Breast Cancer Detection using Machine Learning

Objectives:

The goal is to classify whether the breast cancer is benign or malignant based on several observations/features

Real-valued features are:

- 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)

Target class:

- Malignant
- Benign

Phase 1 — Data Exploration

In [15]:
```
# import libraries
import pandas as pd # Import Pandas for data manipulation using datafra
mes
import numpy as np # Import Numpy for data statistical analysis
import matplotlib.pyplot as plt # Import matplotlib for data visualisat
ion
import seaborn as sns # Statistical data visualization
```

Importing our cancer dataset

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

In [3]:
```
cancer.keys()
```

Out[3]:
```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names'])
```

In [4]:
```
cancer['data'].shape
```

Out[4]:
```
(569, 30)
```

In [5]:
```
cancer['target']
```

Out[5]:
```
array([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, 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, 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, 1, 0, 0, 0, 0, 0, 0, 1])
```

In [6]:
```
cancer['target_names']
```

Out[6]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [7]:
```
cancer['DESCR']
```

Out[7]:
```
'Breast Cancer Wisconsin (Diagnostic) Database\n==========================\n\nNotes\n-----\nData Set Characteristics:\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 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        largest values) of these features were computed for each image,\n        resulting in 30 features.  For instance, field 3 is Mean Radius, field\n        13 is Radius SE, field 23 is Worst Radius.\n\n        - class:\n                - WDBC-Malignant\n                - WDBC-Benign\n\n    :Summary Statistics:\n\n    ============================== ====== ======\n                                   Min    Max\n    ============================== ====== ======\n    radius (mean):                 6.981  28.11\n    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 (wors
```

```
t):                          7.93   36.04\n    texture (worst):
          12.02  49.54\n    perimeter (worst):                   50.4
1  251.2\n    area (worst):                          185.2  4254.0\n
smoothness (worst):                   0.071  0.223\n    compactness (wo
rst):                  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    frac
tal dimension (worst):            0.055  0.208\n    ==================
================== ====== ======\n\n    :Missing Attribute Values: None
\n\n    :Class Distribution: 212 - Malignant, 357 - Benign\n\n    :Crea
tor:  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/U
2Uwz2\n\nFeatures are computed from a digitized image of a fine needle
\naspirate (FNA) of a breast mass.  They describe\ncharacteristics of t
he cell nuclei present in the image.\n\nSeparating plane described abov
e was obtained using\nMultisurface Method-Tree (MSM-T) [K. P. Bennett,
"Decision Tree\nConstruction Via Linear Programming." Proceedings of th
e 4th\nMidwest Artificial Intelligence and Cognitive Science Society,\n
pp. 97-101, 1992], a classification method which uses linear\nprogrammi
ng to construct a decision tree.  Relevant features\nwere selected usin
g 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 plan
e\nin the 3-dimensional space is that described in:\n[K. P. Bennett and
O. L. Mangasarian: "Robust Linear\nProgramming Discrimination of Two Li
nearly Inseparable Sets",\nOptimization Methods and Software 1, 1992, 2
3-34].\n\nThis database is also available through the UW CS ftp serve
r:\n\nftp ftp.cs.wisc.edu\ncd math-prog/cpo-dataset/machine-learn/WDBC/
\n\nReferences\n----------\n   - W.N. Street, W.H. Wolberg and O.L. Man
gasarian. Nuclear feature extraction \n     for breast tumor diagnosis.
IS&T/SPIE 1993 International Symposium on \n     Electronic Imaging: Sc
ience and Technology, volume 1905, pages 861-870,\n     San Jose, CA, 1
993.\n   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cance
r diagnosis and \n     prognosis via linear programming. Operations Res
earch, 43(4), pages 570-577, \n     July-August 1995.\n   - W.H. Wolber
g, 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.\n'
```

In [8]:
```
cancer['feature_names']
```

Out[8]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimensio
n',
       'radius error', 'texture error', 'perimeter error', 'area erro
r',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [17]:
```
df_cancer=pd.DataFrame(np.c_[cancer['data'],cancer['target']],columns=n
p.append(cancer['feature_names'],['target']))
df_cancer.head()
```

Out[17]:

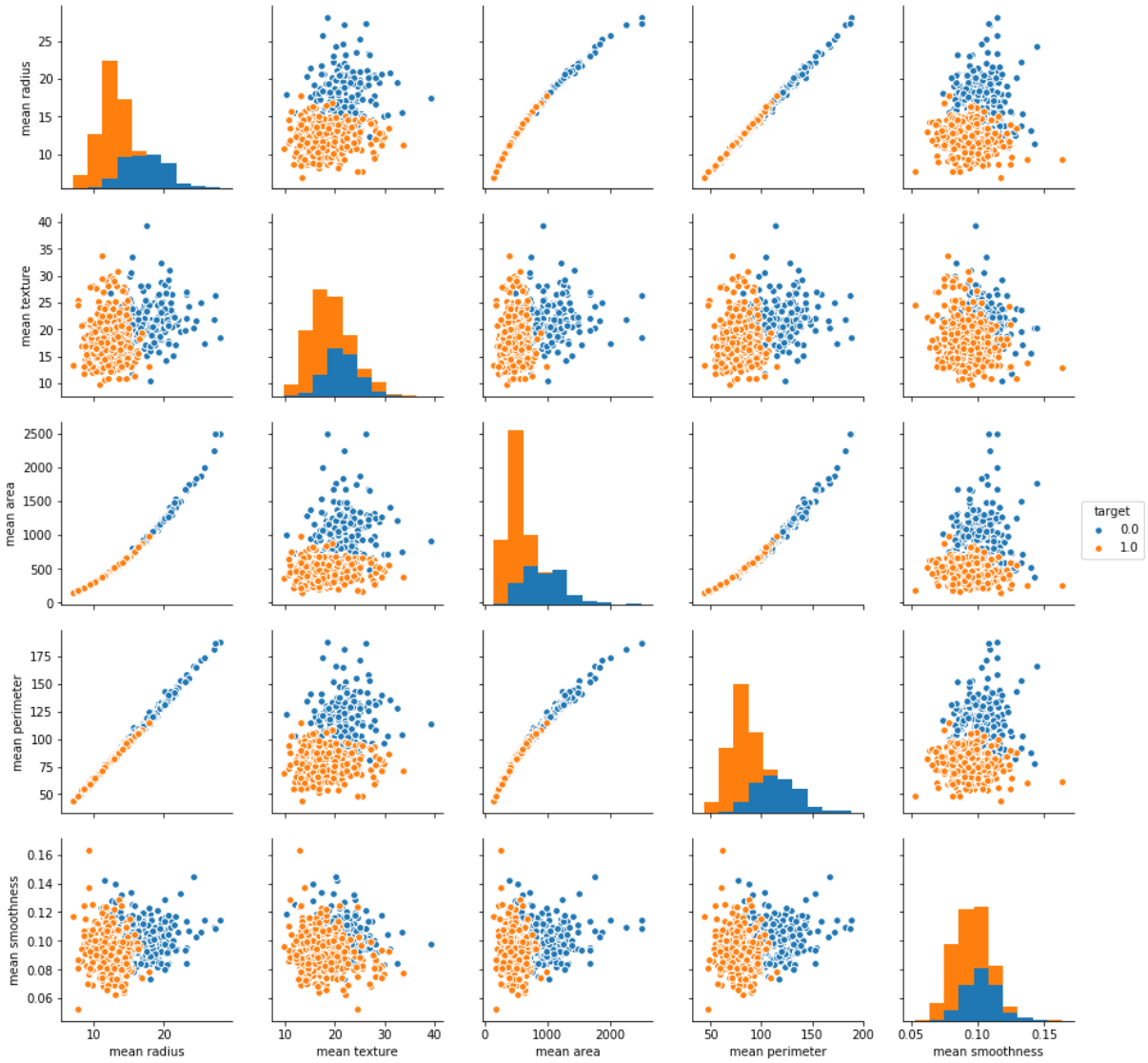
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	syn
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.24
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.18
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.20
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.25
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.18

5 rows × 31 columns

Phase 2- Data Visualization:

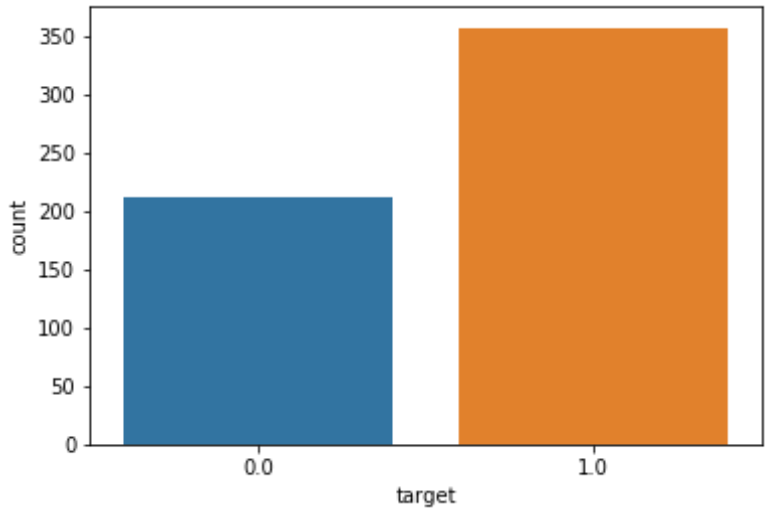
In [18]:
```
sns.pairplot(df_cancer, hue = 'target', vars = ['mean radius', 'mean texture', 'mean area', 'mean perimeter', 'mean smoothness'] )
```

Out[18]: <seaborn.axisgrid.PairGrid at 0x187e690a438>



In [20]:
```
sns.countplot(df_cancer['target'], label = "Count")
```

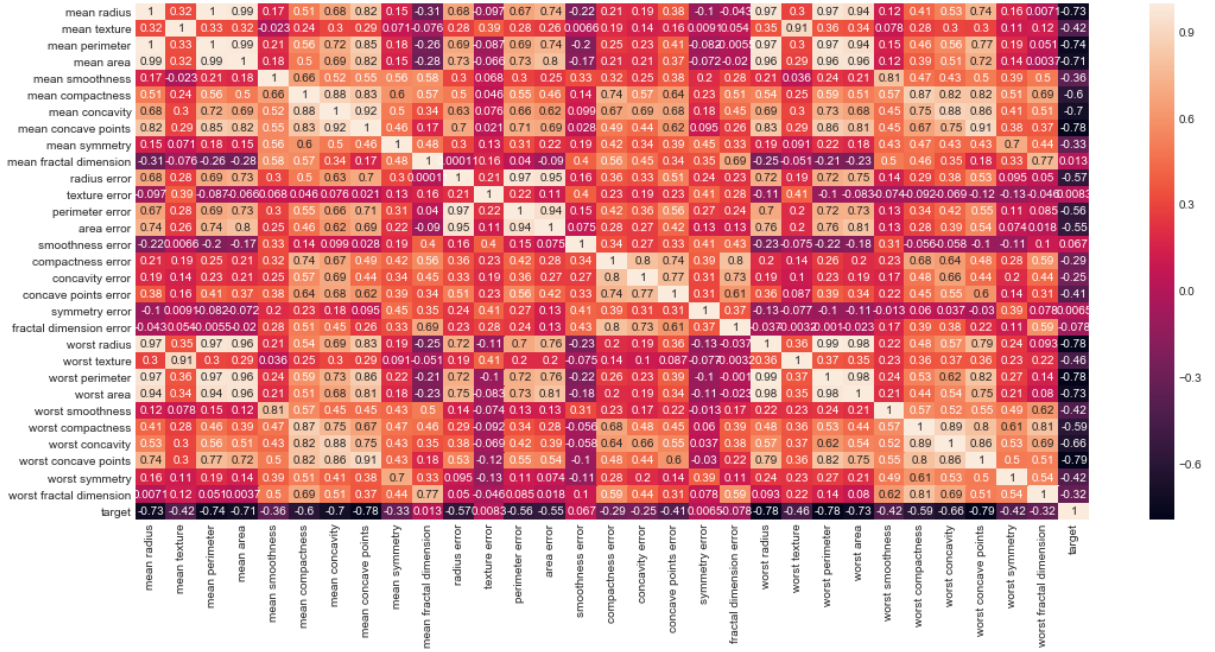
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x187e918ee80>



correlation between the variables

In [37]:
```
plt.figure(figsize=(18,8))
sns.heatmap(df_cancer.corr(), annot=True)
```

Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x187e948beb8>



Phase 3-Training the model:

From our dataset, let's create the target and predictor matrix

-"y" = Is the feature we are trying to predict (Output). In this case we are trying to predict if our "target" is Malignant or Benign. i.e. we are going to use the "target" feature here.

-"X" = The predictors which are the remaining columns (mean radius, mean texture, mean perimeter, mean area, mean smoothness, etc.)

In [39]:
```
#  dropping the target label coloumns
X = df_cancer.drop(['target'],axis=1)
```

In [40]:
```
X
```

Out[40]:

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	s
0	17.990	10.38	122.80	1001.0	0.11840	0.27760	0.300100	0.147100	0
1	20.570	17.77	132.90	1326.0	0.08474	0.07864	0.086900	0.070170	0
2	19.690	21.25	130.00	1203.0	0.10960	0.15990	0.197400	0.127900	0
3	11.420	20.38	77.58	386.1	0.14250	0.28390	0.241400	0.105200	0
4	20.290	14.34	135.10	1297.0	0.10030	0.13280	0.198000	0.104300	0
5	12.450	15.70	82.57	477.1	0.12780	0.17000	0.157800	0.080890	0
6	18.250	19.98	119.60	1040.0	0.09463	0.10900	0.112700	0.074000	0
7	13.710	20.83	90.20	577.9	0.11890	0.16450	0.093660	0.059850	0
8	13.000	21.82	87.50	519.8	0.12730	0.19320	0.185900	0.093530	0
9	12.460	24.04	83.97	475.9	0.11860	0.23960	0.227300	0.085430	0
10	16.020	23.24	102.70	797.8	0.08206	0.06669	0.032990	0.033230	0
11	15.780	17.89	103.60	781.0	0.09710	0.12920	0.099540	0.066060	0
12	19.170	24.80	132.40	1123.0	0.09740	0.24580	0.206500	0.111800	0
13	15.850	23.95	103.70	782.7	0.08401	0.10020	0.099380	0.053640	0
14	13.730	22.61	93.60	578.3	0.11310	0.22930	0.212800	0.080250	0
15	14.540	27.54	96.73	658.8	0.11390	0.15950	0.163900	0.073640	0
16	14.680	20.13	94.74	684.5	0.09867	0.07200	0.073950	0.052590	0
17	16.130	20.68	108.10	798.8	0.11700	0.20220	0.172200	0.102800	0
18	19.810	22.15	130.00	1260.0	0.09831	0.10270	0.147900	0.094980	0
19	13.540	14.36	87.46	566.3	0.09779	0.08129	0.066640	0.047810	0
20	13.080	15.71	85.63	520.0	0.10750	0.12700	0.045680	0.031100	0

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	s
21	9.504	12.44	60.34	273.9	0.10240	0.06492	0.029560	0.020760	0
22	15.340	14.26	102.50	704.4	0.10730	0.21350	0.207700	0.097560	0
23	21.160	23.04	137.20	1404.0	0.09428	0.10220	0.109700	0.086320	0
24	16.650	21.38	110.00	904.6	0.11210	0.14570	0.152500	0.091700	0
25	17.140	16.40	116.00	912.7	0.11860	0.22760	0.222900	0.140100	0
26	14.580	21.53	97.41	644.8	0.10540	0.18680	0.142500	0.087830	0
27	18.610	20.25	122.10	1094.0	0.09440	0.10660	0.149000	0.077310	0
28	15.300	25.27	102.40	732.4	0.10820	0.16970	0.168300	0.087510	0
29	17.570	15.05	115.00	955.1	0.09847	0.11570	0.098750	0.079530	0
...	...	...	...	...	...	...	...	...	.
539	7.691	25.44	48.34	170.4	0.08668	0.11990	0.092520	0.013640	0
540	11.540	14.44	74.65	402.9	0.09984	0.11200	0.067370	0.025940	0
541	14.470	24.99	95.81	656.4	0.08837	0.12300	0.100900	0.038900	0
542	14.740	25.42	94.70	668.6	0.08275	0.07214	0.041050	0.030270	0
543	13.210	28.06	84.88	538.4	0.08671	0.06877	0.029870	0.032750	0
544	13.870	20.70	89.77	584.8	0.09578	0.10180	0.036880	0.023690	0
545	13.620	23.23	87.19	573.2	0.09246	0.06747	0.029740	0.024430	0
546	10.320	16.35	65.31	324.9	0.09434	0.04994	0.010120	0.005495	0
547	10.260	16.58	65.85	320.8	0.08877	0.08066	0.043580	0.024380	0
548	9.683	19.34	61.05	285.7	0.08491	0.05030	0.023370	0.009615	0
549	10.820	24.21	68.89	361.6	0.08192	0.06602	0.015480	0.008160	0

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	s
550	10.860	21.48	68.51	360.5	0.07431	0.04227	0.000000	0.000000	0
551	11.130	22.44	71.49	378.4	0.09566	0.08194	0.048240	0.022570	0
552	12.770	29.43	81.35	507.9	0.08276	0.04234	0.019970	0.014990	0
553	9.333	21.94	59.01	264.0	0.09240	0.05605	0.039960	0.012820	0
554	12.880	28.92	82.50	514.3	0.08123	0.05824	0.061950	0.023430	0
555	10.290	27.61	65.67	321.4	0.09030	0.07658	0.059990	0.027380	0
556	10.160	19.59	64.73	311.7	0.10030	0.07504	0.005025	0.011160	0
557	9.423	27.88	59.26	271.3	0.08123	0.04971	0.000000	0.000000	0
558	14.590	22.68	96.39	657.1	0.08473	0.13300	0.102900	0.037360	0
559	11.510	23.93	74.52	403.5	0.09261	0.10210	0.111200	0.041050	0
560	14.050	27.15	91.38	600.4	0.09929	0.11260	0.044620	0.043040	0
561	11.200	29.37	70.67	386.0	0.07449	0.03558	0.000000	0.000000	0
562	15.220	30.62	103.40	716.9	0.10480	0.20870	0.255000	0.094290	0
563	20.920	25.09	143.00	1347.0	0.10990	0.22360	0.317400	0.147400	0
564	21.560	22.39	142.00	1479.0	0.11100	0.11590	0.243900	0.138900	0
565	20.130	28.25	131.20	1261.0	0.09780	0.10340	0.144000	0.097910	0
566	16.600	28.08	108.30	858.1	0.08455	0.10230	0.092510	0.053020	0
567	20.600	29.33	140.10	1265.0	0.11780	0.27700	0.351400	0.152000	0
568	7.760	24.54	47.92	181.0	0.05263	0.04362	0.000000	0.000000	0

569 rows × 30 columns

In [41]:
```
y=df_cancer['target']
```

In [42]:
```
y
```

Out[42]:
```
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
```

Splitting the dataset

-The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later on. We have the test dataset (or subset) in order to test our model's prediction on this subset.

In [43]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
```

```
0.20, random_state=5)
```

In [46]:
```
X_train.shape
```

Out[46]:
```
(455, 30)
```

In [47]:
```
 X_test.shape
```

Out[47]:
```
(114, 30)
```

In [49]:
```
y_train.shape
```

Out[49]:
```
(455,)
```

In [50]:
```
y_test.shape
```

Out[50]:
```
(114,)
```

In [56]:
```
from sklearn.svm import SVC
from sklearn.metrics import classification_report
```

training our SVM model with our "training" dataset.

In [58]:
```
svc_model=SVC()
svc_model.fit(X_train,y_train)
```

Out[58]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

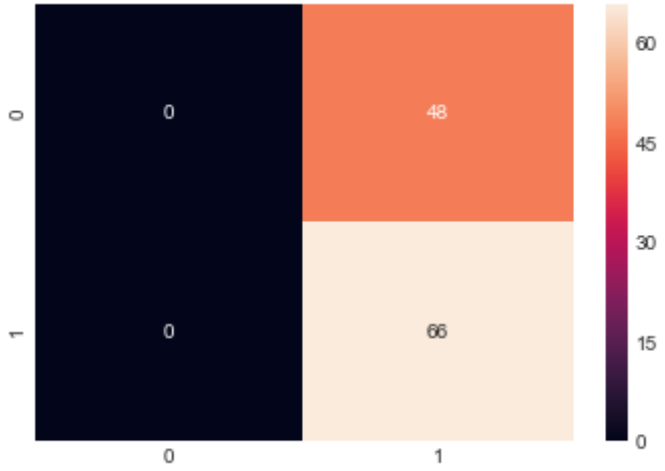
Phase 4-EVALUATING THE MODEL:

Confusion matrix(to evaluate the accuracy of a classification)

The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

In [61]:
```
from sklearn.metrics import confusion_matrix
y_predict = svc_model.predict(X_test)
cm = confusion_matrix(y_test, y_predict)
sns.heatmap(cm, annot=True)
```

Out[61]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x187eb9817f0>
```



In [63]:
```
print(classification_report(y_test, y_predict))
```

```
             precision    recall  f1-score   support

        0.0       0.00      0.00      0.00        48
        1.0       0.58      1.00      0.73        66

avg / total       0.34      0.58      0.42       114
```

```
F:\justforAnaconda\AnaNew\lib\site-packages\sklearn\metrics\classific
ation.py:1135: UndefinedMetricWarning: Precision and F-score are ill-
```

```
defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

As we can see, our model did not do a good job in its predictions. It predicted that 48 healthy patients have cancer. We only achieved 34% accuracy.

Phase 5-Improving our Model 1:

Normalize Training Data

Data normalization is a feature scaling process that brings all values into range [0,1]

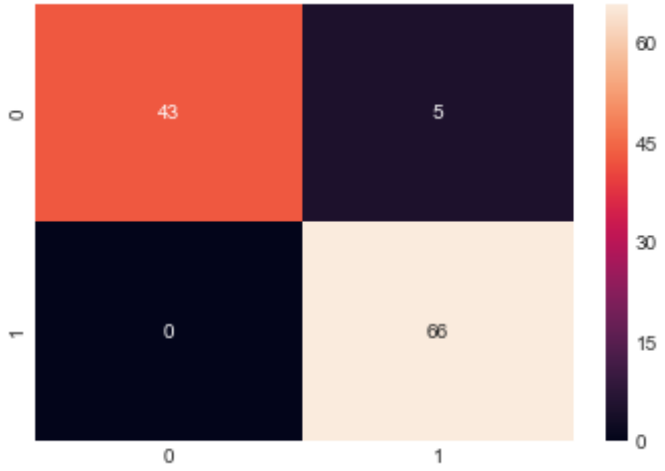
X' = (X-X_min) / (X_max — X_min)

In [65]:
```
min_train = X_train.min()
range_train = (X_train - min_train).max()
X_train_scaled = (X_train - min_train)/range_train
```

In [66]:
```
min_test=X_test.min()
range_test=(X_test - min_test).max()
X_test_scaled = (X_test - min_test)/range_test
```

In [72]:
```
svc_model = SVC()
svc_model.fit(X_train_scaled, y_train)
y_predict = svc_model.predict(X_test_scaled)
cm = confusion_matrix(y_test, y_predict)
sns.heatmap(cm,annot=True,fmt="d")
```

Out[72]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x187eb9f29b0>
```



In [73]:
```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.90      0.95        48
         1.0       0.93      1.00      0.96        66

avg / total       0.96      0.96      0.96       114
```

Improving our Model 2 :

In [80]:
```
param_grid={'C':[.1,1,10,100],
            'gamma':[1,.1,.01,.001],
            'kernel':['rbf']}
```

In [81]:
```
from sklearn.model_selection import GridSearchCV
grid=GridSearchCV(SVC(),param_grid,refit=True,verbose=4)
grid.fit(X_train_scaled, y_train)
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
[CV] C=0.1, gamma=1, kernel=rbf .....................................
```

```
[CV]  C=0.1, gamma=1, kernel=rbf, score=0.9671052631578947, total=   0.
0s
[CV] C=0.1, gamma=1, kernel=rbf .....................................
[CV]  C=0.1, gamma=1, kernel=rbf, score=0.9210526315789473, total=   0.
0s
[CV] C=0.1, gamma=1, kernel=rbf .....................................
[CV]  C=0.1, gamma=1, kernel=rbf, score=0.9470198675496688, total=   0.
0s
[CV] C=0.1, gamma=0.1, kernel=rbf ...................................
[CV]  C=0.1, gamma=0.1, kernel=rbf, score=0.9144736842105263, total=
0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ...................................
[CV]  C=0.1, gamma=0.1, kernel=rbf, score=0.8881578947368421, total=
0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ...................................
[CV]  C=0.1, gamma=0.1, kernel=rbf, score=0.8675496688741722, total=
0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ..................................
[CV]  C=0.1, gamma=0.01, kernel=rbf, score=0.6381578947368421, total=
0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ..................................
[CV]  C=0.1, gamma=0.01, kernel=rbf, score=0.6381578947368421, total=
0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ..................................
[CV]  C=0.1, gamma=0.01, kernel=rbf, score=0.6423841059602649, total=
0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf .................................
[CV]  C=0.1, gamma=0.001, kernel=rbf, score=0.6381578947368421, total=
  0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf .................................
[CV]  C=0.1, gamma=0.001, kernel=rbf, score=0.6381578947368421, total=
  0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf .................................
[CV]  C=0.1, gamma=0.001, kernel=rbf, score=0.6423841059602649, total=
  0.0s
[CV] C=1, gamma=1, kernel=rbf .......................................
[CV]  C=1, gamma=1, kernel=rbf, score=0.993421052631579, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf .......................................
[CV]  C=1, gamma=1, kernel=rbf, score=0.9473684210526315, total=   0.0s
```

```
[CV] C=1, gamma=1, kernel=rbf .....................................
[CV]  C=1, gamma=1, kernel=rbf, score=0.9801324503311258, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf .....................................
[CV]  C=1, gamma=0.1, kernel=rbf, score=0.9736842105263158, total=   0.
0s
[CV] C=1, gamma=0.1, kernel=rbf .....................................
[CV]  C=1, gamma=0.1, kernel=rbf, score=0.9276315789473685, total=   0.
0s
[CV] C=1, gamma=0.1, kernel=rbf .....................................
[CV]  C=1, gamma=0.1, kernel=rbf, score=0.9403973509933775, total=   0.
0s
[CV] C=1, gamma=0.01, kernel=rbf ....................................
[CV]  C=1, gamma=0.01, kernel=rbf, score=0.9144736842105263, total=
0.0s
[CV] C=1, gamma=0.01, kernel=rbf ....................................
[CV]  C=1, gamma=0.01, kernel=rbf, score=0.8947368421052632, total=
0.0s
[CV] C=1, gamma=0.01, kernel=rbf ....................................
[CV]  C=1, gamma=0.01, kernel=rbf, score=0.8675496688741722, total=
0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV]  C=1, gamma=0.001, kernel=rbf, score=0.6381578947368421, total=
0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:
    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:
    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:
    0.0s
[CV]  C=1, gamma=0.001, kernel=rbf, score=0.6381578947368421, total=
0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV]  C=1, gamma=0.001, kernel=rbf, score=0.6423841059602649, total=
0.0s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV]  C=10, gamma=1, kernel=rbf, score=0.993421052631579, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV]  C=10, gamma=1, kernel=rbf, score=0.9605263157894737, total=   0.0
```

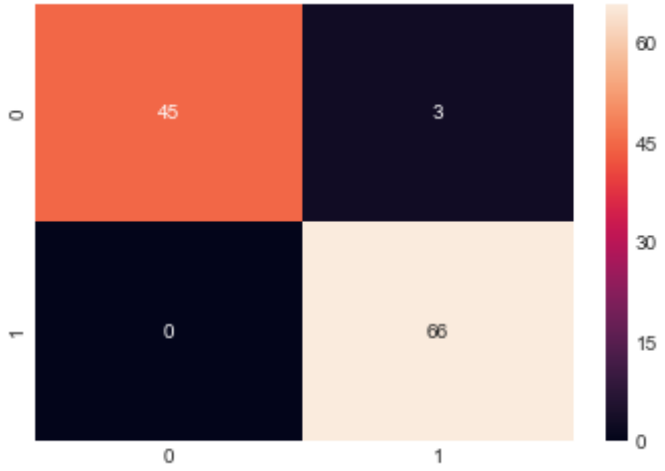
```
[CV] C=10, gamma=1, kernel=rbf ..... [CV] C=10, gamma=1, kernel=rbf, score=0.9735099337748344, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf ..... [CV] C=10, gamma=0.1, kernel=rbf, score=0.993421052631579, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf ..... [CV] C=10, gamma=0.1, kernel=rbf, score=0.9671052631578947,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

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

  0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV]  C=100, gamma=0.001, kernel=rbf, score=0.9210526315789473, total=
  0.0s
[CV] C=100, gamma=0.001, kernel=rbf ..................................
[CV]  C=100, gamma=0.001, kernel=rbf, score=0.9403973509933775, total=
  0.0s

[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:    0.2s finished

Out[81]: GridSearchCV(cv=None, error_score='raise',
       estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.
0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.00
1], 'kernel': ['rbf']},
```

```
        pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
        scoring=None, verbose=4)
```

In [84]:
```
grid_prediction = grid.predict(X_test_scaled)
cm = confusion_matrix(y_test, grid_prediction)
sns.heatmap(cm,annot=True,fmt="d")
```

Out[84]: <matplotlib.axes._subplots.AxesSubplot at 0x187ec1603c8>



In [87]:
```
print(classification_report(y_test,y_predict))
```

```
             precision    recall  f1-score   support

        0.0       1.00      0.90      0.95        48
        1.0       0.93      1.00      0.96        66

avg / total       0.96      0.96      0.96       114
```

Our prediction got a lot better with only 1 false prediction(Predicted cancer instead of healthy)