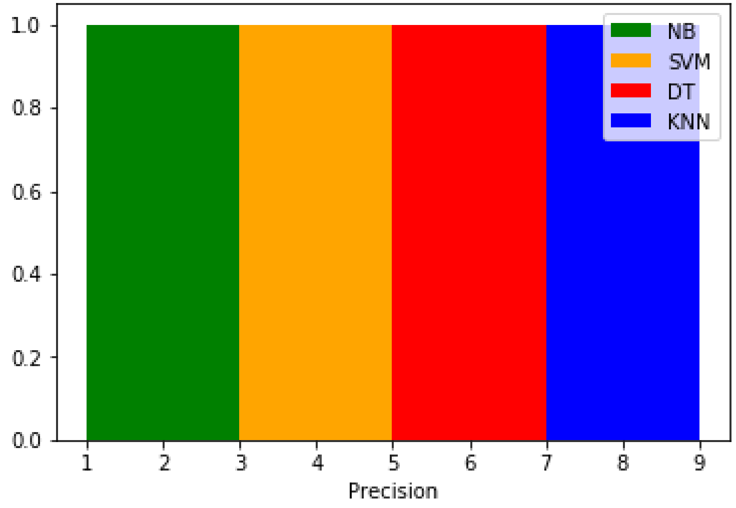


s1=[0.960] s2=[0.940] s3=[0.770] s4=[0.890] plt.xlabel('Precision') plt.hist([s1,s2,s3,s4],color=['green','orange','red','blue'], bins=[0.001,1] label=['NB','SVM','DT','KNN']) plt.legend()

In [119]:

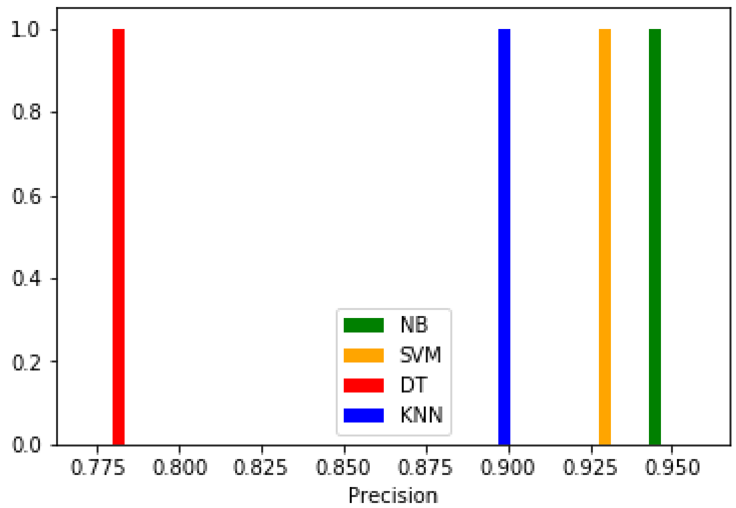
```
s1=[0.960]
s2=[0.940]
s3=[0.770]
s4=[0.890]
plt.xlabel('Precision')
plt.hist([s1,s2,s3,s4],color=['green','orange','red','blue'], bins=[0.0
,10],label=['NB','SVM','DT','KNN'])
plt.legend()
```

Out[119]: <matplotlib.legend.Legend at 0x11e223bab00>



In [120]:
```
s1=[0.960]
s2=[0.940]
s3=[0.770]
s4=[0.890]
plt.xlabel('Precision')
plt.hist([s1,s2,s3,s4],color=['green','orange','red','blue'],label=['N
B','SVM','DT','KNN'])
plt.legend()
```

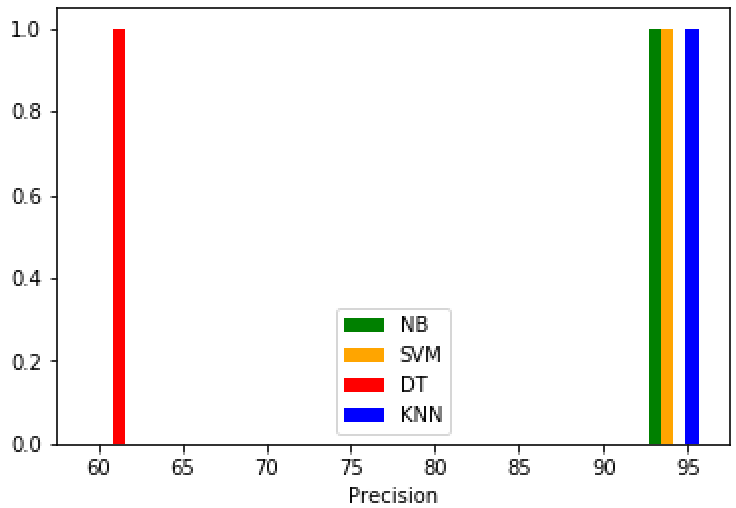
Out[120]:
```
<matplotlib.legend.Legend at 0x11e22124470>
```



In [121]:

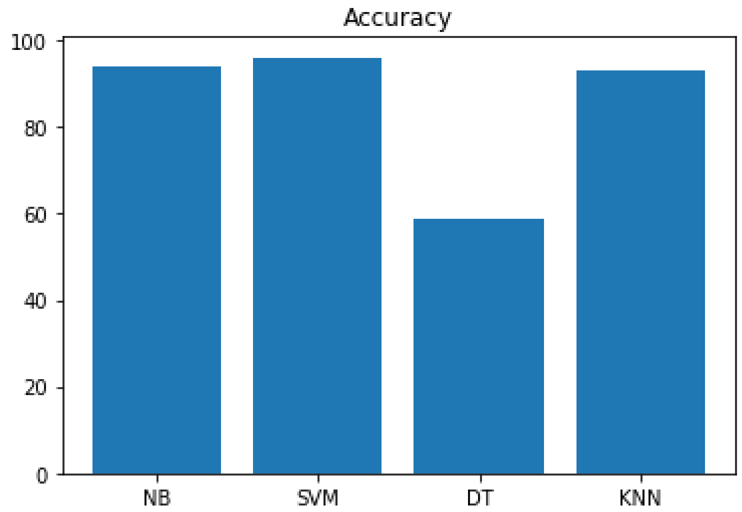
```
s1=[94.00]
s2=[96.00]
s3=[59.00]
s4=[93.00]


plt.xlabel('Precision')
plt.hist([s1,s2,s3,s4],color=['green','orange','red','blue'],label=['N
B','SVM','DT','KNN'])
plt.legend()
```

Out[121]: <matplotlib.legend.Legend at 0x11e1ff4d048>



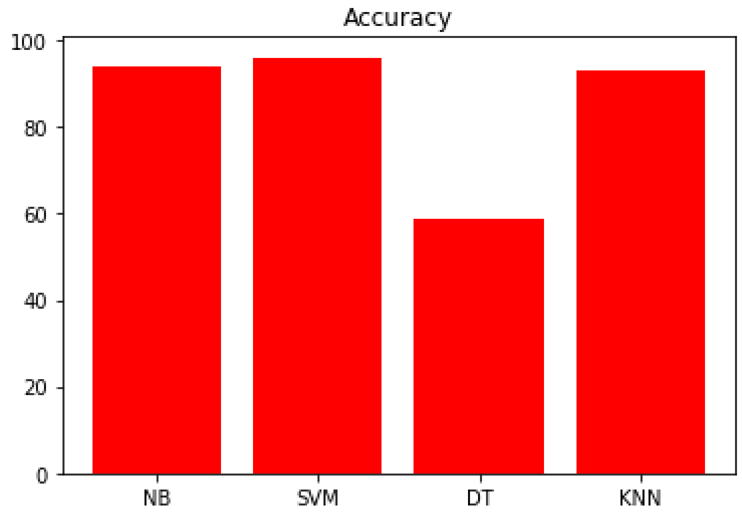
In [122]:
```
cf1=['NB','SVM','DT','KNN']
acc=[94,96,59,93]
ypos=np.arange(len(cf1))
plt.xticks(ypos,cf1)
plt.bar(ypos,acc)
plt.title('Accuracy')
```

Out[122]:
```
Text(0.5, 1.0, 'Accuracy')
```



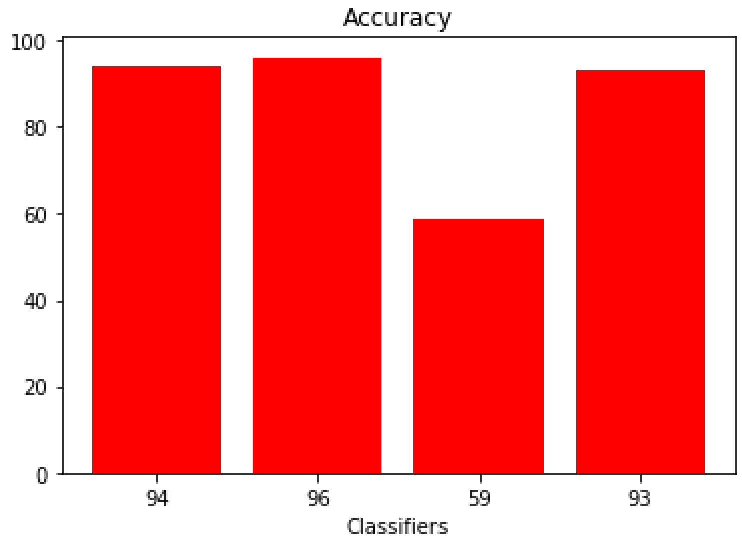
In [123]:
```
cf1=['NB','SVM','DT','KNN']
acc=[94,96,59,93]
ypos=np.arange(len(cf1))
plt.xticks(ypos,cf1)

plt.bar(ypos,acc,color='red')
plt.title('Accuracy')
```

Out[123]: Text(0.5, 1.0, 'Accuracy')



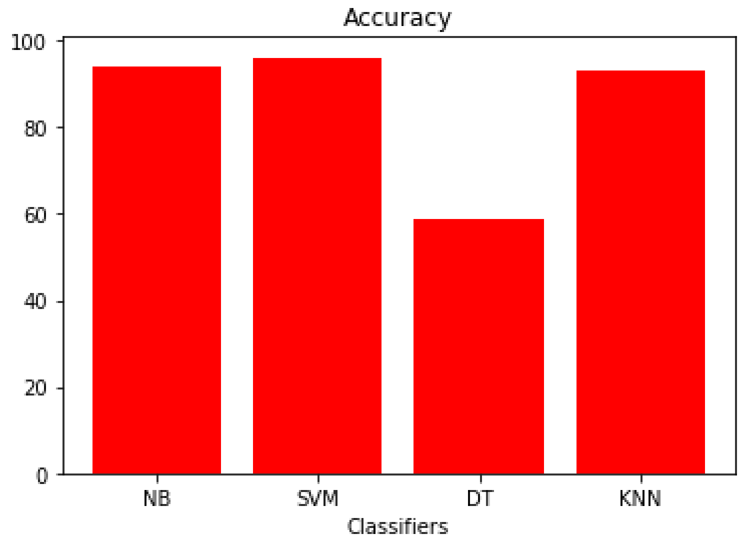
In [124]:
```
cf1=['NB','SVM','DT','KNN']
acc=[94,96,59,93]
ypos=np.arange(len(cf1))
plt.xticks(ypos,cf1)
plt.xlabel('Classifiers')
plt.bar(ypos,acc,color='red',tick_label=acc)
plt.title('Accuracy')
```

Out[124]:
```
Text(0.5, 1.0, 'Accuracy')
```



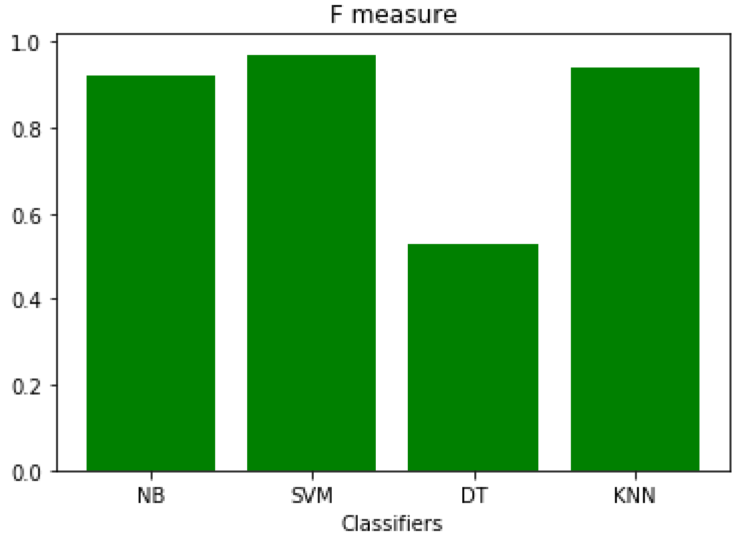
In [125]:
```
cf1=['NB','SVM','DT','KNN']
acc=[94,96,59,93]
ypos=np.arange(len(cf1))
plt.xticks(ypos,cf1)
plt.xlabel('Classifiers')
plt.bar(ypos,acc,color='red')
plt.title('Accuracy')
```

Out[125]: Text(0.5, 1.0, 'Accuracy')



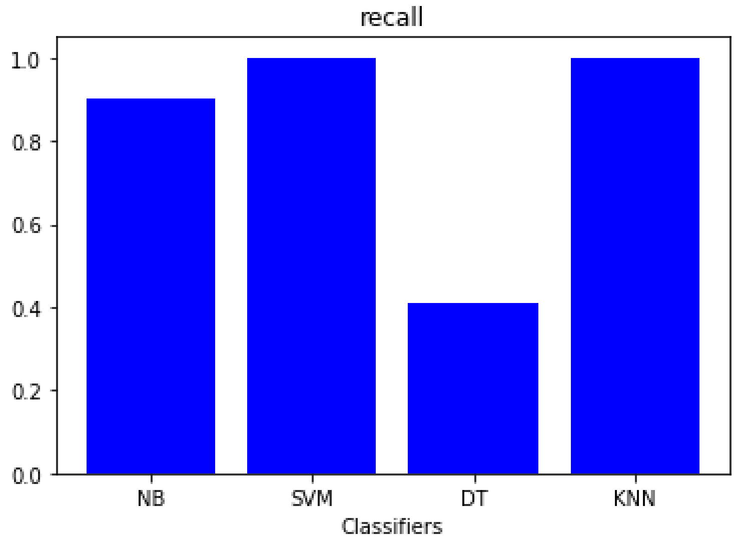
In [126]:
```
cf2=['NB','SVM','DT','KNN']
f1=[0.920,0.970,0.530,0.940]
ypos=np.arange(len(cf2))
plt.xticks(ypos,cf2)
plt.xlabel('Classifiers')
plt.bar(ypos,f1,color='green')
plt.title('F measure')
```

Out[126]:
```
Text(0.5, 1.0, 'F measure')
```



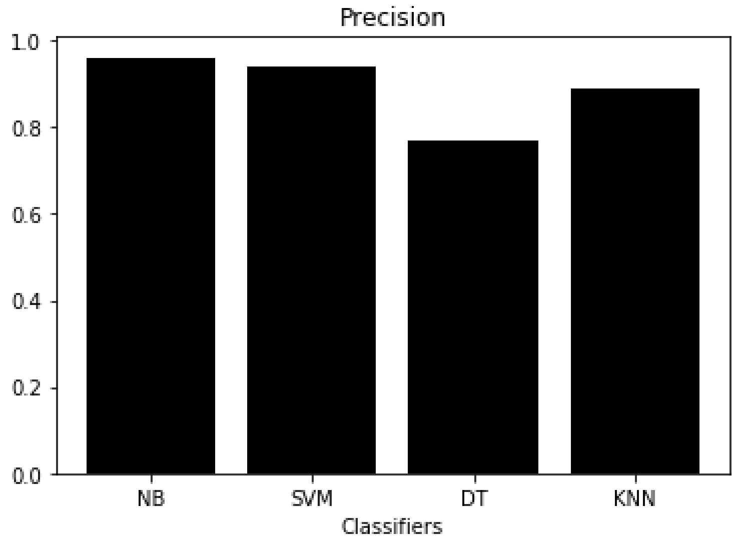
In [127]:
```
cf2=['NB','SVM','DT','KNN']
f1=[0.900,1.000,0.410,1.000]
ypos=np.arange(len(cf2))
plt.xticks(ypos,cf2)
plt.xlabel('Classifiers')
plt.bar(ypos,f1,color='blue')
plt.title('recall')
```

Out[127]:
```
Text(0.5, 1.0, 'recall')
```



In [128]:
```
cf2=['NB','SVM','DT','KNN']
f1=[0.960,0.940,0.770,0.890]

ypos=np.arange(len(cf2))
plt.xticks(ypos,cf2)
plt.xlabel('Classifiers')
plt.bar(ypos,f1,color='black')
plt.title('Precision')
```

Out[128]:
```
Text(0.5, 1.0, 'Precision')
```


Precision
1.0
0.8
0.6
0.4
0.2
0.0
NB
SVM
DT
KNN
Classifiers