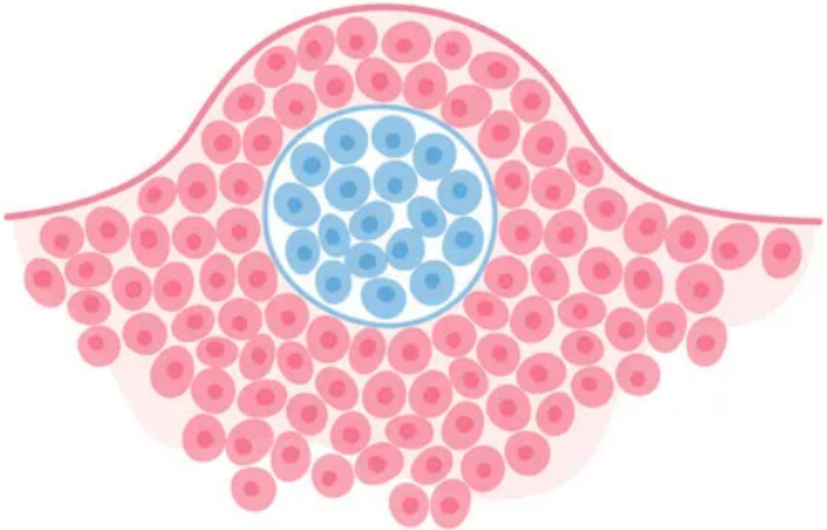
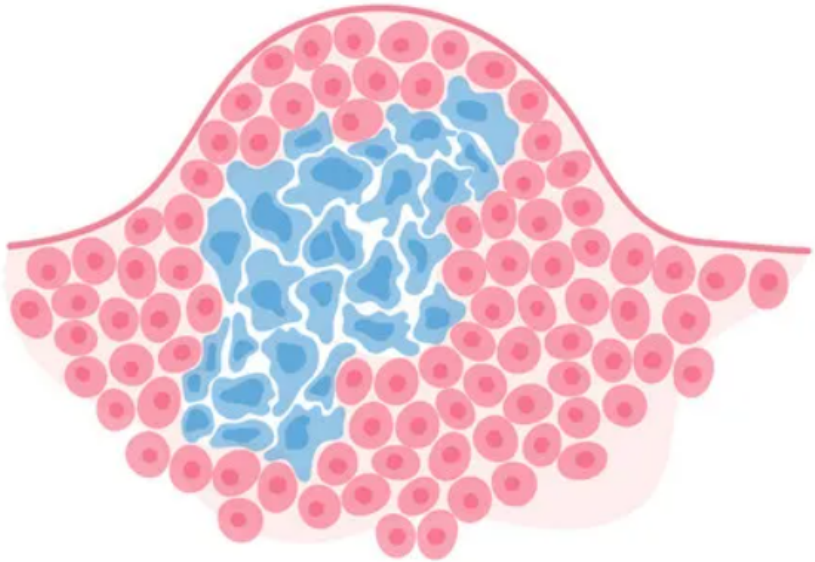
Principal Components Analysis



Benign tumor

- Non-cancerous
- Capsulated
- Non-invasive
- Slow growing
- Do not metastasize (spread) to other parts of the body
- Cells are normal



Malignant tumor

- Cancerous
- Non-capsulated
- Fast growing
- Metastasize (spread) to other parts of the body
- Cells have large, dark nuclei; may have abnormal shape

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
np.c_[data['data'], data['target']][0]
```

Out[3]:
```
array([1.799e+01, 1.038e+01, 1.228e+02, 1.001e+03, 1.184e-01, 2.776e-01,
       3.001e-01, 1.471e-01, 2.419e-01, 7.871e-02, 1.095e+00, 9.053e-01,
       8.589e+00, 1.534e+02, 6.399e-03, 4.904e-02, 5.373e-02, 1.587e-02,
       3.003e-02, 6.193e-03, 2.538e+01, 1.733e+01, 1.846e+02, 2.019e+03,
       1.622e-01, 6.656e-01, 7.119e-01, 2.654e-01, 4.601e-01, 1.189e-01,
       0.000e+00])
```

In [2]:
```
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
data
```

Out[2]:
```
{'data': array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
        1.189e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
        8.902e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
        8.758e-02],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
        7.820e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
        1.240e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
        7.039e-02]]),
 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 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, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 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, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 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, 1, 0, 0, 0, 0, 0, 0, 1]),
 'frame': None,
 'target_names': array(['malignant', 'benign'], dtype='<U9'),
 'DESCR': '.. _breast_cancer_dataset:\n\nBreast cancer wisconsin (diagnostic) dataset\n--------------------------------------
---\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 569\n\n    :Number of Attributes: 30 numeric, predictive attri
butes and the class\n\n    :Attribute Information:\n        - radius (mean of distances from center to points on the perimeter)
```

\n   - texture (standard deviation of gray-scale values)\n   - perimeter\n   - area\n   - smoothness (local variation in radius lengths)\n   - compactness (perimeter^2 / area - 1.0)\n   - concavity (severity of concave portions of the contour)\n   - concave points (number of concave portions of the contour)\n   - symmetry\n   - fractal dimension ("coastline approximation" - 1)\n\n   The mean, standard error, and "worst" or largest (mean of the three\n   worst/largest values) of these features were computed for each image,\n   resulting in 30 features. For instance, field 0 is Mean Radius, field\n   10 is Radius SE, field 20 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   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   concave 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 2.873\n   texture (standard error):   0.36   4.885\n   perimeter (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 points (standard error):   0.0   0.053\n   symmetry (standard error):   0.008 0.079\n   fractal dimension (standard error):   0.001 0.03\n   radius (worst):   7.93   36.04\n   texture (worst):   12.02 49.54\n   perimeter (worst):   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   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 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) datasets.\nhttps://goo.gl/U2Uwz2\n\nFeatures are computed from a digitized image of a fine needle\naspirate (FNA) of a breast mass. They describe\ncharacteristics of the cell nuclei present in the image.\n\nSeparating plane described above was obtained using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree\nConstruction Via Linear Programming." Proceedings of the 4th\nMidwest Artificial Intelligence and Cognitive Science Society,\npp. 97-101, 1992], a classification method which uses linear\nprogramming to construct a decision tree. Relevant features\nwere selected 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 separating plane\nin the 3-dimensional space is that described in:\n[K. P. Bennett and O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Linearly Inseparable Sets",\nOptimization Methods and Software 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\n.. topic:: References\n\n   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction \n   for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on \n   Electronic Imaging: Science 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 programming. Operations Research, 43(4), pages 570-577, \n   July-August 1995.\n   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques\n   to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) \n   163-171.',
 'feature_names': array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
        'mean smoothness', 'mean compactness', 'mean concavity',
        'mean concave points', 'mean symmetry', 'mean fractal dimension',
        'radius error', 'texture error', 'perimeter error', 'area error',
        'smoothness error', 'compactness error', 'concavity error',
        'concave points error', 'symmetry error',
        'fractal dimension error', 'worst radius', 'worst texture',

```
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23'),
 'filename': 'breast_cancer.csv',
 'data_module': 'sklearn.datasets.data'}
```

In [4]:
```
cancer = pd.DataFrame(np.c_[data['data'], data['target']], columns = np.append(data['feature_names'],['target']))

df = pd.read_csv('heart.csv')
cancer.head()
```

Out[4]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	17.33	184.60	2019.0	0.1622
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	23.41	158.80	1956.0	0.1238
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	25.53	152.50	1709.0	0.1444
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	26.50	98.87	567.7	0.2098
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	16.67	152.20	1575.0	0.1374

5 rows × 31 columns

In [5]:
```
data['target_names']
```

Out[5]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [6]:
```
cancer['target_names'] = cancer['target'].replace(to_replace=[0,1], value=['malignant','benign'])
```

In [7]:
```
cancer.shape
```

Out[7]:
```
(569, 32)
```

In [8]:
```
cancer.info()
```

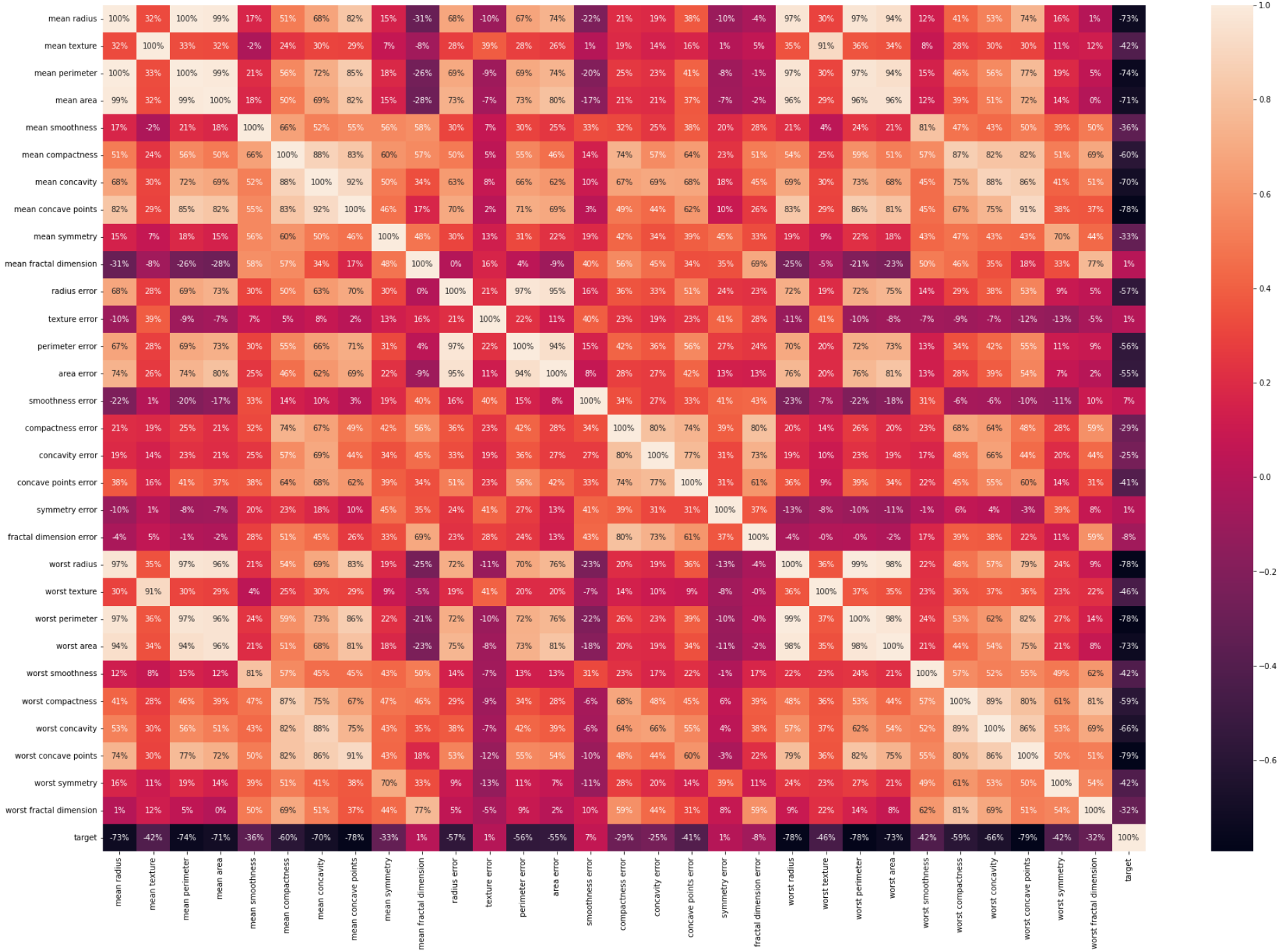
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   mean radius              569 non-null    float64
 1   mean texture             569 non-null    float64
 2   mean perimeter           569 non-null    float64
 3   mean area                569 non-null    float64
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
 30  target                   569 non-null    float64
 31  target_names             569 non-null    object 
dtypes: float64(31), object(1)
memory usage: 142.4+ KB
```

Correlation

In [9]:
```
cancer.corr()['target'].sort_values(ascending=False)
```

Out[9]:
```
target                    1.000000
smoothness error          0.067016
mean fractal dimension    0.012838
texture error             0.008303
symmetry error            0.006522
fractal dimension error  -0.077972
concavity error          -0.253730
compactness error        -0.292999
worst fractal dimension  -0.323872
mean symmetry            -0.330499
mean smoothness          -0.358560
concave points error     -0.408042
mean texture             -0.415185
worst symmetry           -0.416294
worst smoothness         -0.421465
worst texture            -0.456903
area error               -0.548236
perimeter error          -0.556141
radius error             -0.567134
worst compactness        -0.590998
mean compactness         -0.596534
worst concavity          -0.659610
mean concavity           -0.696360
mean area                -0.708984
mean radius              -0.730029
worst area               -0.733825
mean perimeter           -0.742636
worst radius             -0.776454
mean concave points      -0.776614
worst perimeter          -0.782914
worst concave points     -0.793566
Name: target, dtype: float64
```

In [10]:
```
plt.figure(figsize=(30,20))
sns.heatmap(cancer.corr(), annot=True, fmt='.0%')
plt.show()
```


mean radius 100% 32% 100% 99% 17% 51% 68% 82% 15% -31% 68% -10% 67% 74% -22% 21% 19% 38% -10% -4% 97% 30% 97% 94% 12% 41% 53% 74% 16% 1% -73%
mean texture 32% 100% 33% 32% -2% 24% 30% 29% 7% -8% 28% 39% 28% 26% 1% 19% 14% 16% 1% 5% 35% 91% 36% 34% 8% 28% 30% 30% 11% 12% -42%
mean perimeter 100% 33% 100% 99% 21% 56% 72% 85% 18% -26% 69% -9% 69% 74% -20% 25% 23% 41% -8% -1% 97% 30% 97% 94% 15% 46% 56% 77% 19% 5% -74%
mean area 99% 32% 99% 100% 18% 50% 69% 82% 15% -28% 73% -7% 73% 80% -17% 21% 21% 37% -7% -2% 96% 29% 96% 96% 12% 39% 51% 72% 14% 0% -71%
mean smoothness 17% -2% 21% 18% 100% 66% 52% 55% 56% 58% 30% 7% 30% 25% 33% 32% 25% 38% 20% 28% 21% 4% 24% 21% 81% 47% 43% 50% 39% 50% -36%
mean compactness 51% 24% 56% 50% 66% 100% 88% 83% 60% 57% 50% 5% 55% 46% 14% 74% 57% 64% 23% 51% 54% 25% 59% 51% 57% 87% 82% 82% 51% 69% -60%
mean concavity 68% 30% 72% 69% 52% 88% 100% 92% 50% 34% 63% 8% 66% 62% 10% 67% 69% 68% 18% 45% 69% 30% 73% 68% 45% 75% 88% 86% 41% 51% -70%
mean concave points 82% 29% 85% 82% 55% 83% 92% 100% 46% 17% 70% 2% 71% 69% 3% 49% 44% 62% 10% 26% 83% 29% 86% 81% 45% 67% 75% 91% 38% 37% -78%
mean symmetry 15% 7% 18% 15% 56% 60% 50% 46% 100% 48% 30% 13% 31% 22% 19% 42% 34% 39% 45% 33% 19% 9% 22% 18% 43% 47% 43% 43% 70% 44% -33%
mean fractal dimension -31% -8% -26% -28% 58% 57% 34% 17% 48% 100% 0% 16% 4% -9% 40% 56% 45% 34% 35% 69% -25% -5% -21% -23% 50% 46% 35% 18% 33% 77% 1%
radius error 68% 28% 69% 73% 30% 50% 63% 70% 30% 0% 100% 21% 97% 95% 16% 36% 33% 51% 24% 23% 72% 19% 72% 75% 14% 29% 38% 53% 9% 5% -57%
texture error -10% 39% -9% -7% 7% 5% 8% 2% 13% 16% 21% 100% 22% 11% 40% 23% 19% 23% 41% 28% -11% 41% -10% -8% -7% -9% -7% -12% -13% -5% 1%
perimeter error 67% 28% 69% 73% 30% 55% 66% 71% 31% 4% 97% 22% 100% 94% 15% 42% 36% 56% 27% 24% 70% 20% 72% 73% 13% 34% 42% 55% 11% 9% -56%
area error 74% 26% 74% 80% 25% 46% 62% 69% 22% -9% 95% 11% 94% 100% 8% 28% 27% 42% 13% 13% 76% 20% 76% 81% 13% 28% 39% 54% 7% 2% -55%
smoothness error -22% 1% -20% -17% 33% 14% 10% 3% 19% 40% 16% 40% 15% 8% 100% 34% 27% 33% 41% 43% -23% -7% -22% -18% 31% -6% -6% -10% -11% 10% 7%
compactness error 21% 19% 25% 21% 32% 74% 67% 49% 42% 56% 36% 23% 42% 28% 34% 100% 80% 74% 39% 80% 20% 14% 26% 20% 23% 68% 64% 48% 28% 59% -29%
concavity error 19% 14% 23% 21% 25% 57% 69% 44% 34% 45% 33% 19% 36% 27% 27% 80% 100% 77% 31% 73% 19% 10% 23% 19% 17% 48% 66% 44% 20% 44% -25%
concave points error 38% 16% 41% 37% 38% 64% 68% 62% 39% 34% 51% 23% 56% 42% 33% 74% 77% 100% 31% 61% 36% 9% 39% 34% 22% 45% 55% 60% 14% 31% -41%
symmetry error -10% 1% -8% -7% 20% 23% 18% 10% 45% 35% 24% 41% 27% 13% 41% 39% 31% 31% 100% 37% -13% -8% -10% -11% -1% 6% 4% -3% 39% 8% 1%
fractal dimension error -4% 5% -1% -2% 28% 51% 45% 26% 33% 69% 23% 28% 24% 13% 43% 80% 73% 61% 37% 100% -4% -0% -0% -2% 17% 39% 38% 22% 11% 59% -8%
worst radius 97% 35% 97% 96% 21% 54% 69% 83% 19% -25% 72% -11% 70% 76% -23% 20% 19% 36% -13% -4% 100% 36% 99% 98% 22% 48% 57% 79% 24% 9% -78%
worst texture 30% 91% 30% 29% 4% 25% 30% 29% 9% -5% 19% 41% 20% 20% -7% 14% 10% 9% -8% -0% 36% 100% 37% 35% 23% 36% 37% 36% 23% 22% -46%
worst perimeter 97% 36% 97% 96% 24% 59% 73% 86% 22% -21% 72% -10% 72% 76% -22% 26% 23% 39% -10% -0% 99% 37% 100% 98% 24% 53% 62% 82% 27% 14% -78%
worst area 94% 34% 94% 96% 21% 51% 68% 81% 18% -23% 75% -8% 73% 81% -18% 20% 19% 34% -11% -2% 98% 35% 98% 100% 21% 44% 54% 75% 21% 8% -73%
worst smoothness 12% 8% 15% 12% 81% 57% 45% 45% 43% 50% 14% -7% 13% 13% 31% 23% 17% 22% -1% 17% 22% 23% 24% 21% 100% 57% 52% 55% 49% 62% -42%
worst compactness 41% 28% 46% 39% 47% 87% 75% 67% 47% 46% 29% -9% 34% 28% -6% 68% 48% 45% 6% 39% 48% 36% 53% 44% 57% 100% 89% 80% 61% 81% -59%
worst concavity 53% 30% 56% 51% 43% 82% 88% 75% 43% 35% 38% -7% 42% 39% -6% 64% 66% 55% 4% 38% 57% 37% 62% 54% 52% 89% 100% 86% 53% 69% -66%
worst concave points 74% 30% 77% 72% 50% 82% 86% 91% 43% 18% 53% -12% 55% 54% -10% 48% 44% 60% -3% 22% 79% 36% 82% 75% 55% 80% 86% 100% 50% 51% -79%
worst symmetry 16% 11% 19% 14% 39% 51% 41% 38% 70% 33% 9% -13% 11% 7% -11% 28% 20% 14% 39% 11% 24% 23% 27% 21% 49% 61% 53% 50% 100% 54% -42%
worst fractal dimension 1% 12% 5% 0% 50% 69% 51% 37% 44% 77% 5% -5% 9% 2% 10% 59% 44% 31% 8% 59% 9% 22% 14% 8% 62% 81% 69% 51% 54% 100% -32%
target -73% -42% -74% -71% -36% -60% -70% -78% -33% 1% -57% 1% -56% -55% 7% -29% -25% -41% 1% -8% -78% -46% -78% -73% -42% -59% -66% -79% -42% -32% 100%
mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
perimeter error
area error
smoothness error
compactness error
concavity error
concave points error
symmetry error
fractal dimension error
worst radius
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension
target
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

In [11]:
```
cancer.describe()
```

Out[11]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	per
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	...	569.000000	569.
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	...	25.677223	107.
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	...	6.146258	33.
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	...	12.020000	50.
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	...	21.080000	84.
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	...	25.410000	97.
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	...	29.720000	125.
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	...	49.540000	251.

8 rows × 31 columns

Skewness

In [12]:
```
cancer.skew(numeric_only=True).sort_values(ascending=False)
```

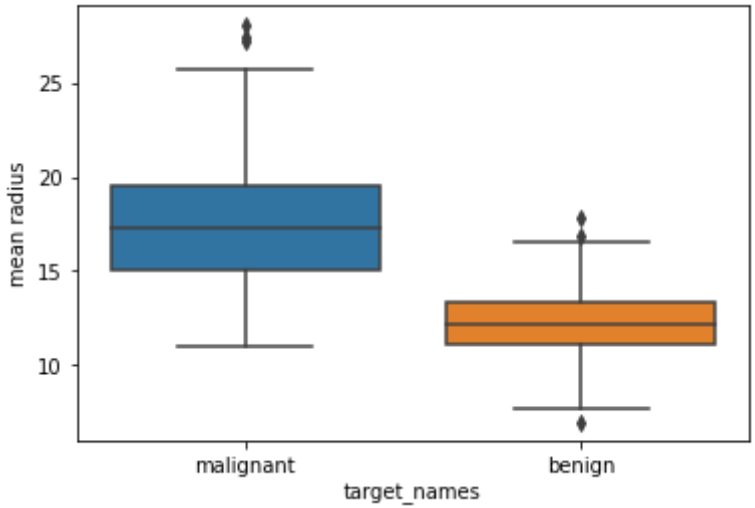
Out[12]:
```
area error                 5.447186
concavity error            5.110463
fractal dimension error    3.923969
perimeter error            3.443615
radius error               3.088612
smoothness error           2.314450
symmetry error             2.195133
compactness error          1.902221
worst area                 1.859373
worst fractal dimension    1.662579
texture error              1.646444
mean area                  1.645732
worst compactness          1.473555
concave points error       1.444678
worst symmetry             1.433928
mean concavity             1.401180
mean fractal dimension     1.304489
mean compactness           1.190123
mean concave points        1.171180
worst concavity            1.150237
worst perimeter            1.128164
worst radius               1.103115
mean perimeter             0.990650
mean radius                0.942380
mean symmetry              0.725609
mean texture               0.650450
worst texture              0.498321
worst concave points       0.492616
mean smoothness            0.456324
worst smoothness           0.415426
target                    -0.528461
dtype: float64
```

In [13]:
```
cancer.iloc[:,30]
```

Out[13]:
```
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0
      ...
564    0.0
565    0.0
566    0.0
567    0.0
568    1.0
Name: target, Length: 569, dtype: float64
```

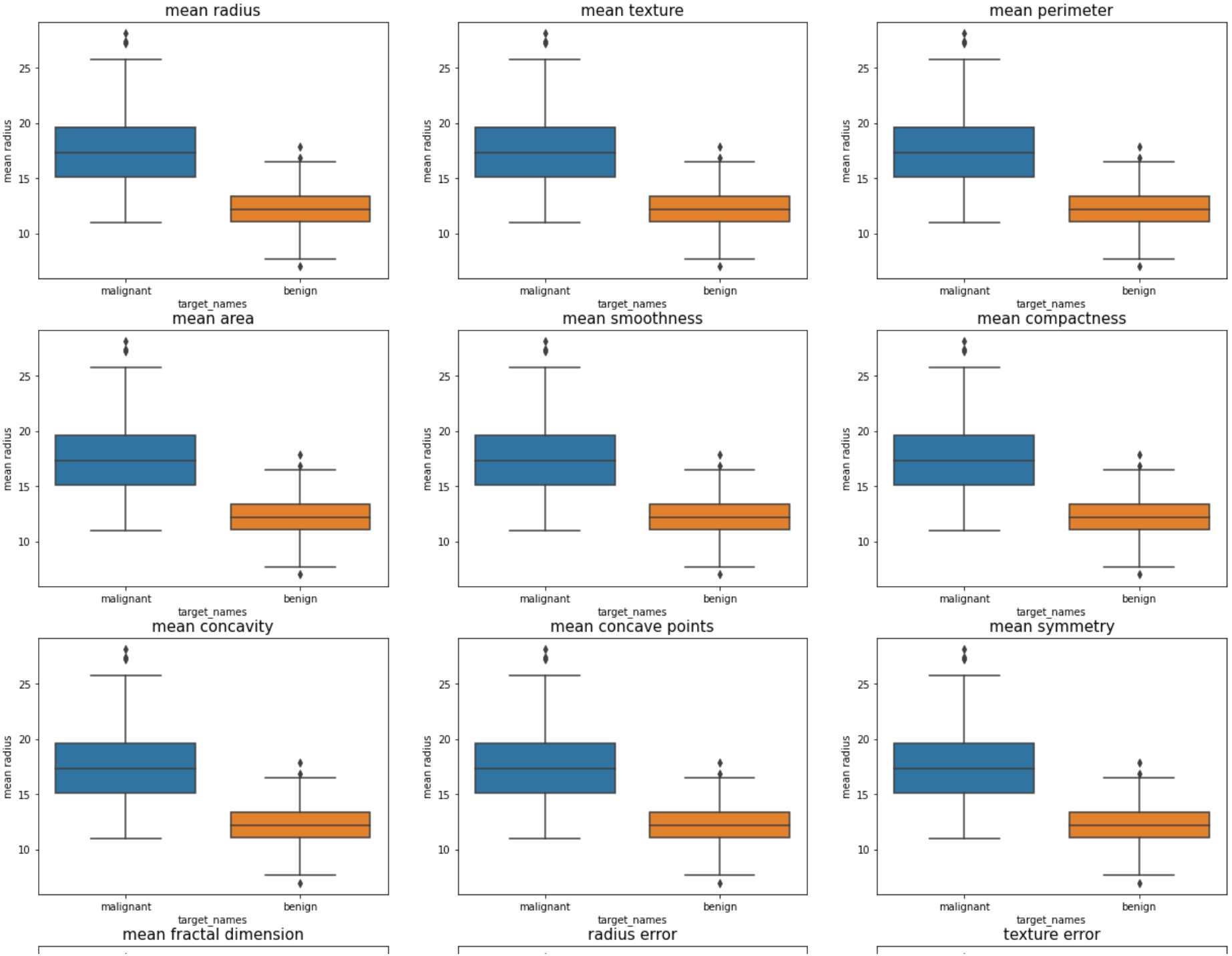
Boxplot

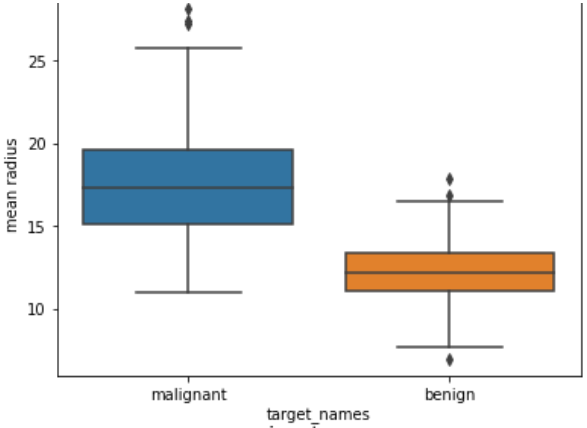
In [14]:
```
#ax = sns.boxplot(x=Cancer[Cancer.columns[0]])
sns.boxplot(x=cancer[cancer.columns[31]],y=cancer[cancer.columns[0]],data=cancer)
plt.show()
```

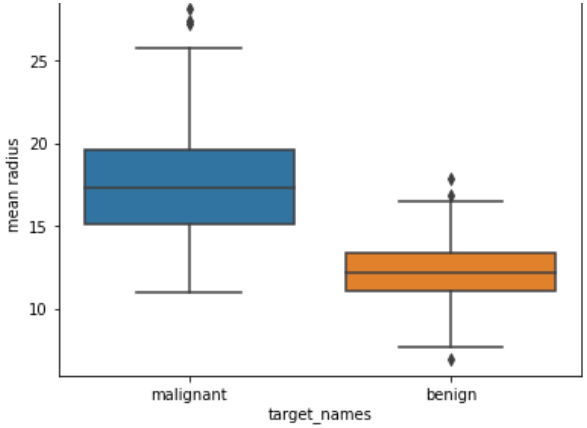


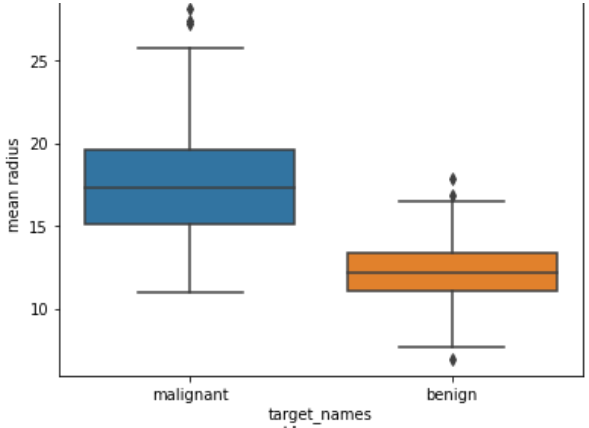
In [15]:
```
plt.figure(figsize=(21,55))

for i in range(30):
    plt.subplot(10,3,i+1)
    ax = sns.boxplot(x=cancer[cancer.columns[31]],y=cancer[cancer.columns[0]],data=cancer)
    #ax = sns.swarmplot(x=cancer[cancer.columns[31]],y=cancer[cancer.columns[0]],color=".35", data=cancer)
```

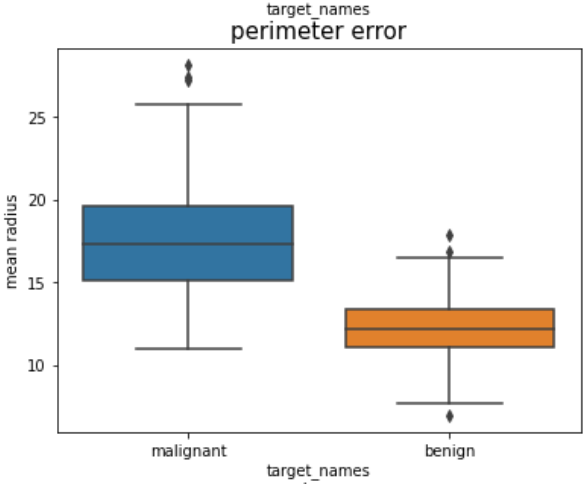
```
    #ax = sns.swarmplot(y=Cancer[Cancer.columns[i]], color=".30")
    plt.title(cancer.columns[i], fontsize=15)

plt.show()
```

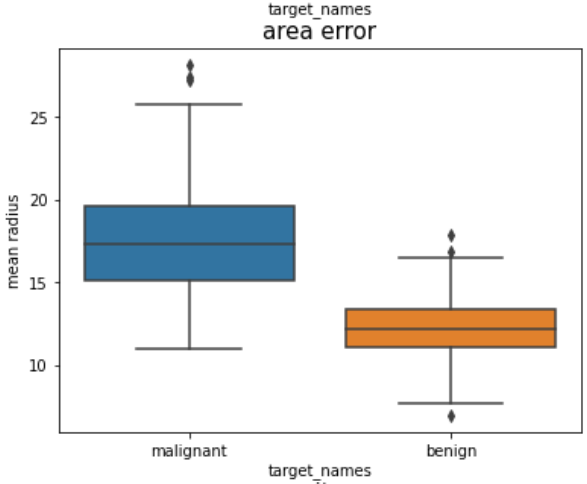

mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
mean radius
target_names
malignant
benign
10
15
20
25

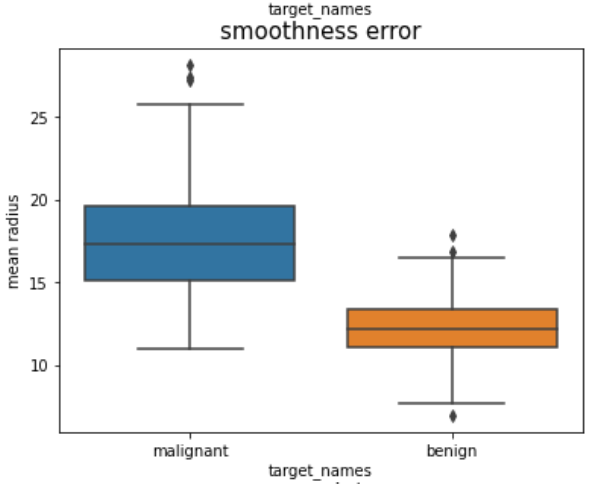

mean radius
25
20
15
10
malignant
benign
target_names

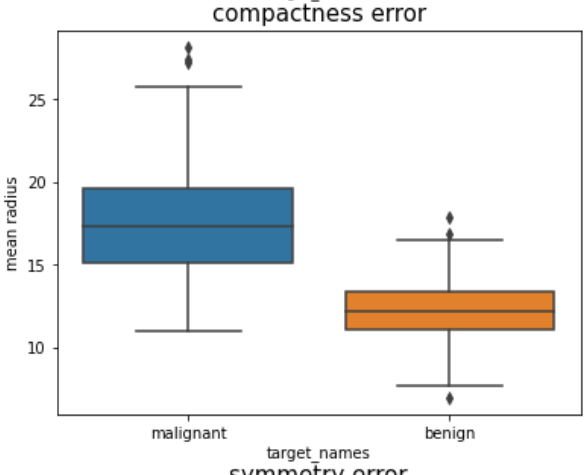

mean radius
25
20
15
10
malignant
benign
target_names

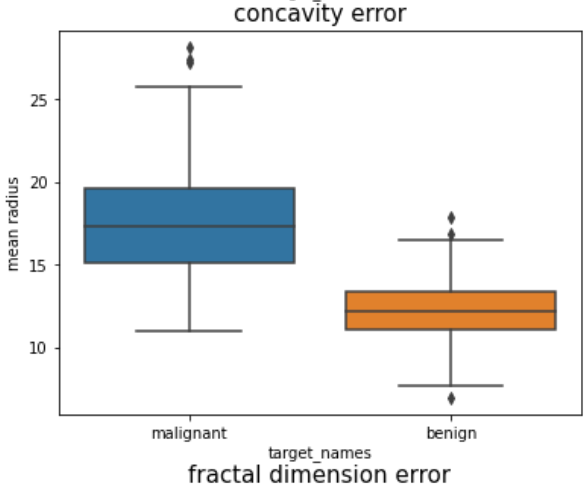

mean radius
25
20
15
10
malignant
benign
target_names

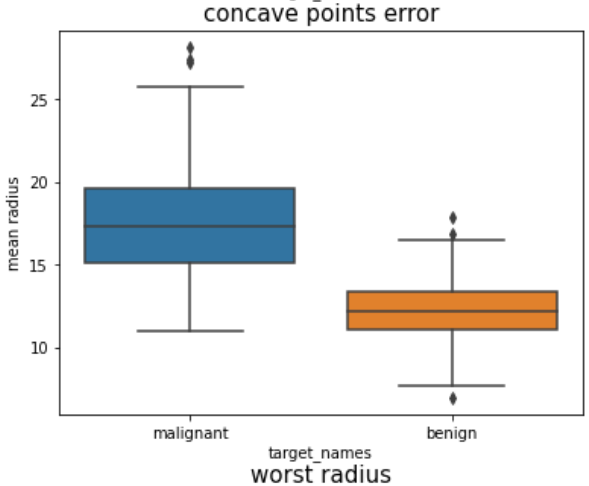

perimeter error
mean radius
25
20
15
10
malignant
benign
target_names

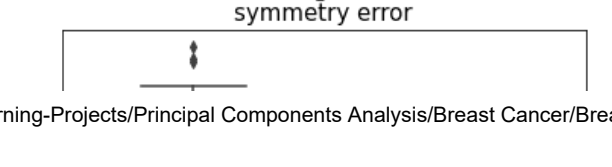

area error
mean radius
25
20
15
10
malignant
benign
target_names

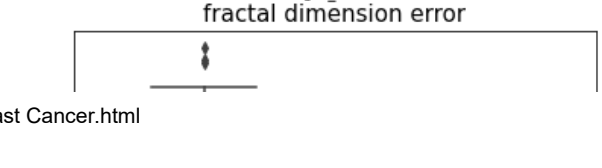

smoothness error
mean radius
25
20
15
10
malignant
benign
target_names

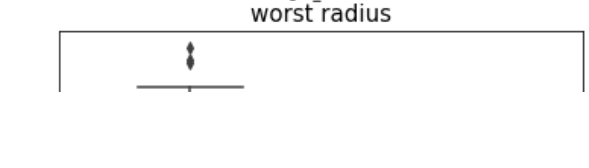

compactness error
mean radius
25
20
15
10
malignant
benign
target_names

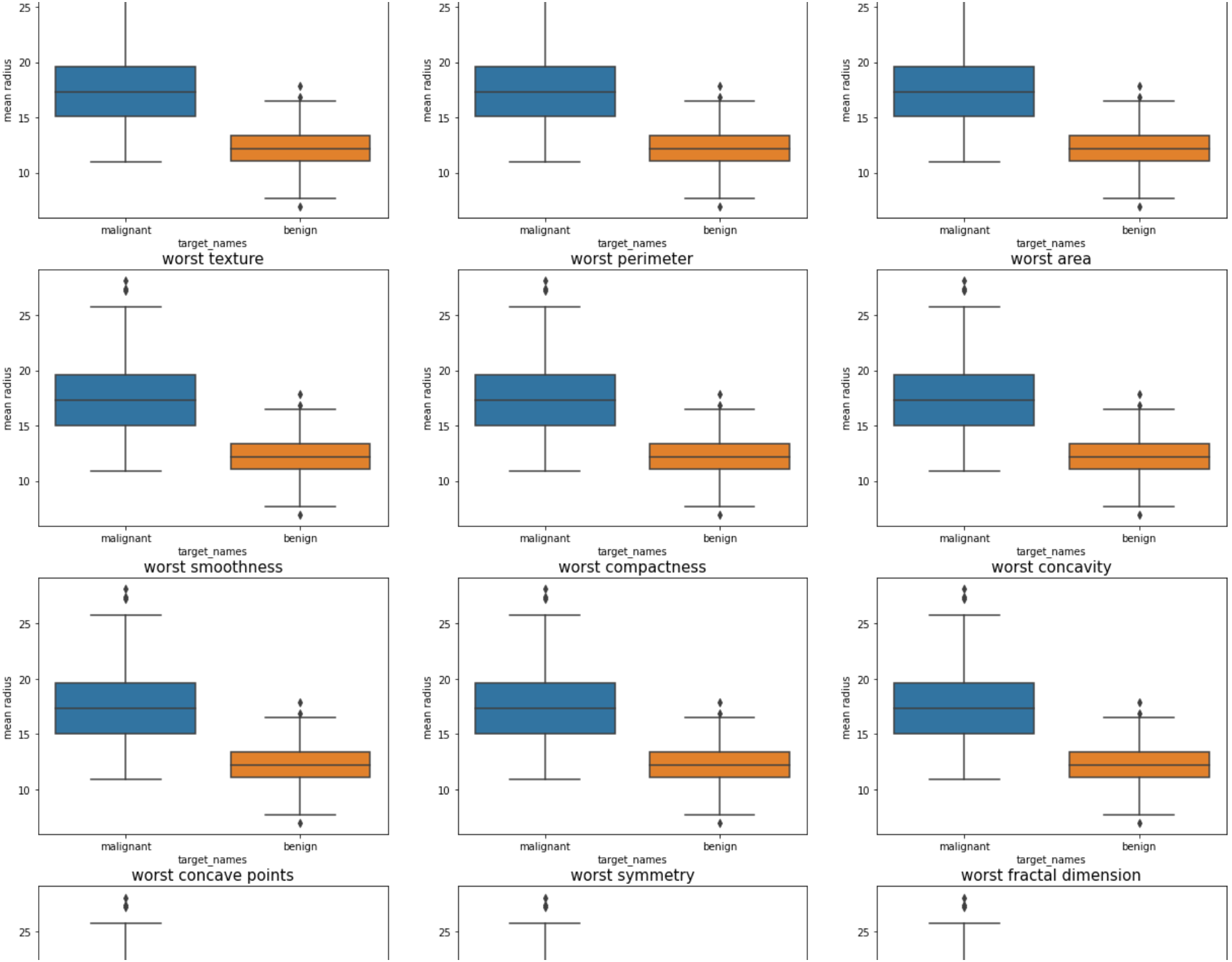

concavity error
mean radius
25
20
15
10
malignant
benign
target_names

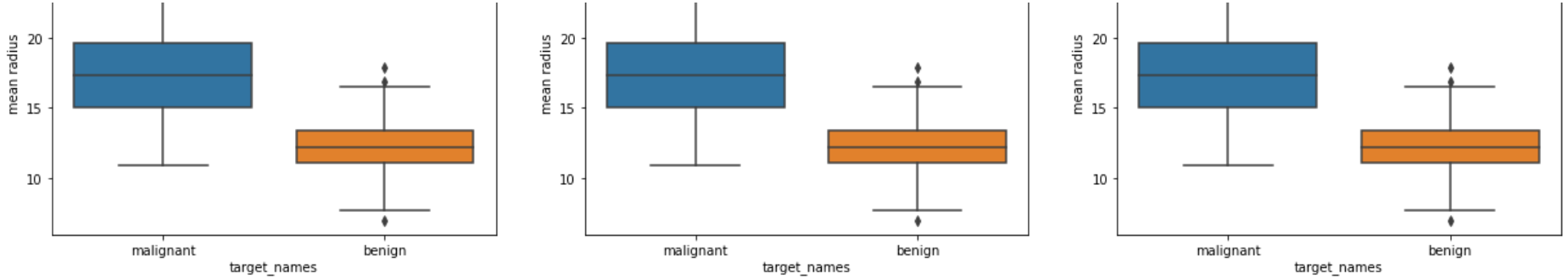

concave points error
mean radius
25
20
15
10
malignant
benign
target_names


symmetry error


fractal dimension error


worst radius


mean radius
target_names
malignant
benign
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension



Z Score

In [16]:
```
from scipy import stats
z = stats.zscore(cancer['mean radius'])
z_abs = np.abs(z)
np.where(z_abs > 3)
```

Out[16]:
```
(array([ 82, 180, 212, 352, 461], dtype=int64),)
```

- Lower bound = Q1 - (1.5 * IQR)
- Upper bound = Q3 + (1.5 * IQR)

In [17]:
```
Q1 = np.percentile(cancer['mean radius'], 20, interpolation = 'midpoint')
Q3 = np.percentile(cancer['mean radius'], 80, interpolation = 'midpoint')
IQR = Q3 - Q1
IQR
```

Out[17]:
```
5.705000000000002
```

In [18]:
```
upper_bound = cancer['mean radius'] >= (Q3+1.5*IQR)
lower_bound = cancer['mean radius'] <= (Q1-1.5*IQR)
```

In [19]:
```
np.where(upper_bound)
```

Out[19]:
```
(array([180, 212, 352, 461], dtype=int64),)
```

In [20]:
```
np.where(lower_bound)
```

Out[20]:
```
(array([], dtype=int64),)
```

In [21]:
```
upper_points = np.where(upper_bound)
#Cancer.drop(upper_points[0], inplace=True)
```

In [22]:
```
for i in range(30):

    z = stats.zscore(cancer[cancer.columns[i]])
    z_abs = np.abs(z)
    #print(Cancer.columns[i], np.where(z_abs > 3))

    print(cancer.columns[i],' : ', np.where(z_abs>3))
```

```
mean radius  :  (array([ 82, 180, 212, 352, 461], dtype=int64),)
mean texture  :  (array([219, 232, 239, 259], dtype=int64),)
mean perimeter  :  (array([ 82, 122, 180, 212, 352, 461, 521], dtype=int64),)
mean area  :  (array([ 82, 122, 180, 212, 339, 352, 461, 521], dtype=int64),)
mean smoothness  :  (array([  3, 105, 122, 504, 568], dtype=int64),)
mean compactness  :  (array([  0,   3,  78,  82, 108, 122, 181, 258, 567], dtype=int64),)
mean concavity  :  (array([ 78,  82, 108, 122, 152, 202, 352, 461, 567], dtype=int64),)
mean concave points  :  (array([ 82, 108, 122, 180, 352, 461], dtype=int64),)
mean symmetry  :  (array([ 25,  60,  78, 122, 146], dtype=int64),)
mean fractal dimension  :  (array([  3,  71, 152, 318, 376, 504, 505], dtype=int64),)
radius error  :  (array([122, 138, 212, 258, 417, 461, 503], dtype=int64),)
texture error  :  (array([ 12,  83, 122, 192, 416, 473, 557, 559, 561], dtype=int64),)
perimeter error  :  (array([ 12, 108, 122, 212, 258, 417, 461, 503], dtype=int64),)
area error  :  (array([122, 212, 265, 368, 461, 503], dtype=int64),)
smoothness error  :  (array([ 71, 116, 122, 213, 314, 345, 505], dtype=int64),)
compactness error  :  (array([ 12,  42,  68,  71, 108, 122, 152, 176, 190, 213, 288, 290],
      dtype=int64),)
concavity error  :  (array([ 68, 112, 122, 152, 213, 376], dtype=int64),)
concave points error  :  (array([ 12,  68, 152, 213, 288, 389], dtype=int64),)
symmetry error  :  (array([  3,  42,  78, 119, 122, 138, 146, 190, 212, 314, 351], dtype=int64),)
fractal dimension error  :  (array([ 12,  71, 112, 151, 152, 176, 213, 290, 376, 388], dtype=int64),)
worst radius  :  (array([180, 236, 265, 352, 461, 503], dtype=int64),)
worst texture  :  (array([219, 239, 259, 265], dtype=int64),)
worst perimeter  :  (array([ 82, 180, 265, 352, 461, 503], dtype=int64),)
worst area  :  (array([ 23, 180, 236, 265, 339, 352, 368, 461, 503, 521], dtype=int64),)
worst smoothness  :  (array([  3, 203, 379], dtype=int64),)
worst compactness  :  (array([  3,   9,  14,  42,  72, 181, 190, 379, 562, 567], dtype=int64),)
worst concavity  :  (array([  9,  68, 108, 400, 430, 562, 567], dtype=int64),)
worst concave points  :  (array([], dtype=int64),)
worst symmetry  :  (array([  3,  31,  35,  78, 119, 146, 190, 323, 370], dtype=int64),)
worst fractal dimension  :  (array([  3,   9,  14,  31, 105, 151, 190, 379, 562], dtype=int64),)
```

In [23]:

```
from scipy import stats

index = []
k=[]

for i in range(30):

    z = stats.zscore(cancer[cancer.columns[i]])
    z_abs = np.abs(z)
    #print(Cancer.columns[i], np.where(z_abs > 3))
    Q1 = np.percentile(cancer[cancer.columns[i]], 25, interpolation = 'midpoint')
    Q3 = np.percentile(cancer[cancer.columns[i]], 75, interpolation = 'midpoint')
```

```
    IQR = Q3 - Q1
    upper_bound = cancer[cancer.columns[i]] > (Q3+3*IQR)
    lower_bound = cancer[cancer.columns[i]] <= (Q1-3*IQR)

    for j in range(569):
        if upper_bound[j]==True:
            index.append(j)
    for p in range(569):
        if lower_bound[p]==True:
            k.append(p)


    #Cancer.drop(upper_points[0], inplace=True)
    #Cancer.reset_index()
    #print(i, Cancer.shape)
len(set(index)), len(set(k))
```

Out[23]: (56, 0)

In [24]:
```
Cancer=cancer
```

In [25]:
```
Cancer.drop(np.array(index), inplace=True)
Cancer.shape,cancer.shape
```

Out[25]: ((513, 32), (513, 32))

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

Out[26]: (513, 32)

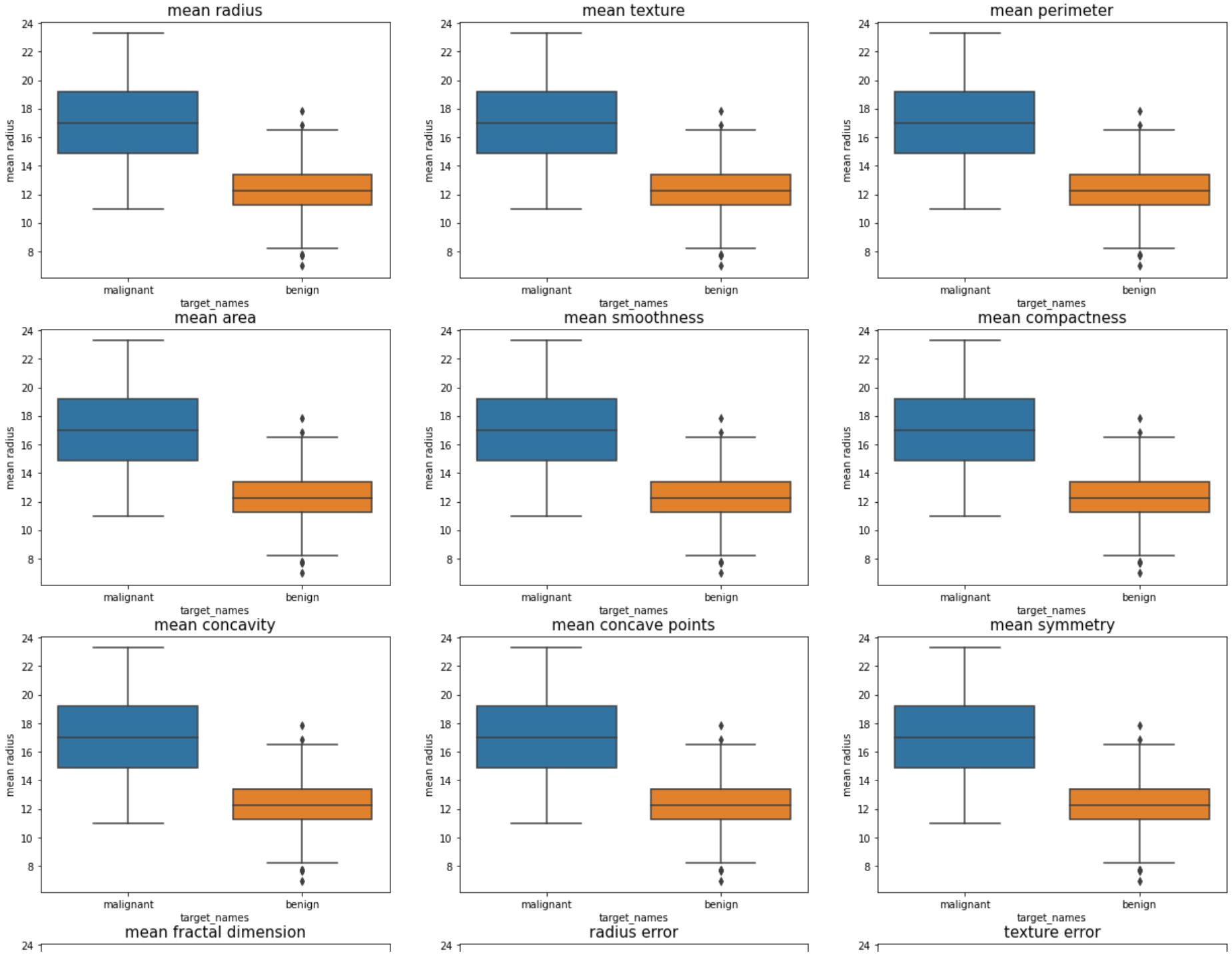
In [27]:
```
Cancer.shape
```

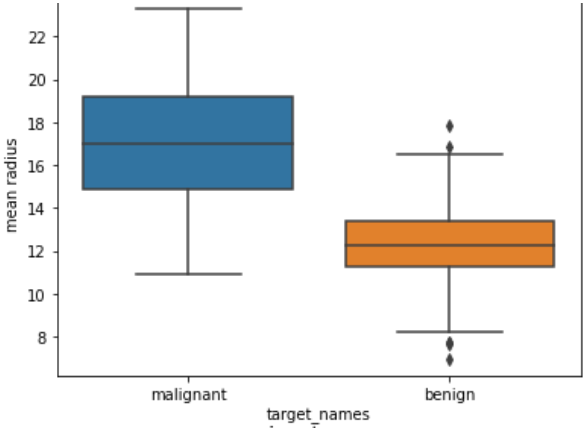
Out[27]: (513, 32)

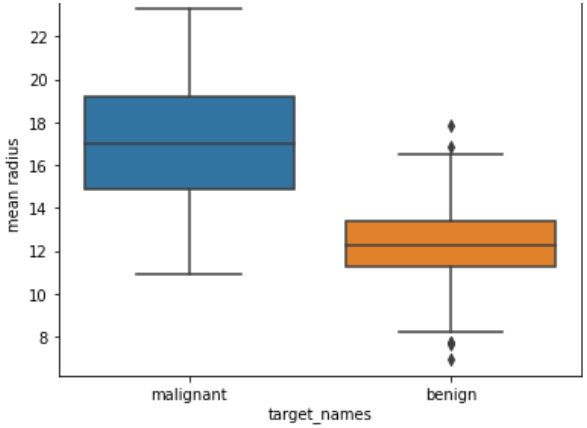
After removing the outliers

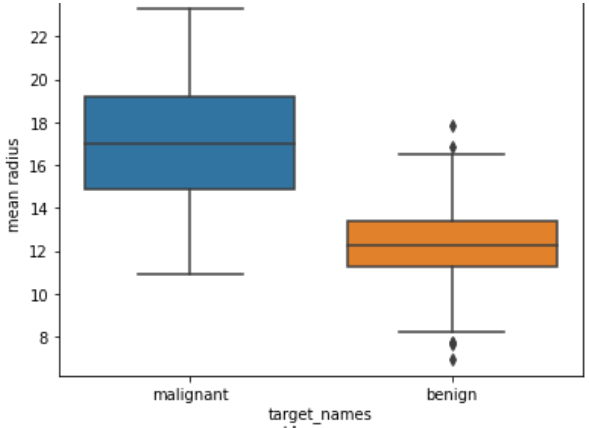
In [28]:
```
plt.figure(figsize=(21,55))
plt.title('Box Plot', fontsize = 25)
for i in range(30):
    plt.subplot(10,3,i+1)
```

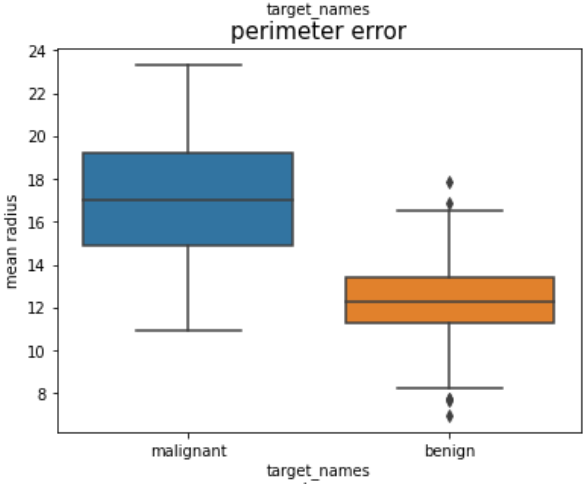
```
    ax = sns.boxplot(x=Cancer[Cancer.columns[31]],y=Cancer[Cancer.columns[0]],data=Cancer)
    #ax = sns.swarmplot(x=Cancer[Cancer.columns[31]],y=Cancer[Cancer.columns[0]],color=".35", data=Cancer)
    plt.title(Cancer.columns[i], fontsize=15)
plt.show()
```

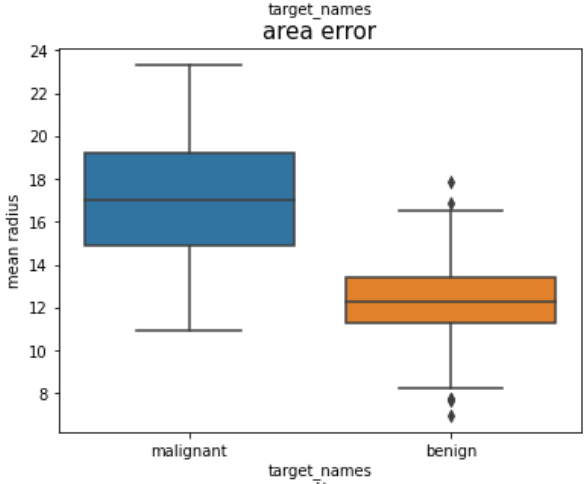

mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
mean radius
target_names
malignant
benign

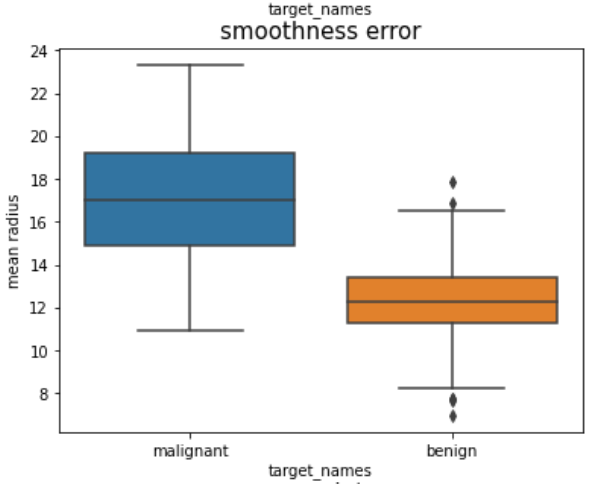

mean radius
target_names
malignant
benign

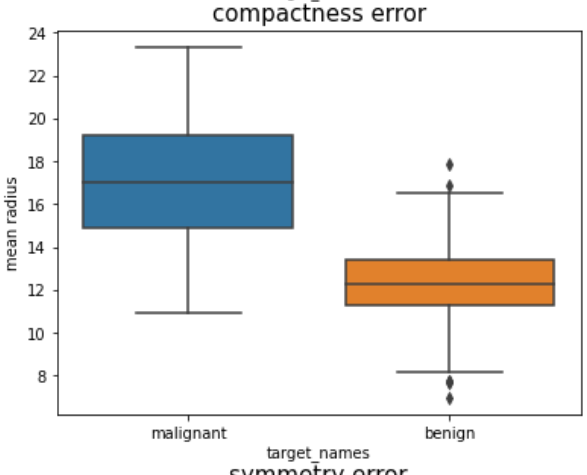

mean radius
target_names
malignant
benign

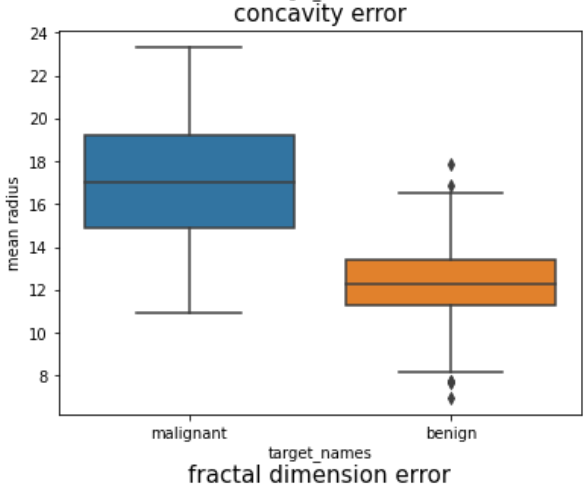

mean radius
target_names
malignant
benign

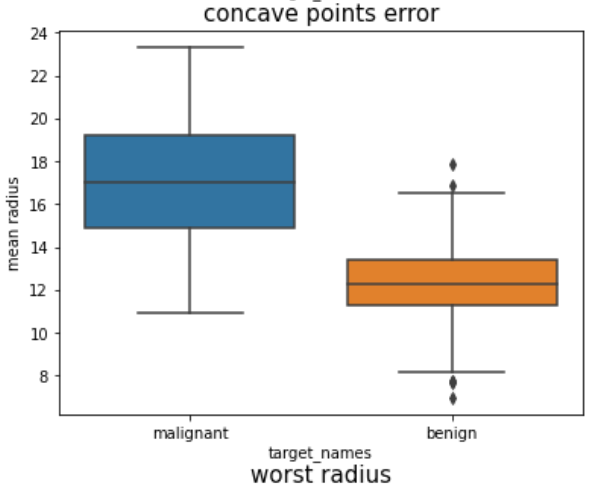

perimeter error
mean radius
target_names
malignant
benign

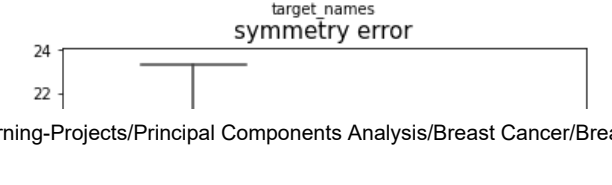

area error
mean radius
target_names
malignant
benign

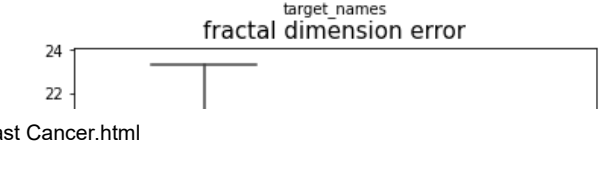

smoothness error
mean radius
target_names
malignant
benign

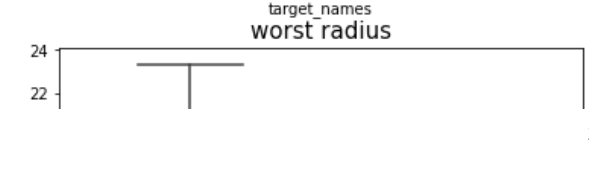

compactness error
mean radius
target_names
malignant
benign

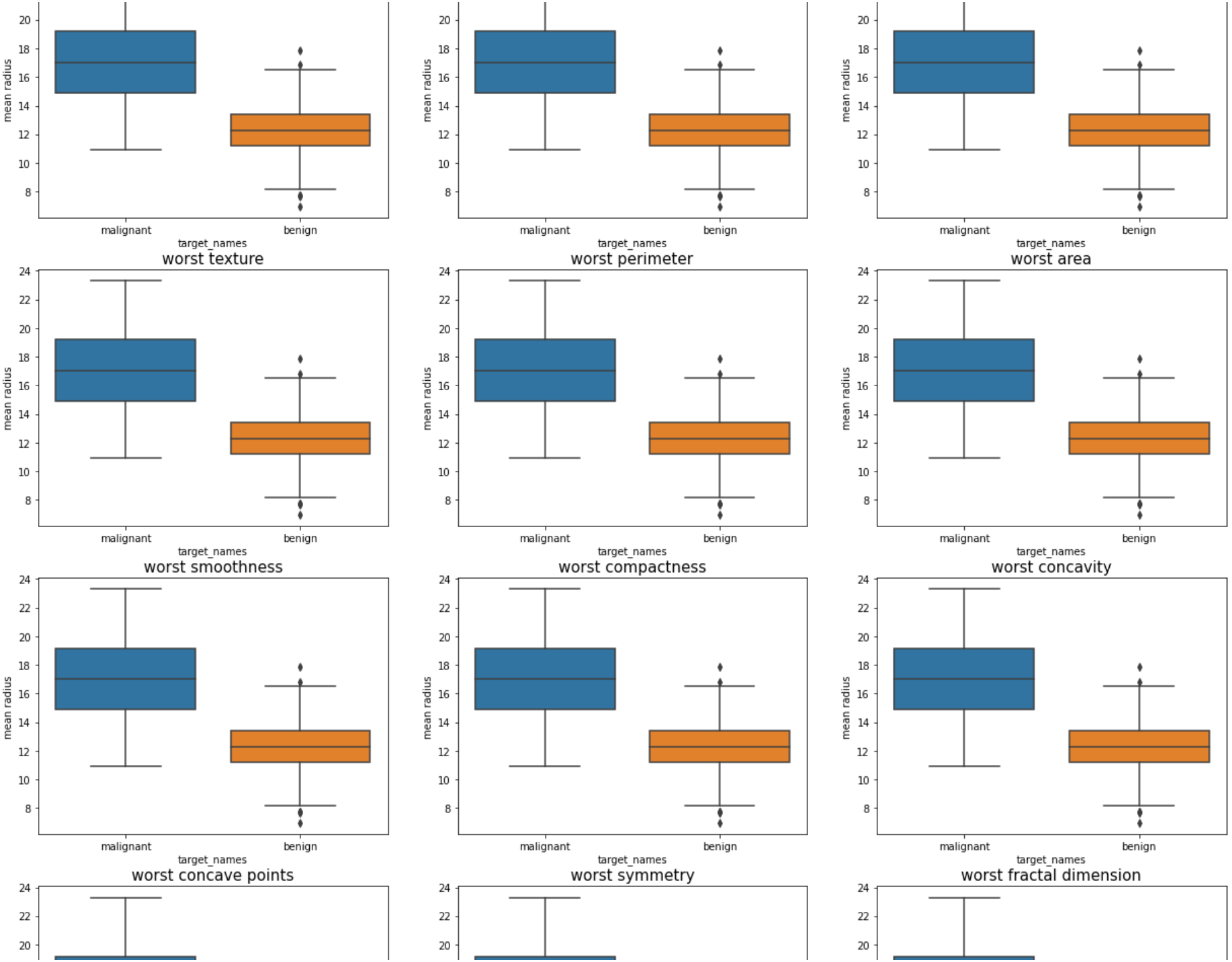

concavity error
mean radius
target_names
malignant
benign

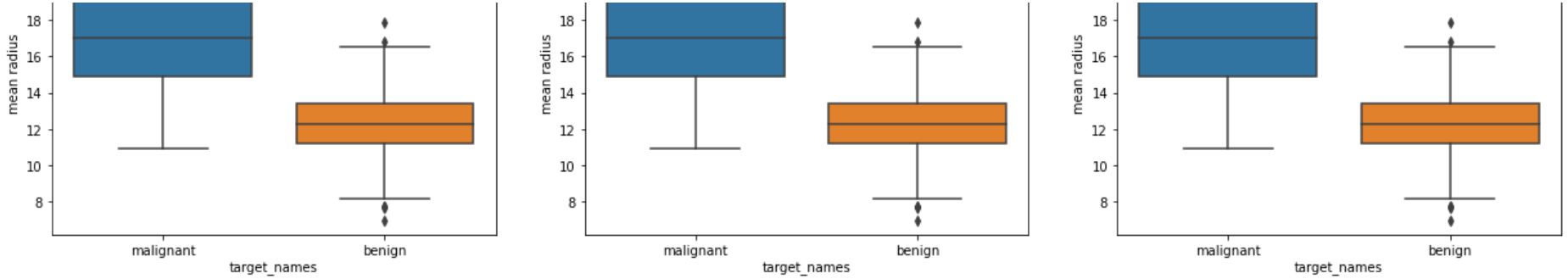

concave points error
mean radius
target_names
malignant
benign


symmetry error

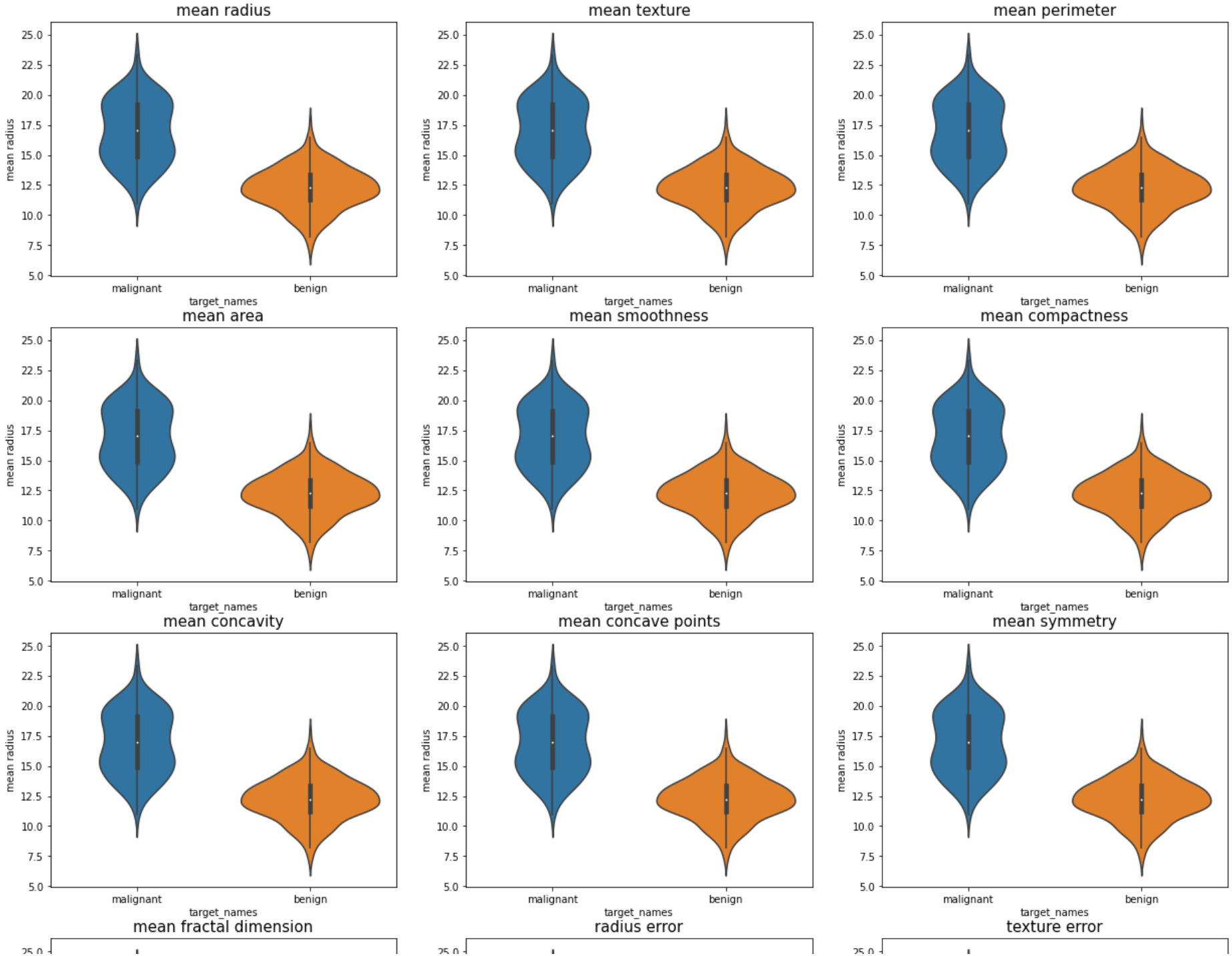

fractal dimension error

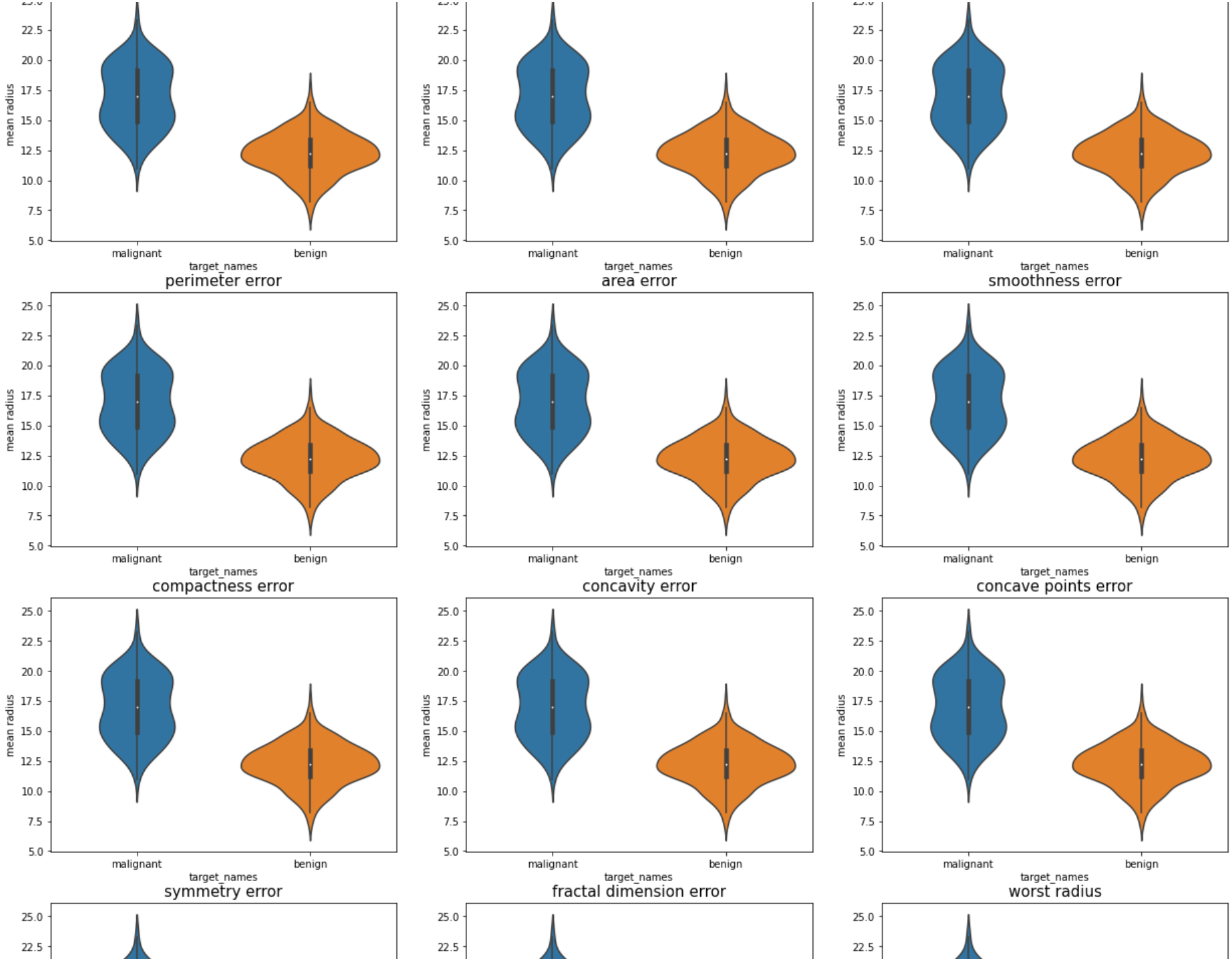

worst radius

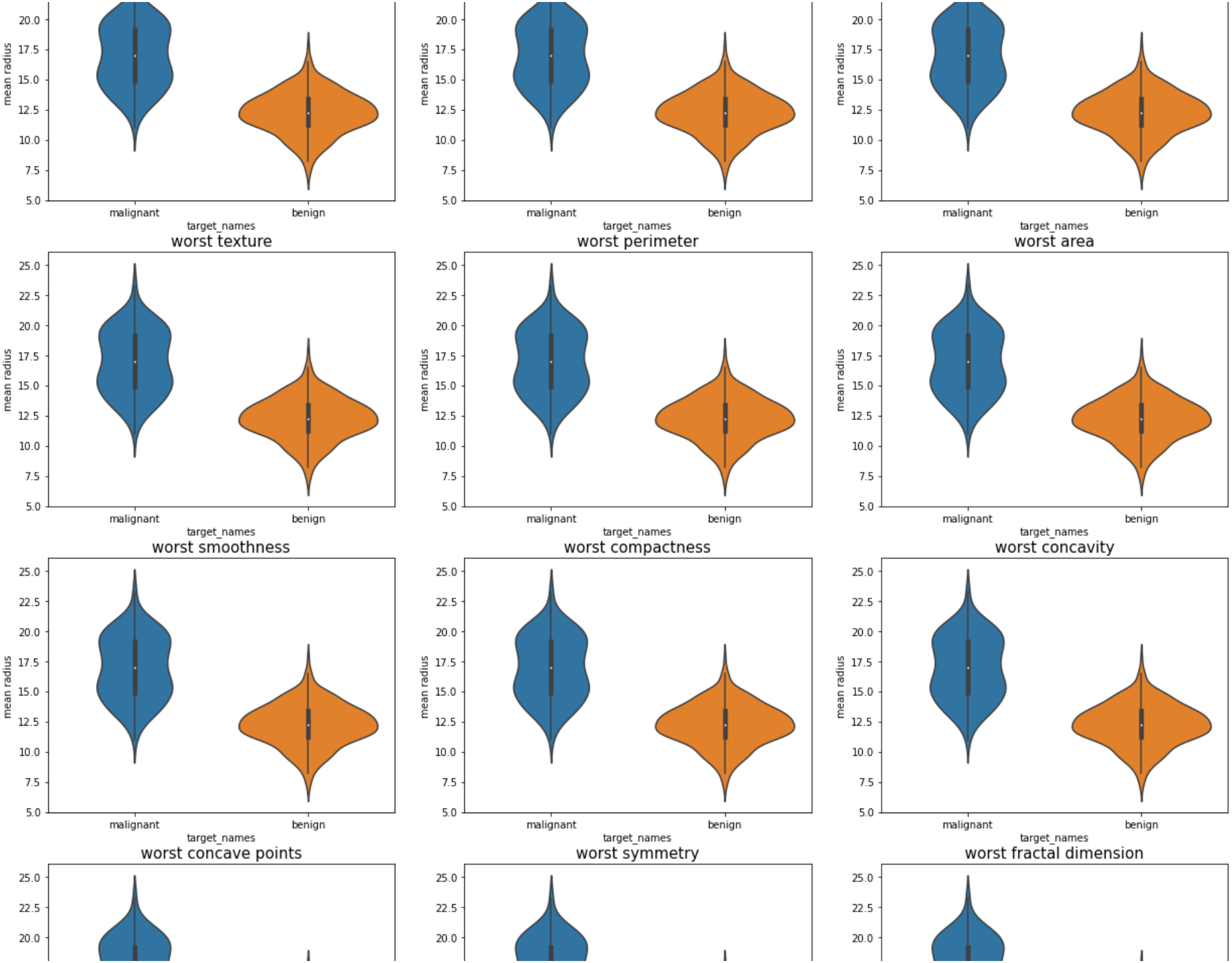

mean radius
target_names
malignant
benign
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension

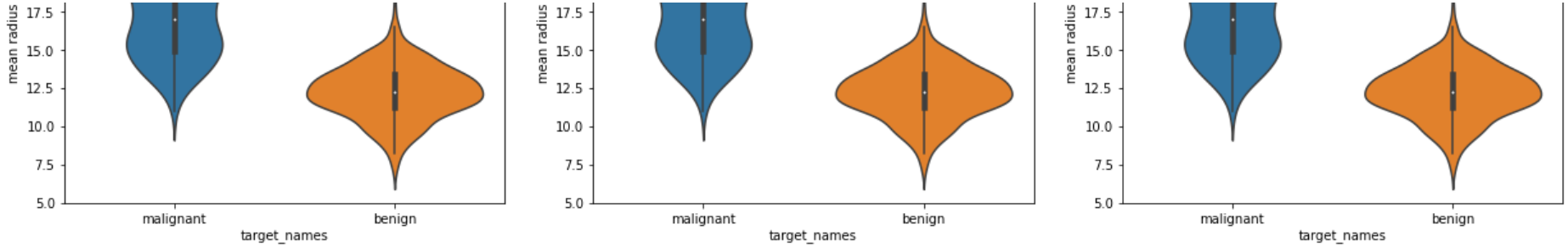


In [29]:
```
plt.figure(figsize=(21,55))

for i in range(30):
    plt.subplot(10,3,i+1)
    ax = sns.violinplot(x=Cancer[Cancer.columns[31]],y=Cancer[Cancer.columns[0]],data=Cancer)
    #ax = sns.swarmplot(x=Cancer[cancer.columns[31]],y=cancer[Cancer.columns[0]],color=".35", data=cancer)
    #ax = sns.swarmplot(y=Cancer[Cancer.columns[i]], color=".30")
    plt.title(Cancer.columns[i], fontsize=15)

plt.show()
```

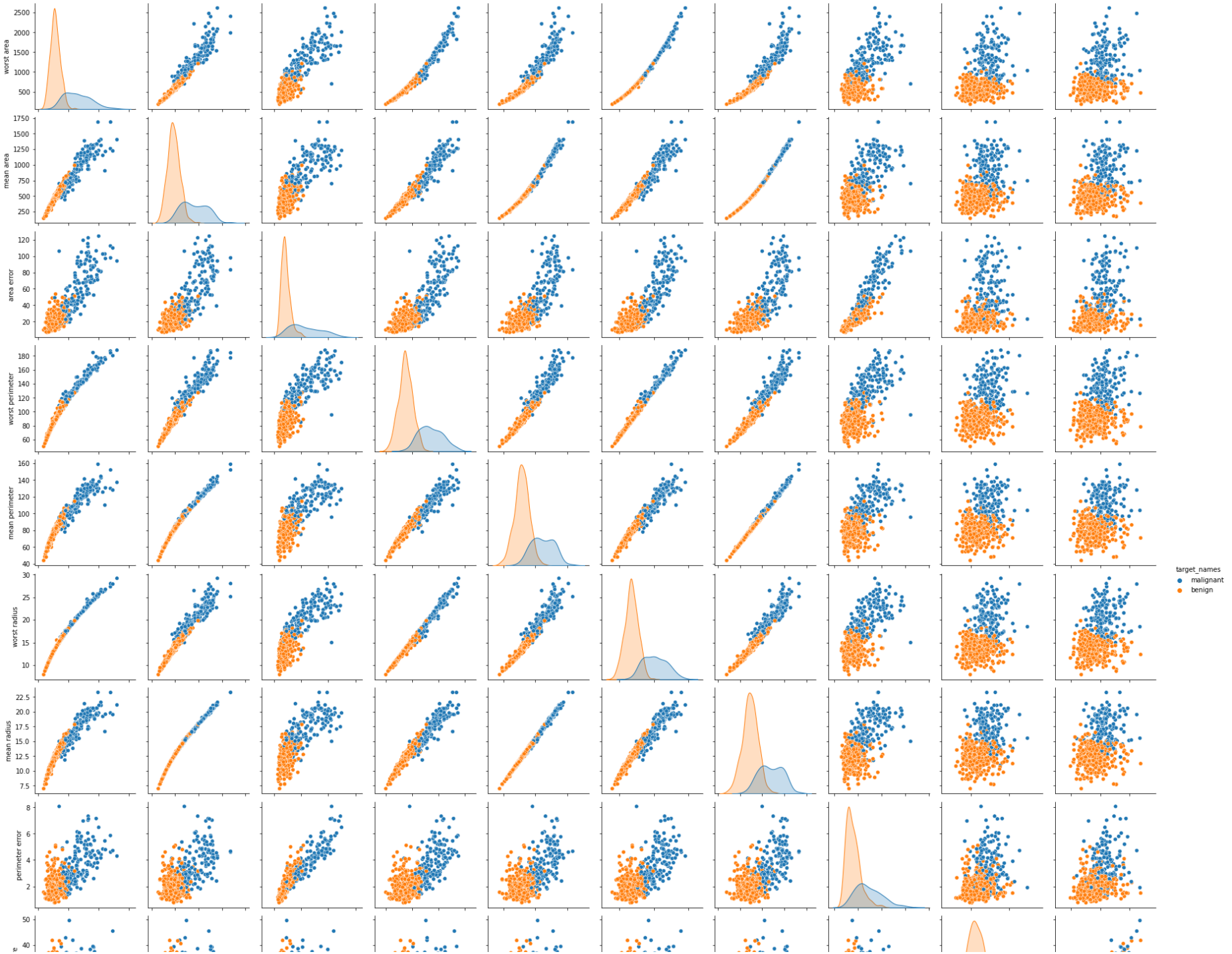

mean radius
mean texture
mean perimeter
mean area
mean smoothness
mean compactness
mean concavity
mean concave points
mean symmetry
mean fractal dimension
radius error
texture error
mean radius
target_names
malignant
benign
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0

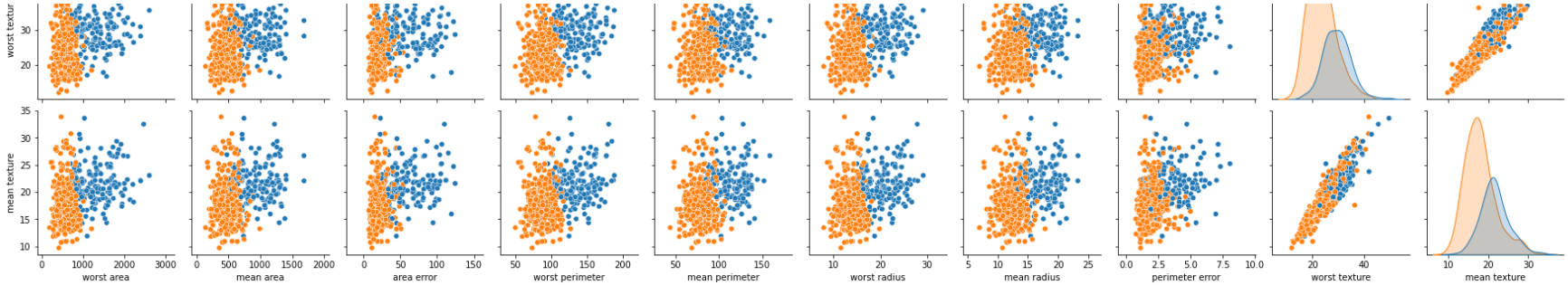

mean radius
target_names
malignant
benign
perimeter error
area error
smoothness error
compactness error
concavity error
concave points error
symmetry error
fractal dimension error
worst radius


mean radius
target_names
malignant
benign
worst texture
worst perimeter
worst area
worst smoothness
worst compactness
worst concavity
worst concave points
worst symmetry
worst fractal dimension



In [30]:
```
sns.pairplot(Cancer, vars = ['worst area', 'mean area', 'area error', 'worst perimeter',
                             'mean perimeter', 'worst radius', 'mean radius','perimeter error',
                             'worst texture', 'mean texture'],
             hue ='target_names')
plt.show()
```


worst area
mean area
area error
worst perimeter
mean perimeter
worst radius
mean radius
perimeter error
target_names
malignant
benign



In [31]:
```
Cancer['target_names'].value_counts()
```

Out[31]:
```
benign       340
malignant    173
Name: target_names, dtype: int64
```

In [32]:
```
X = Cancer.drop(columns= ['target','target_names'],axis='columns')
y = Cancer.target
```

Scaling the Data

In [33]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled
```

Out[33]:
```
array([[ 2.22607023, -0.32242245,  2.08387925, ...,  1.30686705,
        -0.21346267,  0.42289696],
       [ 1.93576969,  0.50874698,  1.94405268, ...,  2.25879191,
         1.42614976,  0.33167552],
       [ 2.13370188, -1.14164978,  2.18995458, ...,  0.9144068 ,
        -0.94682351, -0.35248535],
       ...,
       [ 0.91641894,  2.14003639,  0.89776421, ...,  0.56870777,
        -1.2242087 , -0.26253086],
       [ 2.23596684,  2.43858863,  2.43103488, ...,  2.62620151,
         2.32670167,  2.63881798],
       [-1.99978191,  1.29453646, -2.0135214 , ..., -1.79941409,
         0.01642505, -0.75728052]])
```

Seperate Data in Training & Testing

In [34]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=2)
```

In [35]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[35]:
```
((410, 30), (103, 30), (410,), (103,))
```

Apply Logistic Regression

In [36]:
```
from sklearn.linear_model import LogisticRegression
log_reg= LogisticRegression(random_state= 2)
```

In [37]:
```
log_reg.fit(X_train, y_train)
```

Out[37]:
```
LogisticRegression
LogisticRegression(random_state=2)
```

In [38]:
```
log_reg.score(X_test, y_test)
```

Out[38]:
```
0.9611650485436893
```

Accuracy = 96%

Apply SVM

In [39]:
```
from sklearn.svm import SVC
svm = SVC(random_state=2)
```

In [40]:
```
svm.fit(X_train, y_train)
```

Out[40]:
```
SVC
SVC(random_state=2)
```

In [41]:
```
svm.score(X_test, y_test)
```

Out[41]:
```
0.970873786407767
```

Accuracy = 97 %

Apply Random Forest Classifier

In [42]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(random_state=2)
```

In [43]:
```
rf.fit(X_train, y_train)
```

Out[43]:
```
RandomForestClassifier
RandomForestClassifier(random_state=2)
```

In [44]:
```
rf.score(X_test, y_test)
```

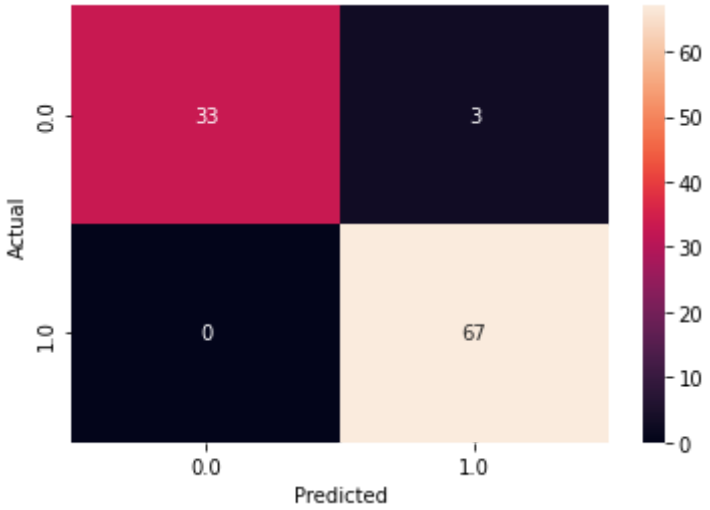
Out[44]:
```
0.970873786407767
```

In [45]:
```
y_predict= rf.predict(X_test)
```

In [46]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_predict))
```

	precision	recall	f1-score	support
0.0	1.00	0.92	0.96	36
1.0	0.96	1.00	0.98	67
accuracy			0.97	103
macro avg	0.98	0.96	0.97	103
weighted avg	0.97	0.97	0.97	103

In [47]:
```
confusion_matrix = pd.crosstab(y_test, y_predict, rownames=['Actual'], colnames=['Predicted'])

sns.heatmap(confusion_matrix, annot=True)
plt.show()
```



PCA (Principal Components Analysis)

In [48]:
```
from sklearn.decomposition import PCA
pca = PCA()
```

In [49]:
```
pca.fit(X)
```

Out[49]:
```
PCA()
```

In [50]:
```
eigenValues=pca.explained_variance_
eigenValues
```

Out[50]:
```
array([2.80062021e+05, 4.51068272e+03, 1.55400794e+02, 4.63677579e+01,
       2.87841924e+01, 2.27964488e+00, 1.54864466e+00, 1.47264573e-01,
       8.44270911e-02, 4.89952388e-02, 1.90119682e-02, 4.83729058e-03,
       2.16182248e-03, 1.56201245e-03, 5.61616341e-04, 5.00728651e-04,
       2.08916018e-04, 1.96705250e-04, 1.48300807e-04, 1.02656363e-04,
       6.26465858e-05, 4.83895402e-05, 2.32534778e-05, 1.53549853e-05,
       1.05064726e-05, 7.57040366e-06, 2.58229950e-06, 2.19509990e-06,
       9.87106163e-07, 2.33812485e-07])
```

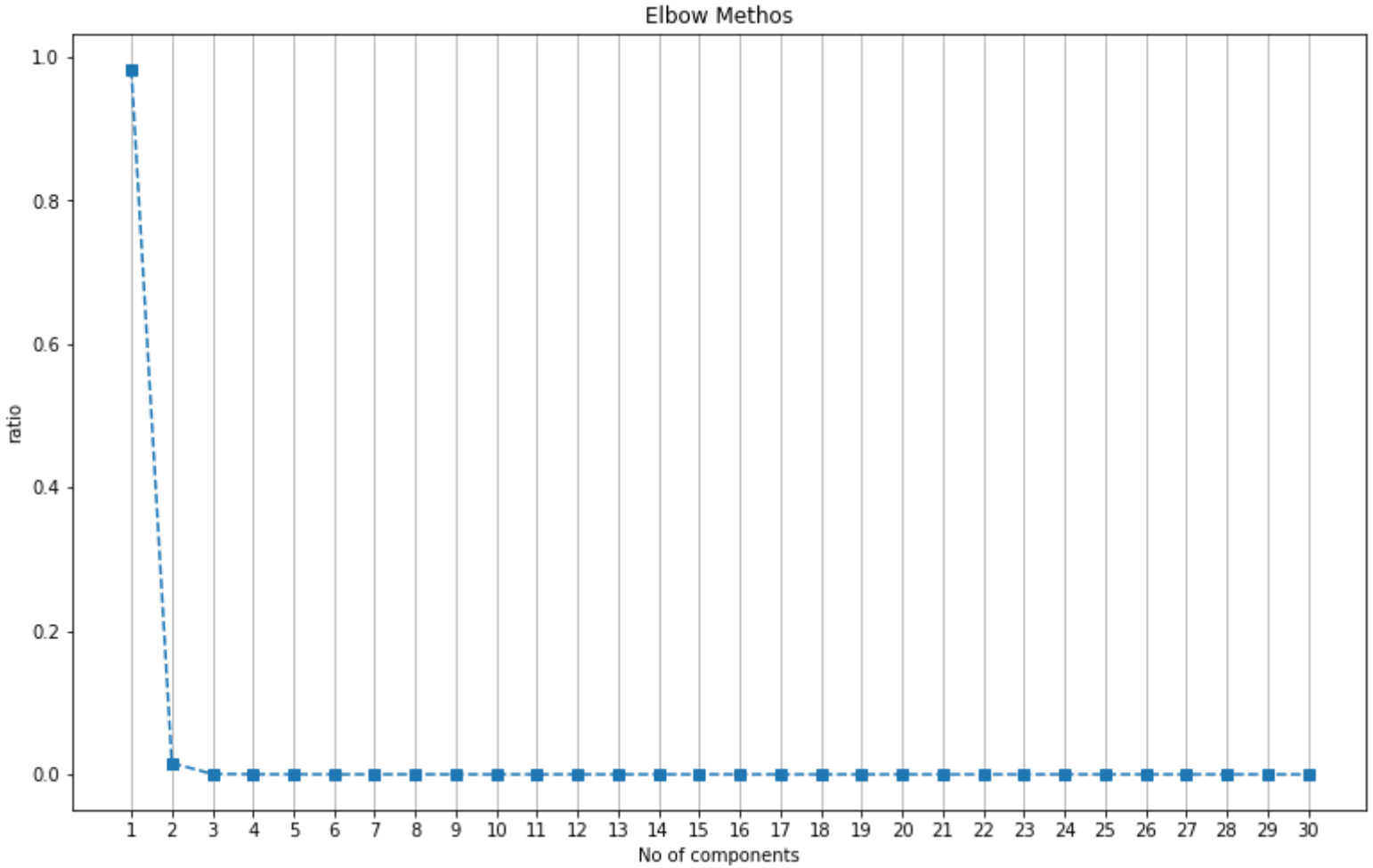
In [51]:
```
ratio= pca.explained_variance_ratio_
ratio
```

Out[51]:
```
array([9.83338305e-01, 1.58376601e-02, 5.45634689e-04, 1.62803912e-04,
       1.01065467e-04, 8.00416324e-06, 5.43751563e-06, 5.17067238e-07,
       2.96435741e-07, 1.72029377e-07, 6.67537732e-08, 1.69844276e-08,
       7.59047175e-09, 5.48445187e-09, 1.97191629e-09, 1.75813079e-09,
       7.33534387e-10, 6.90660614e-10, 5.20705606e-10, 3.60441353e-10,
       2.19961233e-10, 1.69902682e-10, 8.16463272e-11, 5.39135766e-11,
       3.68897465e-11, 2.65807833e-11, 9.06682743e-12, 7.70731358e-12,
       3.46587266e-12, 8.20949486e-13])
```

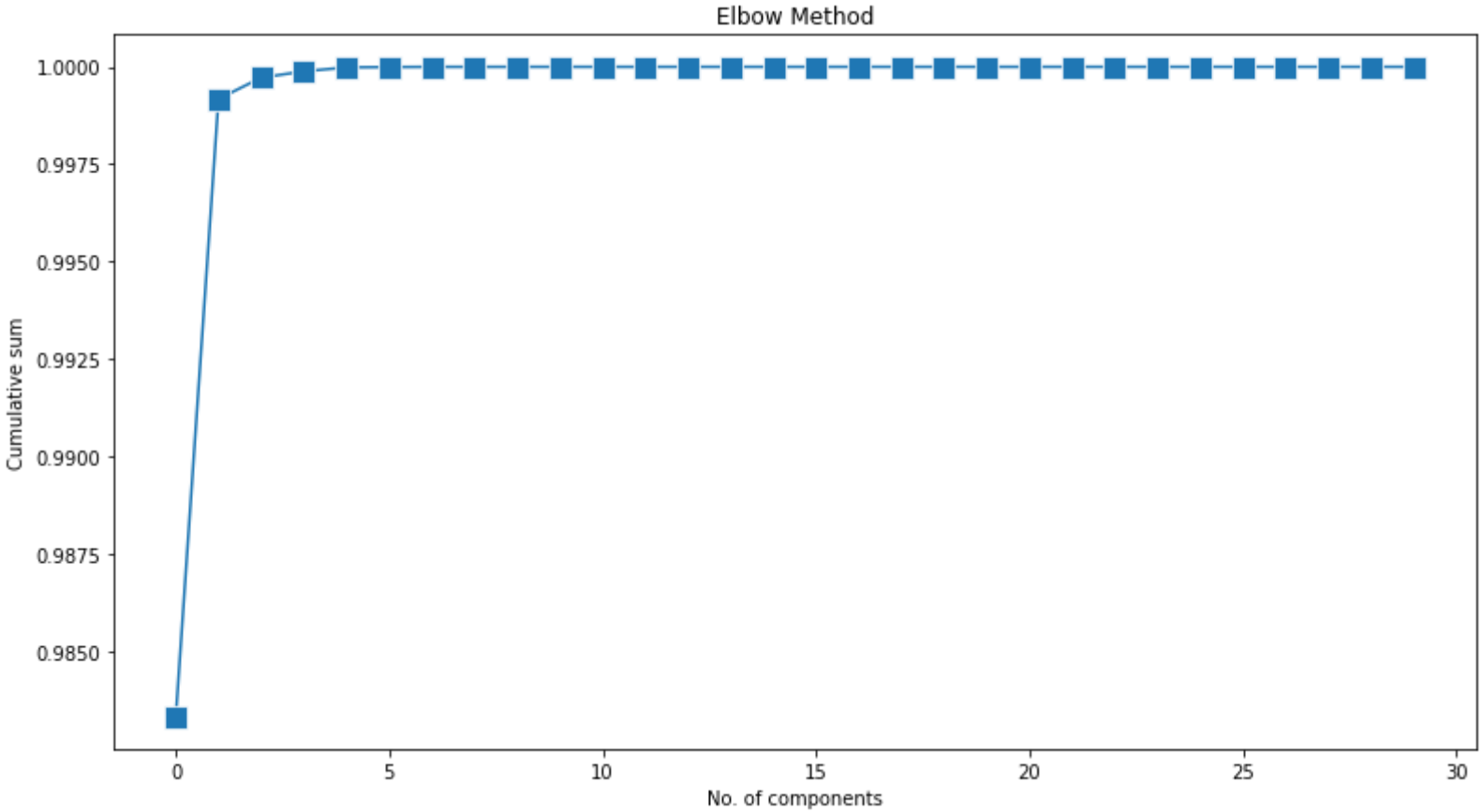
In [52]:
```
ratio_cum = np.cumsum(ratio)
ratio_cum[2]
```

Out[52]:
```
0.9997215999517084
```

In [53]:
```
#Elbow Method
plt.figure(figsize=(13,8))
plt.plot(ratio,'s--')
plt.title('Elbow Methos')
plt.xlabel('No of components')
plt.ylabel('ratio')
plt.grid(axis ='x')
plt.xticks(list(range(0,len(ratio))), list(range(1, len(ratio)+1)))
plt.show()
```



In [54]:
```
plt.figure(figsize=(13,7))
g=sns.lineplot(data=ratio_cum, marker="s", ms=12)
g.set( xlabel = "No. of components", ylabel = "Cumulative sum")
g.set_title("Elbow Method")
plt.show()
```



For 2 components

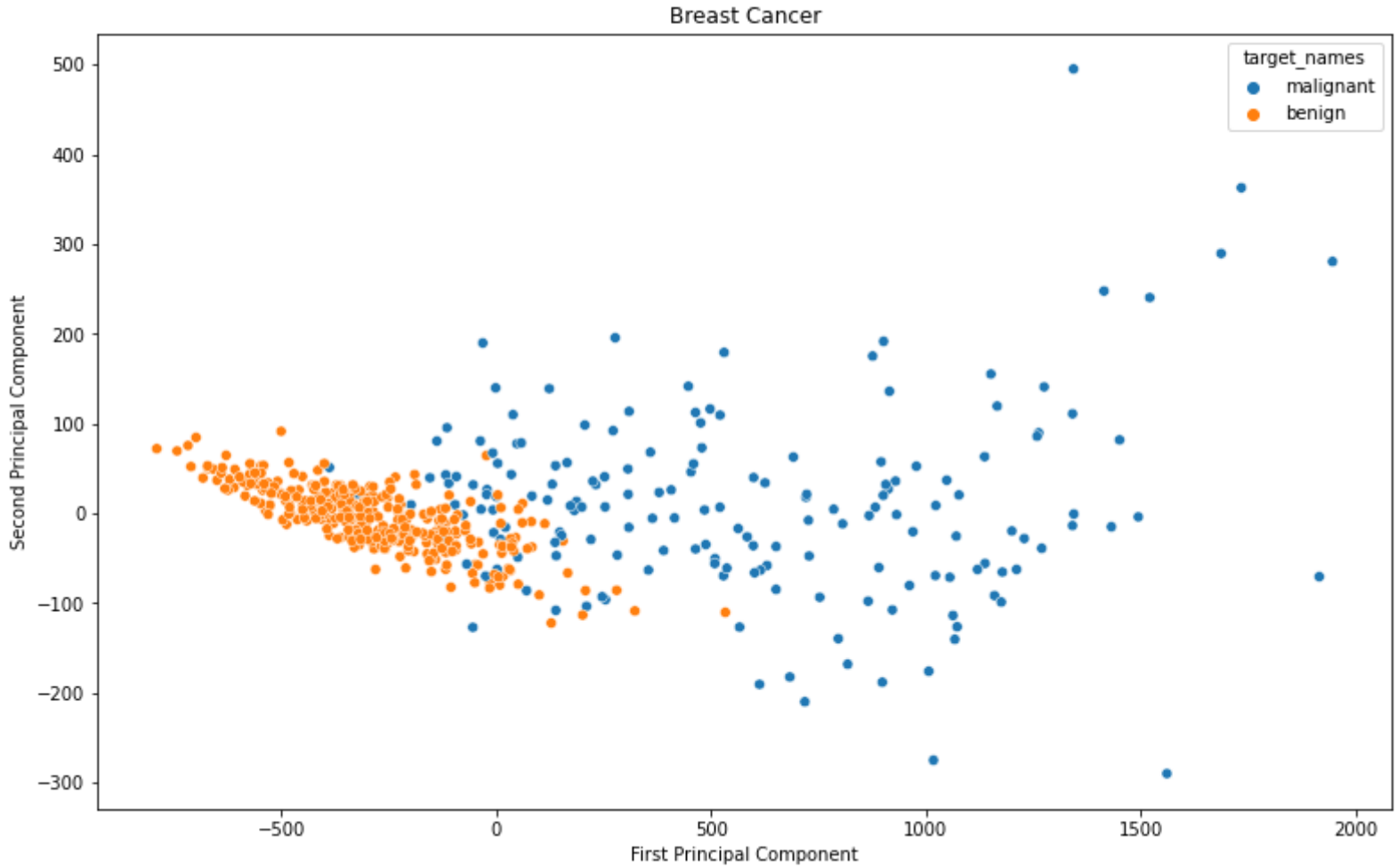
In [55]:
```
pca = PCA(n_components=2)
pca.fit(X)
```

Out[55]:
```
PCA
PCA(n_components=2)
```

In [56]:
```
X_pca = pca.transform(X)
```

In [57]:
```
plt.figure(figsize =(13, 8))
```

```
g=sns.scatterplot(data=Cancer, x=X_pca[:, 0], y=X_pca[:, 1], hue= 'target_names')
g.set( xlabel = "First Principal Component", ylabel = "Second Principal Component")
g.set_title("Breast Cancer")

plt.show()
```



In [58]:
```
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=2)
```

In [59]:
```
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier(random_state=2)
```

In [60]:
```
model_rf.fit(X_train_pca, y_train)
```

Out[60]:
```
RandomForestClassifier(random_state=2)
```

In [61]:
```
model_rf.score(X_test_pca, y_test)
```

Out[61]:
```
0.9611650485436893
```

Accuracy = 96 %

In [62]:
```
X.shape
```

Out[62]:
```
(513, 30)
```

In [63]:
```
for i in range(1,30):
    pca = PCA(n_components = i)
    X_pca = pca.fit_transform(X)
    X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=2)
    model_rf = RandomForestClassifier(random_state=2)
    model_rf.fit(X_train_pca, y_train)
    s=model_rf.score(X_test_pca, y_test)
    print(f'n :{i}, accuracy={round(s,3)}')
```

```
n :1, accuracy=0.883
n :2, accuracy=0.961
n :3, accuracy=0.961
n :4, accuracy=0.942
n :5, accuracy=0.951
n :6, accuracy=0.951
n :7, accuracy=0.961
n :8, accuracy=0.951
n :9, accuracy=0.961
n :10, accuracy=0.951
n :11, accuracy=0.951
n :12, accuracy=0.951
n :13, accuracy=0.951
n :14, accuracy=0.951
n :15, accuracy=0.951
n :16, accuracy=0.961
n :17, accuracy=0.951
n :18, accuracy=0.981
n :19, accuracy=0.961
n :20, accuracy=0.951
n :21, accuracy=0.951
n :22, accuracy=0.961
n :23, accuracy=0.961
n :24, accuracy=0.971
n :25, accuracy=0.961
n :26, accuracy=0.961
n :27, accuracy=0.971
n :28, accuracy=0.942
n :29, accuracy=0.942
```

In [64]:
```
pca = PCA(n_components = 18)
pca.fit(X)
```

Out[64]:
```
PCA
PCA(n_components=18)
```

In [65]:
```
X_pca = pca.transform(X)
```

In [66]:
```
X_train_pca, X_test_pca, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=2)
```

In [67]:
```
model_rf = RandomForestClassifier(random_state=2)
```

```
model_rf.fit(X_train_pca, y_train)
```

Out[67]:

RandomForestClassifier

RandomForestClassifier(random_state=2)

In [68]:
```
model_rf.score(X_test_pca, y_test)
```

Out[68]:
```
0.9805825242718447
```

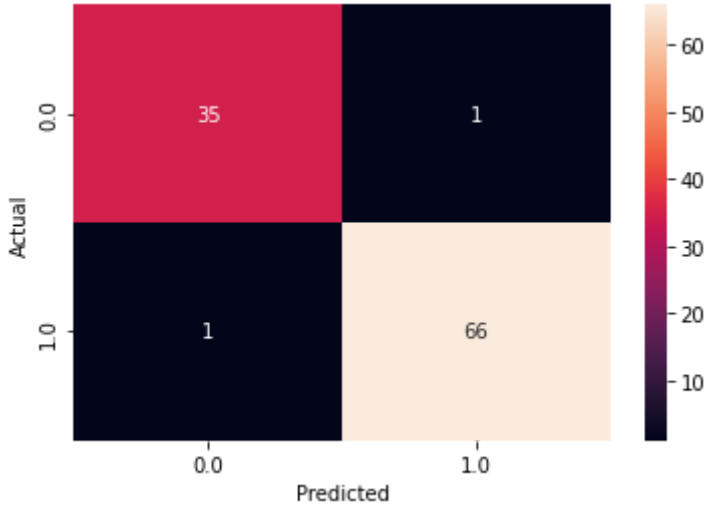
In [69]:
```
y_predict_pca= model_rf.predict(X_test_pca)
y_predict_pca
```

Out[69]:
```
array([0., 1., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 1., 0., 0.,
       1., 1., 1., 0., 0., 1., 1., 0., 0., 1., 1., 1., 1., 1., 1., 1., 0.,
       1., 1., 1., 1., 1., 1., 1., 0., 0., 1., 1., 0., 0., 1., 1., 1., 0.,
       1., 1., 1., 1., 0., 0., 0., 0., 1., 1., 1., 0., 1., 0., 1., 1., 1.,
       0., 1., 1., 0., 0., 0., 1., 0., 1., 0., 1., 0., 1., 1., 1., 0., 0.,
       1., 1., 1., 1., 0., 1., 0., 1., 1., 1., 1., 0., 0., 0., 1., 0., 1.,
       1.])
```

In [70]:
```
print(classification_report(y_test, y_predict_pca))
```

```
              precision    recall  f1-score   support

         0.0       0.97      0.97      0.97        36
         1.0       0.99      0.99      0.99        67

    accuracy                           0.98       103
   macro avg       0.98      0.98      0.98       103
weighted avg       0.98      0.98      0.98       103
```

In [71]:
```
confusion_matrix = pd.crosstab(y_test, y_predict_pca, rownames=['Actual'], colnames=['Predicted'])

sns.heatmap(confusion_matrix, annot=True)
plt.show()
```



Accuracy = 98%