Breast_Cancer_Detection

June 5, 2019

1 Breast Cancer Detection

1.1 Importing libraries and dataset

In [1]:
```
#Importing libraries

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

In [2]:
```
#Loading dataset

from sklearn.datasets import load_breast_cancer
```

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

In [4]:
```
cancer.keys()
```

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

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

```
.. _breast_cancer_dataset:

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

**Data Set Characteristics:**

    :Number of Instances: 569

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

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
```

- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

- class:
    - WDBC-Malignant
    - WDBC-Benign

:Summary Statistics:

	Min	Max
radius (mean):	6.981	28.11
texture (mean):	9.71	39.28
perimeter (mean):	43.79	188.5
area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54
perimeter (worst):	50.41	251.2
area (worst):	185.2	4254.0
smoothness (worst):	0.071	0.223
compactness (worst):	0.027	1.058
concavity (worst):	0.0	1.252

```
concave points (worst):             0.0    0.291
symmetry (worst):                   0.156  0.664
fractal dimension (worst):          0.055  0.208
===================================== ====== ======
```

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

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

:Donor: Nick Street

:Date: November, 1995

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

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

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

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

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

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

.. topic:: References

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

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

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

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

In [7]:
```
print(cancer['target'])
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 0 1 0 0
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1 1 1 0 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0 0 0 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 1 1
 1 1 0 1 0 1 0 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0
 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1 0 1 0 1 0 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

In [8]:
```
cancer['data'].shape
```

Out[8]: (569, 30)

In [9]:
```
#Save this in a dataframe for further analysis

df_cancer=pd.DataFrame(np.c_[cancer['data'],cancer['target']],columns=np.append(cancer
```

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

Out[10]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

```
   mean compactness  mean concavity  mean concave points  mean symmetry  \
0           0.27760          0.3001              0.14710         0.2419
1           0.07864          0.0869              0.07017         0.1812
2           0.15990          0.1974              0.12790         0.2069
3           0.28390          0.2414              0.10520         0.2597
4           0.13280          0.1980              0.10430         0.1809

   mean fractal dimension  ...  worst texture  worst perimeter  worst area  \
0                 0.07871  ...          17.33           184.60      2019.0
1                 0.05667  ...          23.41           158.80      1956.0
2                 0.05999  ...          25.53           152.50      1709.0
3                 0.09744  ...          26.50            98.87       567.7
4                 0.05883  ...          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   worst symmetry  worst fractal dimension  target
0          0.4601                  0.11890     0.0
1          0.2750                  0.08902     0.0
2          0.3613                  0.08758     0.0
3          0.6638                  0.17300     0.0
4          0.2364                  0.07678     0.0

[5 rows x 31 columns]
```

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

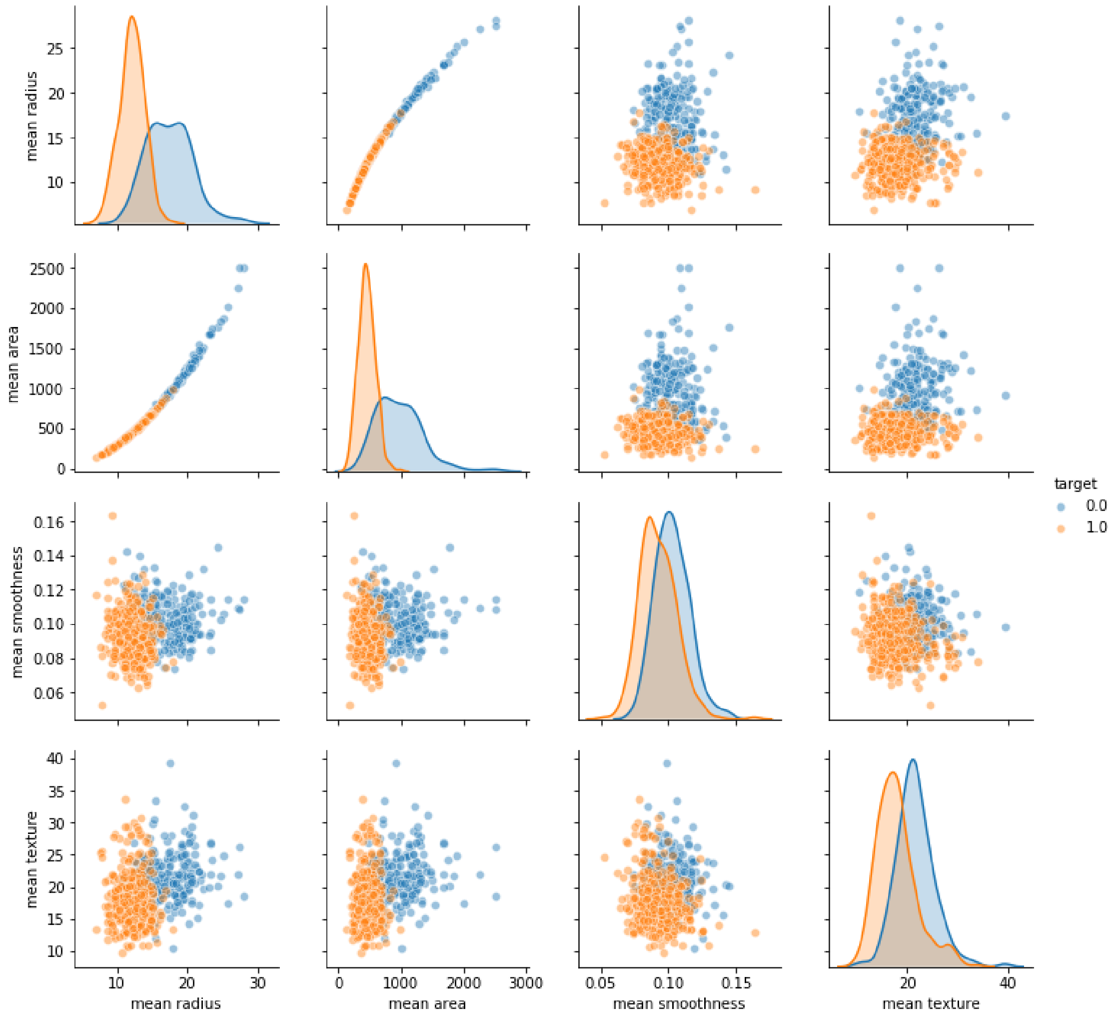
```
Out[11]:      mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
         564        21.56         22.39          142.00     1479.0          0.11100
         565        20.13         28.25          131.20     1261.0          0.09780
         566        16.60         28.08          108.30      858.1          0.08455
         567        20.60         29.33          140.10     1265.0          0.11780
         568         7.76         24.54           47.92      181.0          0.05263

              mean compactness  mean concavity  mean concave points  mean symmetry  \
         564           0.11590         0.24390              0.13890         0.1726
         565           0.10340         0.14400              0.09791         0.1752
         566           0.10230         0.09251              0.05302         0.1590
         567           0.27700         0.35140              0.15200         0.2397
         568           0.04362         0.00000              0.00000         0.1587

              mean fractal dimension  ...  worst texture  worst perimeter  worst area  \
```

```
564             0.05623  ...         26.40         166.10      2027.0
565             0.05533  ...         38.25         155.00      1731.0
566             0.05648  ...         34.12         126.70      1124.0
567             0.07016  ...         39.42         184.60      1821.0
568             0.05884  ...         30.37          59.16       268.6

     worst smoothness  worst compactness  worst concavity  \
564           0.14100            0.21130           0.4107
565           0.11660            0.19220           0.3215
566           0.11390            0.30940           0.3403
567           0.16500            0.86810           0.9387
568           0.08996            0.06444           0.0000

     worst concave points  worst symmetry  worst fractal dimension  target
564                0.2216          0.2060                  0.07115     0.0
565                0.1628          0.2572                  0.06637     0.0
566                0.1418          0.2218                  0.07820     0.0
567                0.2650          0.4087                  0.12400     0.0
568                0.0000          0.2871                  0.07039     1.0

[5 rows x 31 columns]
```

1.2 Visualizing the Data

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

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x7f67238cfa58>
```

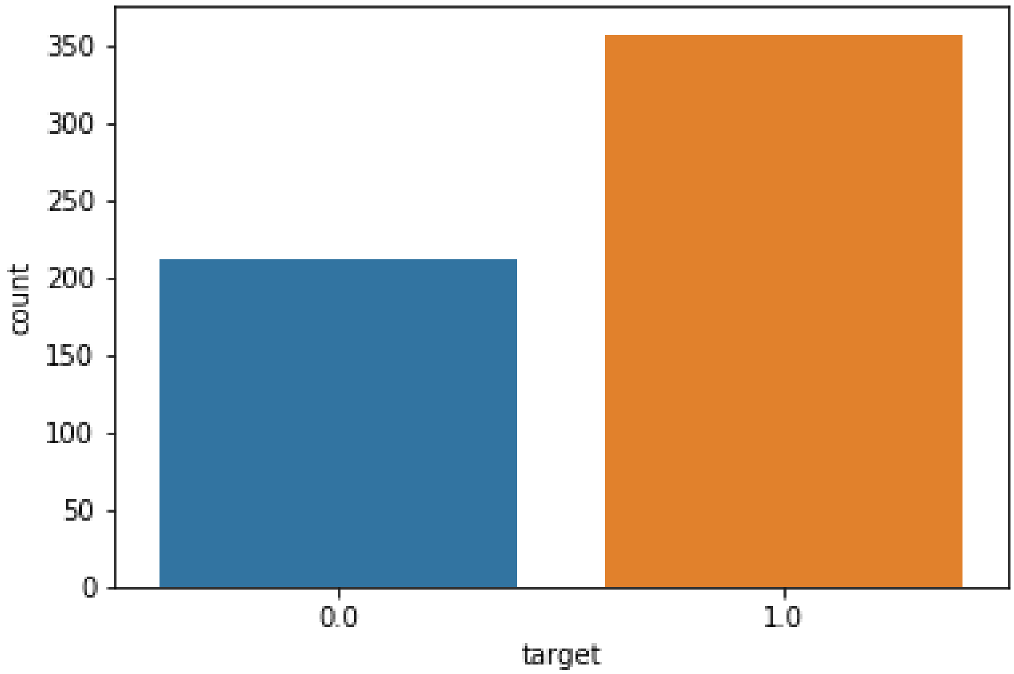


1.2.1 Insights

1. Mean radius of benign(1) tumours is lesser than that of cancerous tumours.
2. Area follows the same trend since area is proportional to radius.
3. Mean smoothness/texture vs mean radius/area also show linearly separable regions.

```
In [13]: sns.countplot(df_cancer['target'],label='count')
```

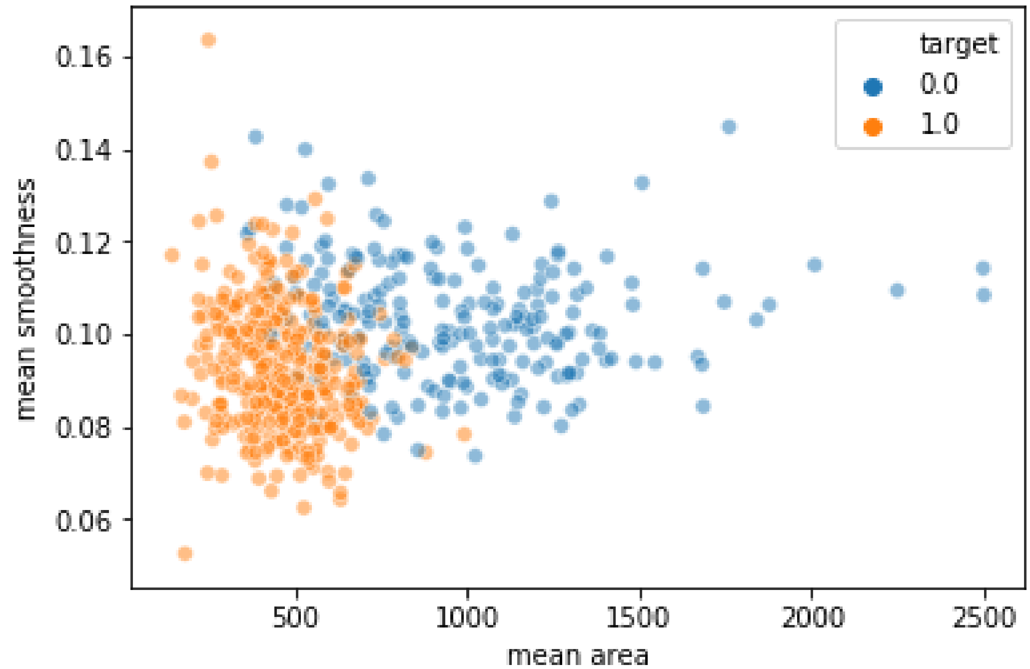
```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6722ebd4e0>
```



1.2.2 Insights

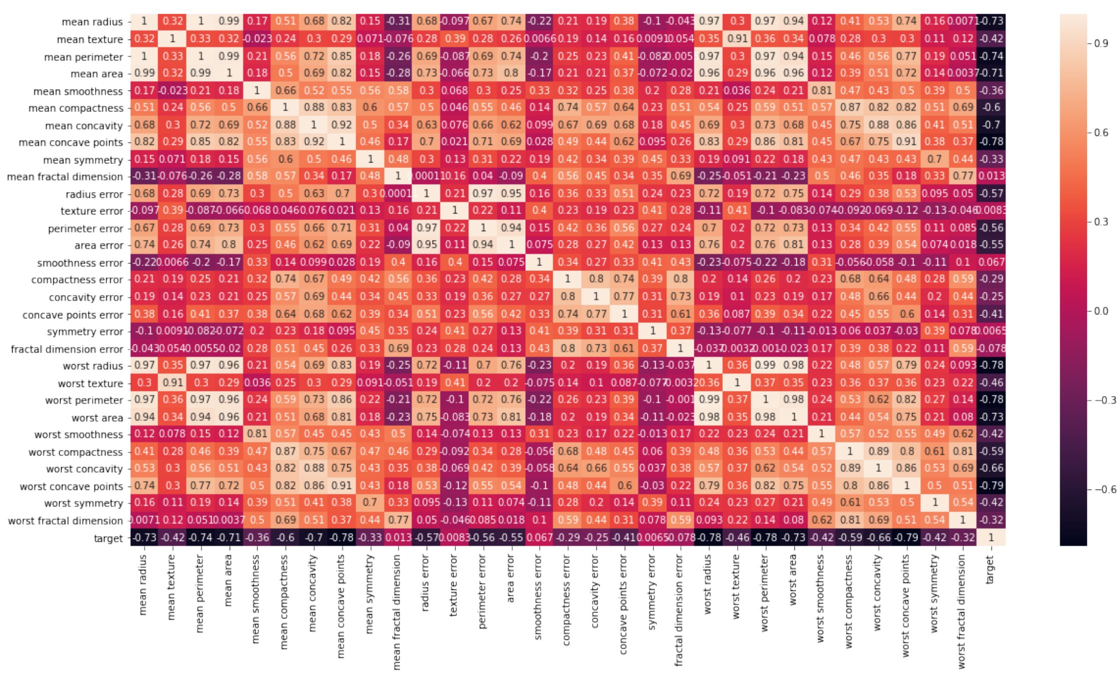
1. Benign tumours outnumber cancerous tumours.
2. Nearly double.

```
In [14]: df_cancer['target'].value_counts()

Out[14]: 1.0    357
         0.0    212
         Name: target, dtype: int64

In [15]: sns.scatterplot(x='mean area',y='mean smoothness',hue='target',alpha=0.50,data=df_canc

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f672110f4a8>
```

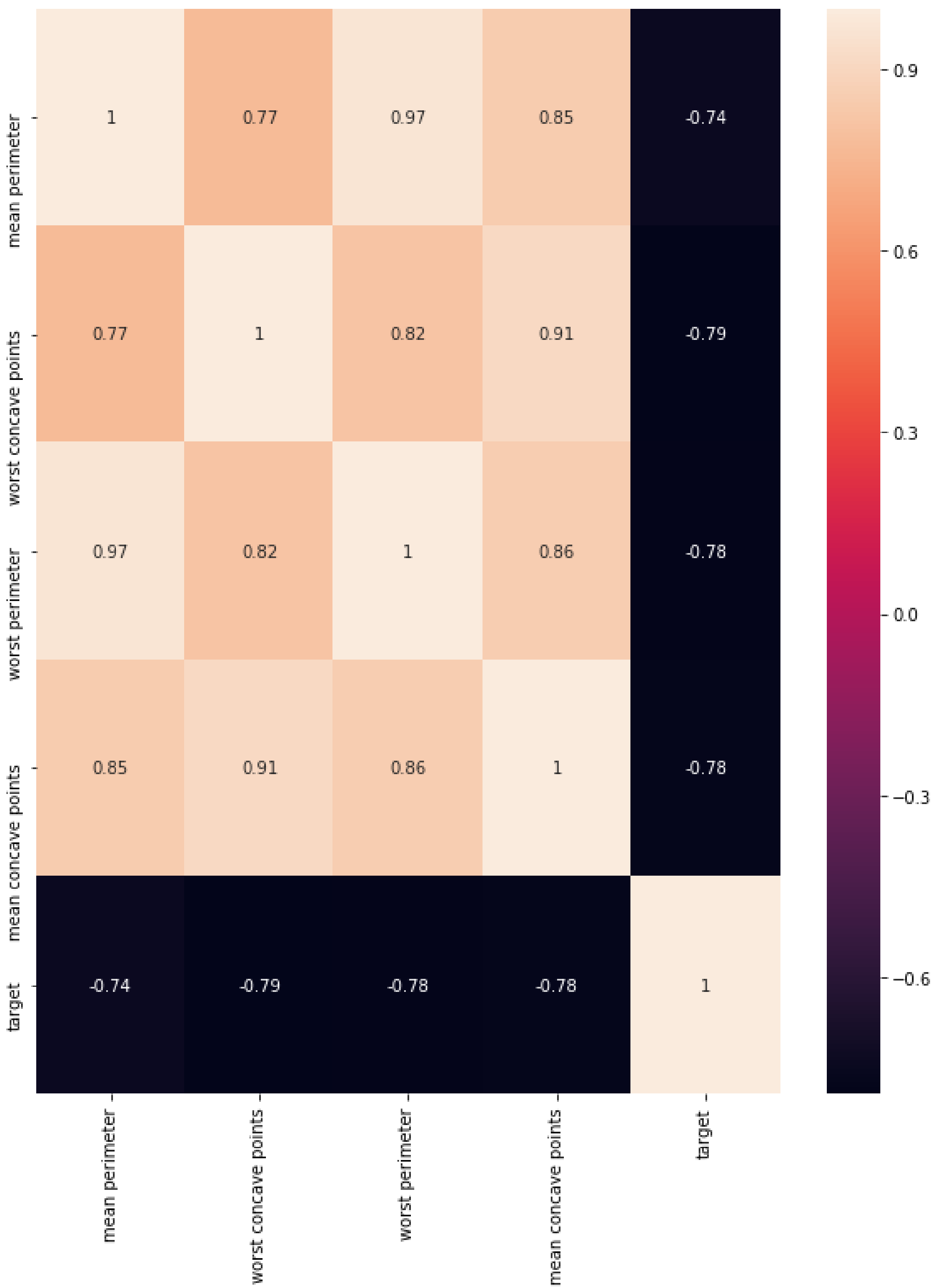



```
In [16]: plt.figure(figsize=(20,10))
         sns.heatmap(df_cancer.corr(), annot=True)
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f672105e7b8>
```



In [17]:
```
fig = plt.figure(figsize=(10,12))
fig = sns.heatmap(df_cancer[['mean perimeter','worst concave points','worst perimeter
```



1.2.3 Insights

The features in second correlation matrix are the ones that have high correlations with target. Note: Correlation not necessarily != Causation

1.3 Model Building

In [18]:
```
X=df_cancer.iloc[:,:-1]
print(X.columns)
print(X.shape)
```

```
Index(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error', 'fractal dimension error',
       'worst radius', 'worst texture', 'worst perimeter', 'worst area',
       'worst smoothness', 'worst compactness', 'worst concavity',
       'worst concave points', 'worst symmetry', 'worst fractal dimension'],
      dtype='object')
(569, 30)
```

In [19]:
```
y=df_cancer['target']
print(y.shape)
```

```
(569,)
```

In [20]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
```

In [21]:
```
X_train.head()
```

Out[21]:
```
     mean radius  mean texture  mean perimeter  mean area  mean smoothness  \
338        10.05         17.53           64.41      310.8          0.10070
427        10.80         21.98           68.79      359.9          0.08801
406        16.14         14.86          104.30      800.0          0.09495
96         12.18         17.84           77.79      451.1          0.10450
490        12.25         22.44           78.18      466.5          0.08192

     mean compactness  mean concavity  mean concave points  mean symmetry  \
338           0.07326         0.02511              0.01775         0.1890
427           0.05743         0.03614              0.01404         0.2016
406           0.08501         0.05500              0.04528         0.1735
```

```
96          0.07057        0.02490             0.02941        0.1900
490         0.05200        0.01714             0.01261        0.1544

     mean fractal dimension  ...  worst radius  worst texture  \
338                 0.06331  ...         11.16          26.84
427                 0.05977  ...         12.76          32.04
406                 0.05875  ...         17.71          19.58
96                  0.06635  ...         12.83          20.92
490                 0.05976  ...         14.17          31.99

     worst perimeter  worst area  worst smoothness  worst compactness  \
338            71.98       384.0            0.1402            0.14020
427            83.69       489.5            0.1303            0.16960
406           115.90       947.9            0.1206            0.17220
96             82.14       495.2            0.1140            0.09358
490            92.74       622.9            0.1256            0.18040

     worst concavity  worst concave points  worst symmetry  \
338           0.1055               0.06499          0.2894
427           0.1927               0.07485          0.2965
406           0.2310               0.11290          0.2778
96            0.0498               0.05882          0.2227
490           0.1230               0.06335          0.3100

     worst fractal dimension
338                  0.07664
427                  0.07662
406                  0.07012
96                   0.07376
490                  0.08203

[5 rows x 30 columns]
```

The model we are going to use is SVM. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial ker- nel, large attribute values might cause numerical problems. Recommended: linearly scaling each attribute to the range [-1,+1] or [0,1].

Here MinMaxScaler is used. Output range is [0,1]. Outliers are not affected. Minimum distortion or loss of information.

In [22]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

In [23]: pd.DataFrame(X_train).head()

Out[23]:

```
          0         1         2         3         4         5         6   \
0  0.145251  0.324481  0.142492  0.070965  0.522103  0.184508  0.058833
1  0.180747  0.509129  0.172759  0.091792  0.384273  0.130299  0.084677
2  0.433480  0.213693  0.418147  0.278473  0.459650  0.224745  0.128866
3  0.246060  0.337344  0.234953  0.130477  0.563376  0.175296  0.058341
4  0.249373  0.528216  0.237648  0.137010  0.318128  0.111705  0.040159

          7         8         9   ...        20        21        22        23  \
0  0.088221  0.419192  0.281171  ...  0.114906  0.394989  0.107426  0.048860
1  0.069781  0.482828  0.206613  ...  0.171825  0.533582  0.165745  0.074789
2  0.225050  0.340909  0.185131  ...  0.347919  0.201493  0.326162  0.187451
3  0.146173  0.424242  0.345198  ...  0.174315  0.237207  0.158026  0.076190
4  0.062674  0.244444  0.206403  ...  0.221985  0.532249  0.210817  0.107575

         24        25        26        27        28        29
0  0.455854  0.109546  0.084265  0.223872  0.261975  0.141677
1  0.390477  0.138070  0.153914  0.257837  0.275971  0.141545
2  0.326421  0.140592  0.184505  0.388908  0.239109  0.098911
3  0.282837  0.064315  0.039776  0.202618  0.130495  0.122786
4  0.359440  0.148548  0.098243  0.218223  0.302582  0.177030

[5 rows x 30 columns]
```

In [24]:

```python
#Building a SUPPORT VECTOR MACHINE classifier

from sklearn.svm import SVC
classifier=SVC(kernel='linear',random_state=0)
```

In [25]:

```python
classifier.fit(X_train,y_train)
```

Out[25]:

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=0,
  shrinking=True, tol=0.001, verbose=False)
```

In [26]:

```python
y_pred=classifier.predict(X_test)
```

In [27]:

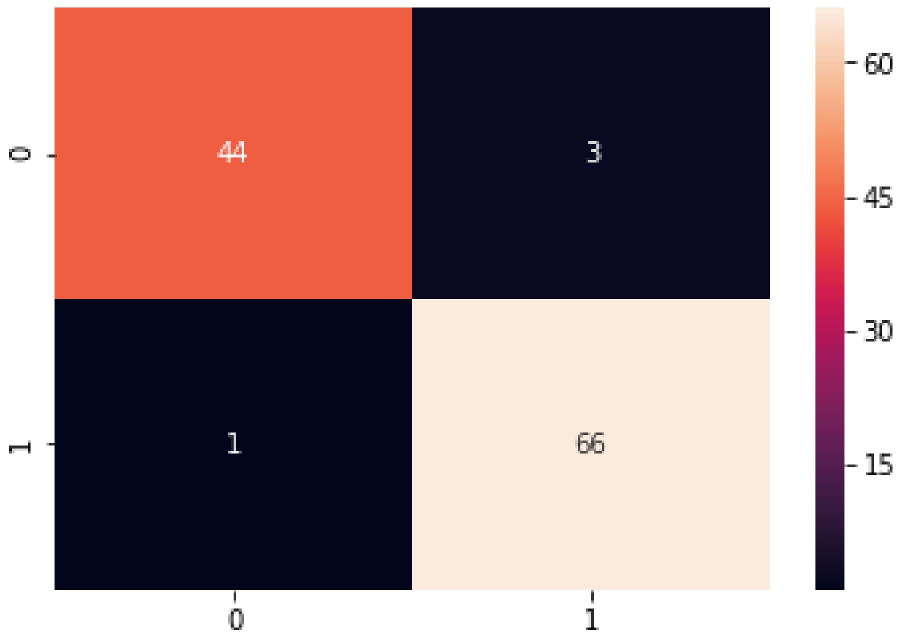
```python
from sklearn.metrics import confusion_matrix,classification_report
cm=confusion_matrix(y_test,y_pred)
report=classification_report(y_test,y_pred)
```

In [28]:

```python
sns.heatmap(cm,annot=True)
```

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f672035d2e8>
```




```
In [29]: print(report)
```

```
              precision    recall  f1-score   support

         0.0       0.98      0.94      0.96        47
         1.0       0.96      0.99      0.97        67

   micro avg       0.96      0.96      0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

Cross Validation

```
In [30]: from sklearn.model_selection import cross_val_score
         accuracies=cross_val_score(estimator=classifier,X=X_train,y=y_train,cv=10,n_jobs=-1,s
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    2.7s finished
```

```
In [31]: print('Mean accuracy is {:.2f} and standard deviation is {:.2f}'.format(accuracies.mea
```

```
Mean accuracy is 0.97 and standard deviation is 0.02
```

```
In [32]: from sklearn.model_selection import GridSearchCV
         parameters=[{'C':[0.1,10,100,1000],'kernel':['linear']},{'C':[0.1,10,100,1000],'kerne
         grid=GridSearchCV(estimator=SVC(),param_grid=parameters,scoring='accuracy',cv=10,n_jol
```

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

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.

Fitting 10 folds for each of 20 candidates, totalling 200 fits

[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:    0.8s finished
```

```
Out[33]: GridSearchCV(cv=10, error_score='raise-deprecating',
                estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
           kernel='rbf', max_iter=-1, probability=False, random_state=None,
           shrinking=True, tol=0.001, verbose=False),
                fit_params=None, iid=True, n_jobs=-1,
                param_grid=[{'C': [0.1, 10, 100, 1000], 'kernel': ['linear']}, {'C': [0.1, 10,
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring='accuracy', verbose=True)
```

```
In [34]: grid.best_estimator_
```

```
Out[34]: SVC(C=10, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma=1, kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)
```

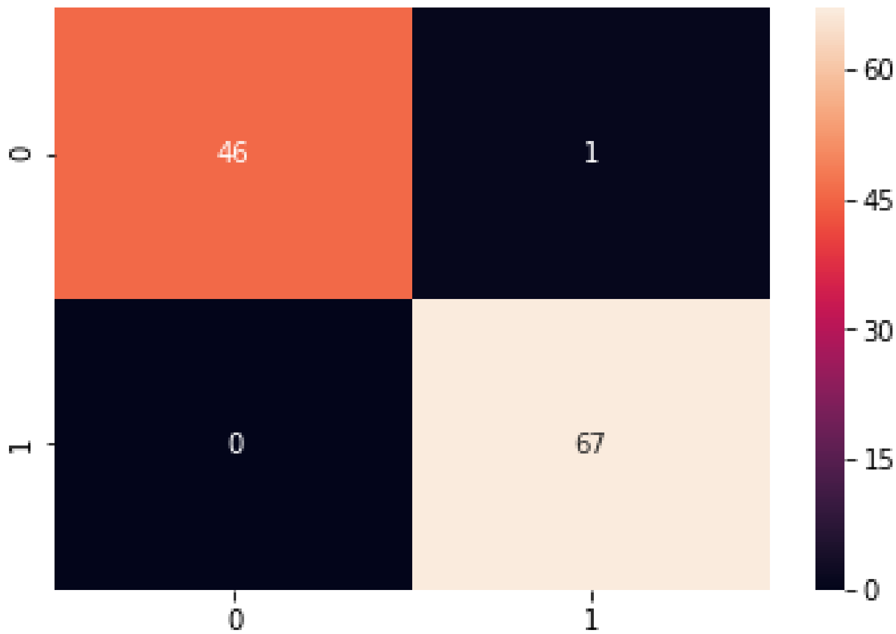
```
In [35]: grid.best_params_
```

```
Out[35]: {'C': 10, 'gamma': 1, 'kernel': 'rbf'}
```

```
In [36]: best_estimator=grid.best_estimator_
         y_pred_tuned=best_estimator.predict(X_test)
```

```
In [37]: cm_tuned=confusion_matrix(y_test,y_pred_tuned)
         sns.heatmap(cm_tuned,annot=True)
```

```
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x7f671905bf60>
```




```
In [38]: print(classification_report(y_test,y_pred_tuned))

              precision    recall  f1-score   support

         0.0       1.00      0.98      0.99        47
         1.0       0.99      1.00      0.99        67

   micro avg       0.99      0.99      0.99       114
   macro avg       0.99      0.99      0.99       114
weighted avg       0.99      0.99      0.99       114
```

```
In [39]: from sklearn.model_selection import cross_val_score
         accuracies=cross_val_score(estimator=best_estimator,X=X_train,y=y_train,cv=10,n_jobs=
         print('Mean accuracy is {:.2f} and standard deviation is {:.2f}'.format(accuracies.mea

Mean accuracy is 0.98 and standard deviation is 0.02


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done   3 out of  10 | elapsed:    0.0s remaining:    0.1s
[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:    0.0s finished
```