IMPORTING LIBRARIES

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

LOADING DATASETS

In [2]:
```
#Loading Datasets
data=pd.read_csv('Cancer.csv')
cancer=pd.DataFrame(data)
```

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

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100

5 rows × 33 columns

In [4]:
```
#checking shape of the dataframe
cancer.shape
```

Out[4]:
```
(569, 33)
```

In [5]:
```
cancer['diagnosis'].value_counts()
```

Out[5]:
```
B    357
M    212
Name: diagnosis, dtype: int64
```

In [6]:
```
cancer.columns
```

Out[6]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'Unnamed: 32'],
      dtype='object')
```

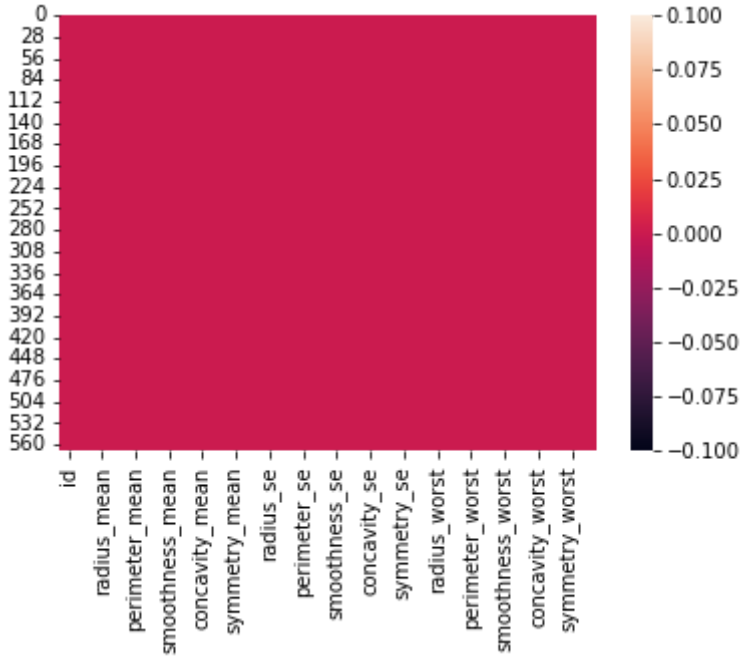
In [7]:
```
cancer.isnull().sum()
```

Out[7]:
```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32                569
dtype: int64
```

In [8]:
```
del cancer['Unnamed: 32']
```

In [9]:
```
sns.heatmap(cancer.isnull())
```

Out[9]:
```
<AxesSubplot:>
```



In [10]:
```
cancer.corr()
```

Out[10]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smo
id	1.000000	0.074626	0.099770	0.073159	0.096893	
radius_mean	0.074626	1.000000	0.323782	0.997855	0.987357	
texture_mean	0.099770	0.323782	1.000000	0.329533	0.321086	
perimeter_mean	0.073159	0.997855	0.329533	1.000000	0.986507	
area_mean	0.096893	0.987357	0.321086	0.986507	1.000000	
smoothness_mean	-0.012968	0.170581	-0.023389	0.207278	0.177028	
compactness_mean	0.000096	0.506124	0.236702	0.556936	0.498502	
concavity_mean	0.050080	0.676764	0.302418	0.716136	0.685983	
concave points_mean	0.044158	0.822529	0.293464	0.850977	0.823269	
symmetry_mean	-0.022114	0.147741	0.071401	0.183027	0.151293	
fractal_dimension_mean	-0.052511	-0.311631	-0.076437	-0.261477	-0.283110	
radius_se	0.143048	0.679090	0.275869	0.691765	0.732562	
texture_se	-0.007526	-0.097317	0.386358	-0.086761	-0.066280	
perimeter_se	0.137331	0.674172	0.281673	0.693135	0.726628	
area_se	0.177742	0.735864	0.259845	0.744983	0.800086	
smoothness_se	0.096781	-0.222600	0.006614	-0.202694	-0.166777	
compactness_se	0.033961	0.206000	0.191975	0.250744	0.212583	
concavity_se	0.055239	0.194204	0.143293	0.228082	0.207660	
concave points_se	0.078768	0.376169	0.163851	0.407217	0.372320	
symmetry_se	-0.017306	-0.104321	0.009127	-0.081629	-0.072497	
fractal_dimension_se	0.025725	-0.042641	0.054458	-0.005523	-0.019887	
radius_worst	0.082405	0.969539	0.352573	0.969476	0.962746	
texture_worst	0.064720	0.297008	0.912045	0.303038	0.287489	
perimeter_worst	0.079986	0.965137	0.358040	0.970387	0.959120	
area_worst	0.107187	0.941082	0.343546	0.941550	0.959213	
smoothness_worst	0.010338	0.119616	0.077503	0.150549	0.123523	
compactness_worst	-0.002968	0.413463	0.277830	0.455774	0.390410	
concavity_worst	0.023203	0.526911	0.301025	0.563879	0.512606	
concave points_worst	0.035174	0.744214	0.295316	0.771241	0.722017	
symmetry_worst	-0.044224	0.163953	0.105008	0.189115	0.143570	
fractal_dimension_worst	-0.029866	0.007066	0.119205	0.051019	0.003738	

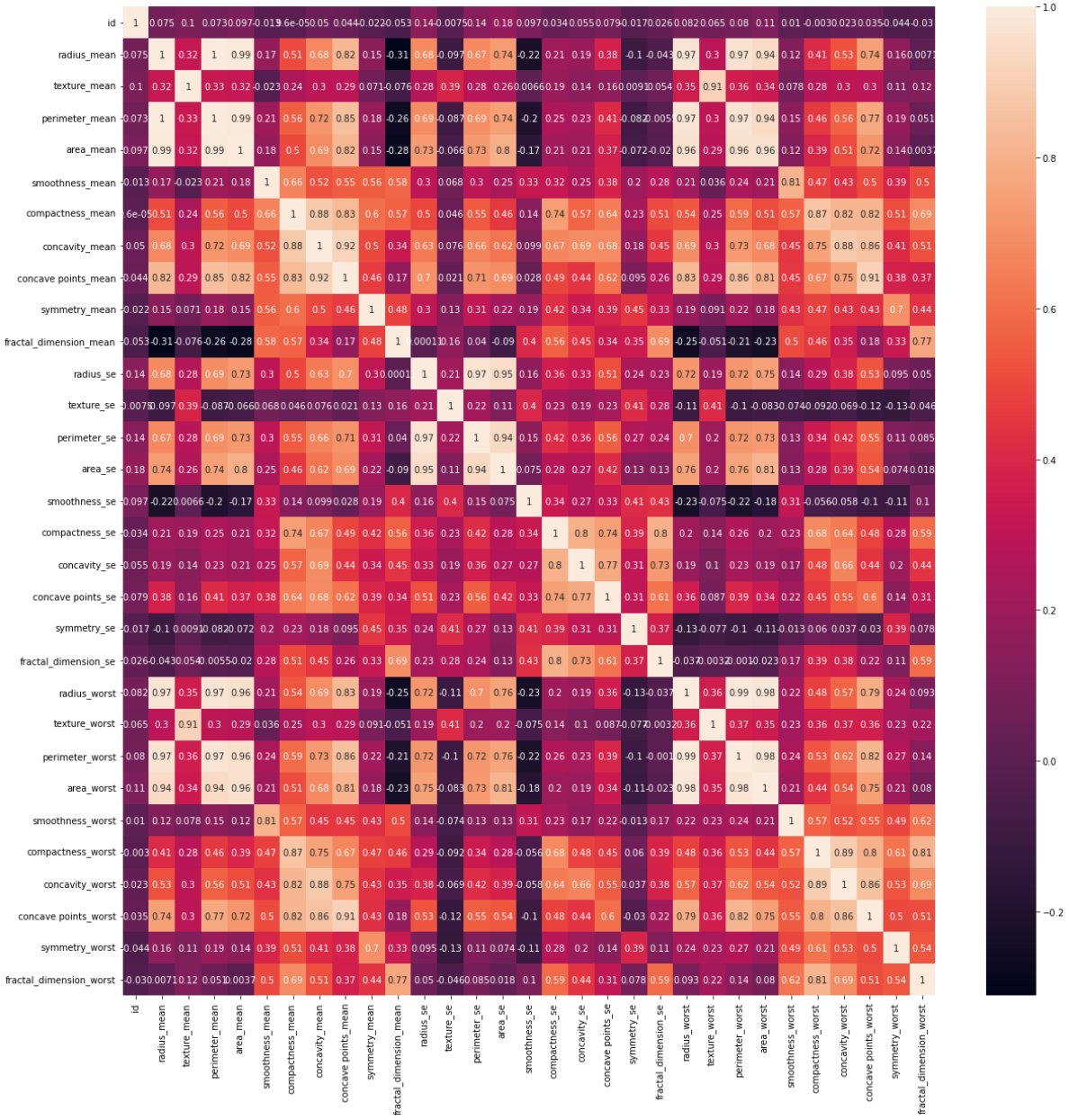
31 rows × 31 columns

In [11]:
```
f = plt.figure()
f.set_figwidth(20)
```

```
f.set_figheight(20)
sns.heatmap(cancer.corr(),annot=True)
```

Out[11]:

```
<AxesSubplot:>
```



In [12]:

```
# 1 for BENIGN and 0 for MALIGNANT
cancer['target']=np.where(cancer['diagnosis']=='B',1,0)
```

In [13]:

```
cancer.head()
```

Out[13]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
0	842302	M	17.99	10.38	122.80	1001.0	0.118
1	842517	M	20.57	17.77	132.90	1326.0	0.084
2	84300903	M	19.69	21.25	130.00	1203.0	0.109
3	84348301	M	11.42	20.38	77.58	386.1	0.142
4	84358402	M	20.29	14.34	135.10	1297.0	0.100

5 rows × 33 columns

In [14]:
```
cancer.tail()
```

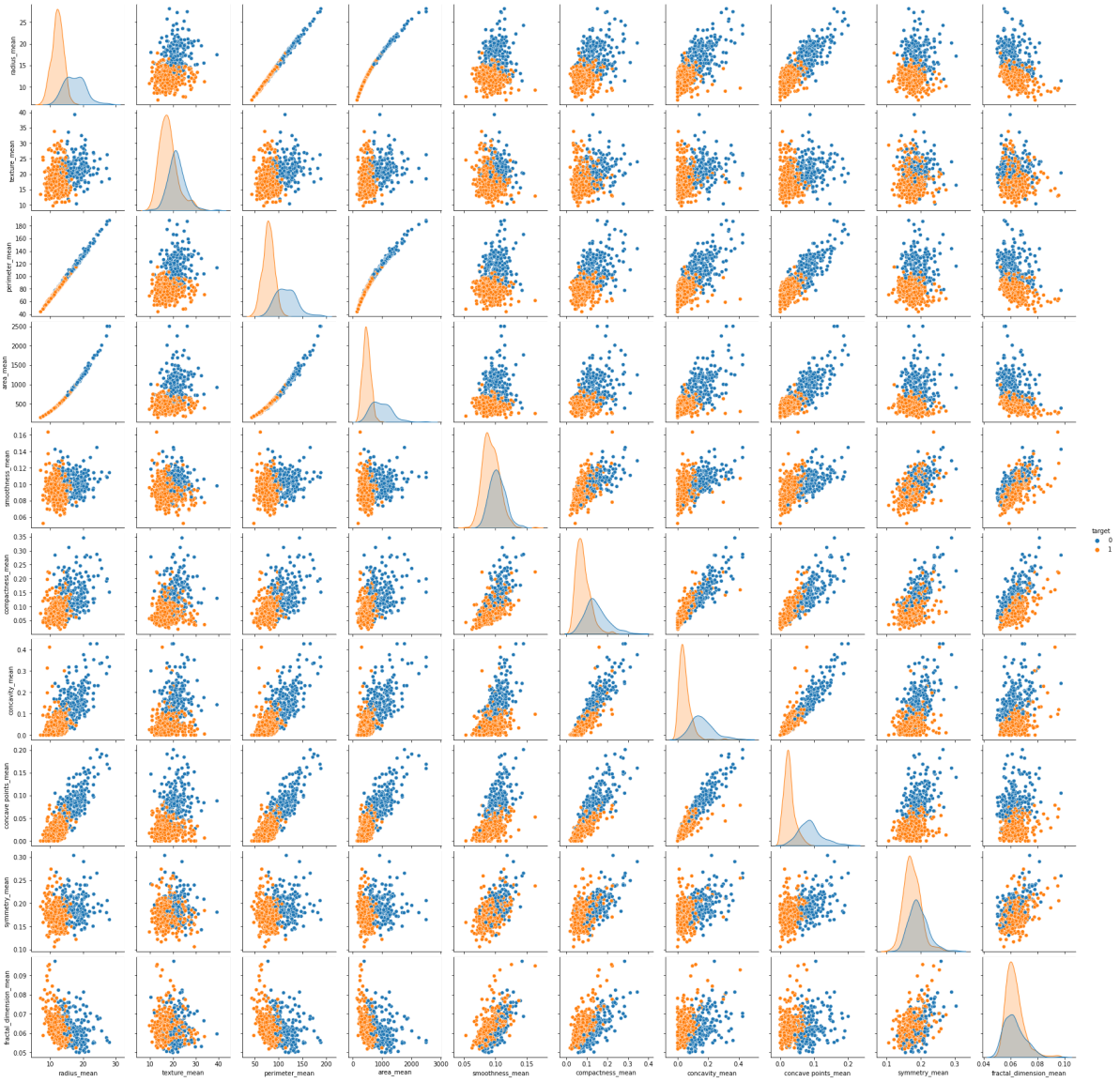
Out[14]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_me
564	926424	M	21.56	22.39	142.00	1479.0	0.111
565	926682	M	20.13	28.25	131.20	1261.0	0.097
566	926954	M	16.60	28.08	108.30	858.1	0.084
567	927241	M	20.60	29.33	140.10	1265.0	0.117
568	92751	B	7.76	24.54	47.92	181.0	0.052

5 rows × 33 columns

In [15]:
```
sns.pairplot(cancer,hue='target',vars=['radius_mean', 'texture_mean', 'perimeter_me
        'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
        'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean'])
```

Out[15]:
```
<seaborn.axisgrid.PairGrid at 0x27a85c4aca0>
```



In [16]:
```
cancer.columns
```

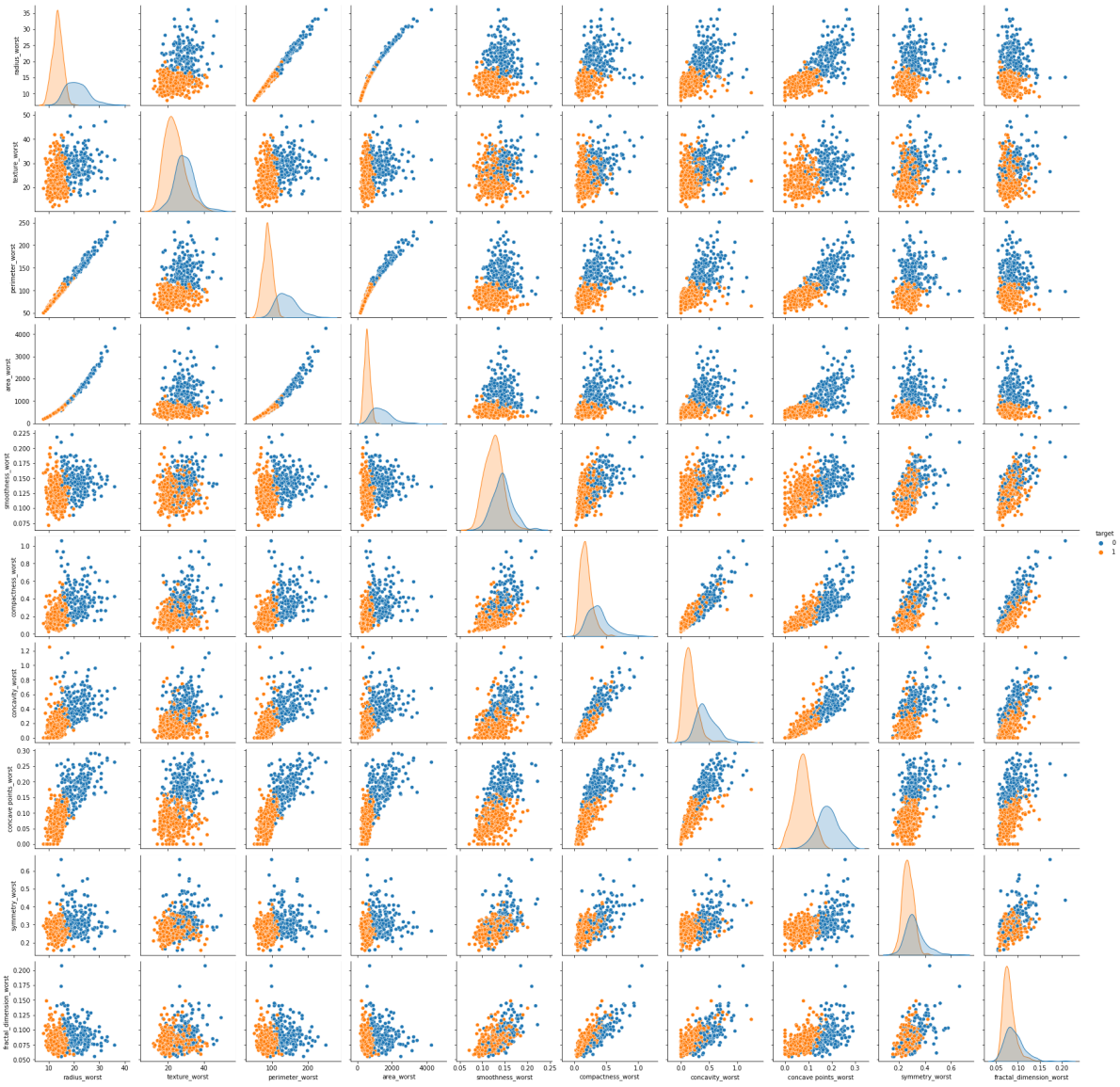
Out[16]:
```
Index(['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'target'],
      dtype='object')
```

In [17]:
```
sns.pairplot(cancer,hue='target',vars=['radius_se', 'texture_se', 'perimeter_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se'])
```

Out[17]:
```
<seaborn.axisgrid.PairGrid at 0x27a8bae8b20>
```

In [18]:
```
sns.pairplot(cancer,hue='target',vars=['radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst'])
```

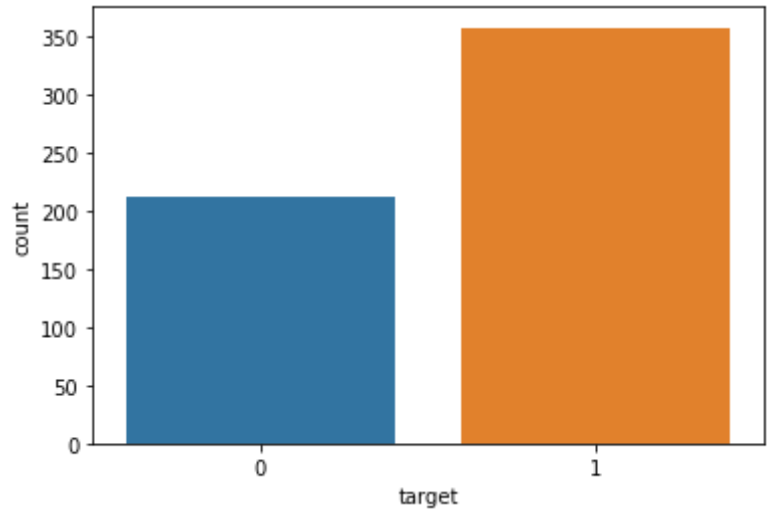
Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x27a91af0820>
```



In [19]:
```
sns.countplot('target',data=cancer)
```

```
C:\Users\kgyan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variable as a keyword arg: x. From version 0.12, the only va
lid positional argument will be `data`, and passing other arguments without an exp
licit keyword will result in an error or misinterpretation.
  warnings.warn(
```

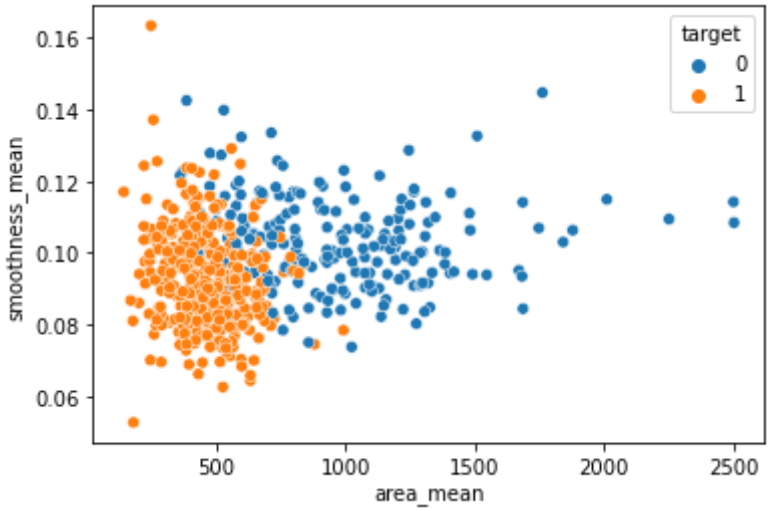
Out[19]:
```
<AxesSubplot:xlabel='target', ylabel='count'>
```



In [20]:
```
sns.scatterplot('area_mean','smoothness_mean',data=cancer,hue='target')
```

```
C:\Users\kgyan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

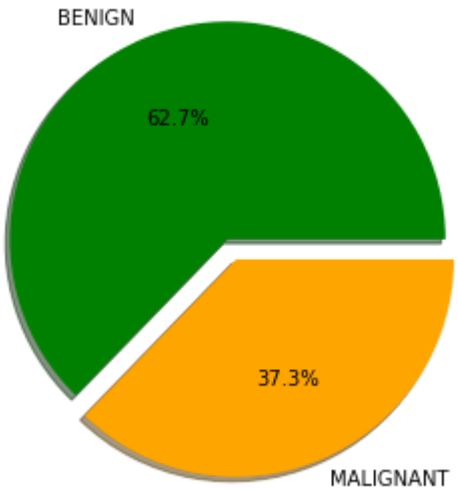
Out[20]:
```
<AxesSubplot:xlabel='area_mean', ylabel='smoothness_mean'>
```



In [21]:
```
data=cancer['diagnosis'].value_counts()
labels=['BENIGN','MALIGNANT']
explode=[0,0.1]
```

In [22]:
```
plt.figure(figsize=(5,5))
plt.pie(data,labels=labels,explode=explode,shadow=True,autopct='%1.1f%%',colors=['
```

Out[22]:
```
([<matplotlib.patches.Wedge at 0x27a96e776d0>,
  <matplotlib.patches.Wedge at 0x27a96e860a0>],
 [Text(-0.4286546999573329, 1.0130425204326268, 'BENIGN'),
  Text(0.46762320557394, -1.1051373387994603, 'MALIGNANT')],
 [Text(-0.23381165452218156, 0.5525686475087055, '62.7%'),
  Text(0.27278020325146496, -0.6446634476330184, '37.3%')])
```



In [23]:
```
df=cancer[['radius_mean', 'texture_mean', 'perimeter_mean',
       'area_mean', 'smoothness_mean', 'compactness_mean', 'concavity_mean',
       'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean',
       'radius_se', 'texture_se', 'perimeter_se', 'area_se', 'smoothness_se',
       'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
       'fractal_dimension_se', 'radius_worst', 'texture_worst',
       'perimeter_worst', 'area_worst', 'smoothness_worst',
```

```
       'compactness_worst', 'concavity_worst', 'concave points_worst',
       'symmetry_worst', 'fractal_dimension_worst', 'target']]
```

In [24]:
```
#Training our data set
X=df.drop('target',axis=1)
y=df['target']
```

In [25]:
```
X
```

Out[25]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_m
0	17.99	10.38	122.80	1001.0	0.11840	0.27
1	20.57	17.77	132.90	1326.0	0.08474	0.07
2	19.69	21.25	130.00	1203.0	0.10960	0.15
3	11.42	20.38	77.58	386.1	0.14250	0.28
4	20.29	14.34	135.10	1297.0	0.10030	0.13
...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11
565	20.13	28.25	131.20	1261.0	0.09780	0.10
566	16.60	28.08	108.30	858.1	0.08455	0.10
567	20.60	29.33	140.10	1265.0	0.11780	0.27
568	7.76	24.54	47.92	181.0	0.05263	0.04

569 rows × 30 columns

In [26]:
```
y
```

Out[26]:
```
0      0
1      0
2      0
3      0
4      0
      ..
564    0
565    0
566    0
567    0
568    1
Name: target, Length: 569, dtype: int32
```

In [27]:
```
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_sta
```

In [28]:
```
X_train
```

Out[28]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_m
306	13.200	15.82	84.07	537.3	0.08511	0.05
410	11.360	17.57	72.49	399.8	0.08858	0.05
197	18.080	21.84	117.40	1024.0	0.07371	0.08
376	10.570	20.22	70.15	338.3	0.09073	0.16
244	19.400	23.50	129.10	1155.0	0.10270	0.15
...	...	...	...	...	...	
8	13.000	21.82	87.50	519.8	0.12730	0.19
73	13.800	15.79	90.43	584.1	0.10070	0.12
400	17.910	21.02	124.40	994.0	0.12300	0.25
118	15.780	22.91	105.70	782.6	0.11550	0.17
206	9.876	17.27	62.92	295.4	0.10890	0.07

455 rows × 30 columns

In [29]:
```
y_train
```

Out[29]:
```
306    1
410    1
197    0
376    1
244    0
      ..
8      0
73     0
400    0
118    0
206    1
Name: target, Length: 455, dtype: int32
```

In [30]:
```
X_test
```

Out[30]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_m
28	15.30	25.27	102.40	732.4	0.10820	0.16
163	12.34	22.22	79.85	464.5	0.10120	0.10
123	14.50	10.89	94.28	640.7	0.11010	0.10
361	13.30	21.57	85.24	546.1	0.08582	0.06
549	10.82	24.21	68.89	361.6	0.08192	0.06
...	...	...	...	...	...	
414	15.13	29.81	96.71	719.5	0.08320	0.04
515	11.34	18.61	72.76	391.2	0.10490	0.08
186	18.31	18.58	118.60	1041.0	0.08588	0.08
3	11.42	20.38	77.58	386.1	0.14250	0.28
261	17.35	23.06	111.00	933.1	0.08662	0.06

114 rows × 30 columns

In [31]:
```
y_test
```

Out[31]:
```
28     0
163    1
123    1
361    1
549    1
      ..
414    0
515    1
186    0
3      0
261    0
Name: target, Length: 114, dtype: int32
```

In [32]:
```
from sklearn.svm import SVC
from sklearn.metrics import classification_report,confusion_matrix
```

In [33]:
```
svc_model=SVC()
```

In [34]:
```
svc_model.fit(X_train,y_train)
```

Out[34]:
```
SVC()
```

In [35]:
```
X_train
```

Out[35]:

	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_m
306	13.200	15.82	84.07	537.3	0.08511	0.05
410	11.360	17.57	72.49	399.8	0.08858	0.05
197	18.080	21.84	117.40	1024.0	0.07371	0.08
376	10.570	20.22	70.15	338.3	0.09073	0.16
244	19.400	23.50	129.10	1155.0	0.10270	0.15
...	...	...	...	...	...	
8	13.000	21.82	87.50	519.8	0.12730	0.19
73	13.800	15.79	90.43	584.1	0.10070	0.12
400	17.910	21.02	124.40	994.0	0.12300	0.25
118	15.780	22.91	105.70	782.6	0.11550	0.17
206	9.876	17.27	62.92	295.4	0.10890	0.07

455 rows × 30 columns

In [36]:
```
y_train
```

Out[36]:
```
306    1
410    1
197    0
376    1
244    0
      ..
8      0
73     0
400    0
118    0
206    1
Name: target, Length: 455, dtype: int32
```

In [37]:
```
y_predict=svc_model.predict(X_test)
```

In [38]:
```
y_predict
```

Out[38]:
```
array([0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0])
```

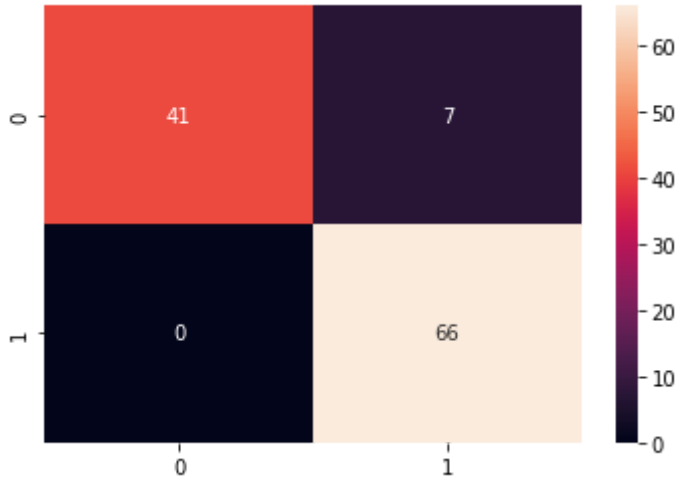
In [39]:
```
y_test
```

Out[39]:
```
28     0
163    1
123    1
361    1
549    1
      ..
414    0
515    1
186    0
3      0
261    0
Name: target, Length: 114, dtype: int32
```

In [40]:
```
met=confusion_matrix(y_test,y_predict)
```

In [41]:
```
sns.heatmap(met,annot=True)
```

Out[41]:
```
<AxesSubplot:>
```



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

```
              precision    recall  f1-score   support

           0       1.00      0.85      0.92        48
           1       0.90      1.00      0.95        66

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

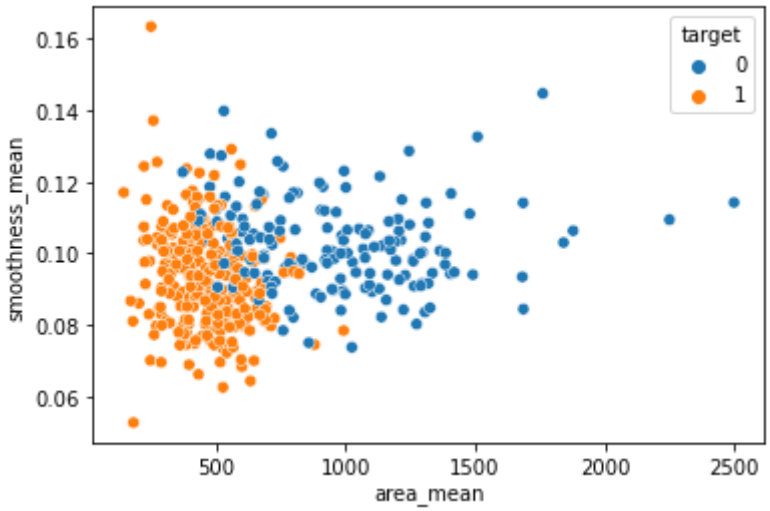
IMPROVING THE MODEL

In [43]:
```
# TYPE 1
#DATA NORMALIIZATION
min_train=X_train.min()
range_train=(X_train-min_train).max()
new_train=(X_train-min_train)/range_train
```

In [44]:
```
#BEFORE APPLYING NORMALIZATION
sns.scatterplot(X_train['area_mean'],X_train['smoothness_mean'],hue=y_train)
```

```
C:\Users\kgyan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

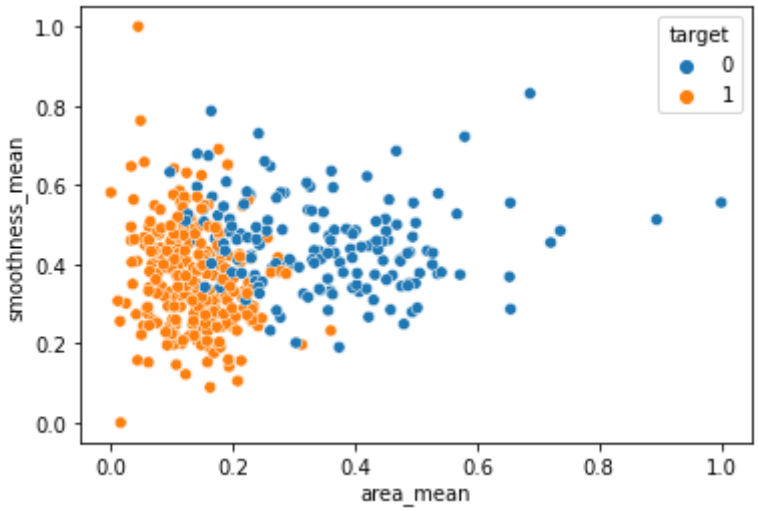
Out[44]:
```
<AxesSubplot:xlabel='area_mean', ylabel='smoothness_mean'>
```



In [45]:
```
#AFTER APPLYING NORMALIZATION
sns.scatterplot(new_train['area_mean'],new_train['smoothness_mean'],hue=y_train)
```

```
C:\Users\kgyan\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarnin
g: Pass the following variables as keyword args: x, y. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an e
xplicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[45]:
```
<AxesSubplot:xlabel='area_mean', ylabel='smoothness_mean'>
```



In [46]:
```
min_test=X_test.min()
range_test=(X_test-min_test).max()
new_test=(X_test-min_test)/range_test
```

In [47]:
```
svc_model.fit(new_train,y_train)
```

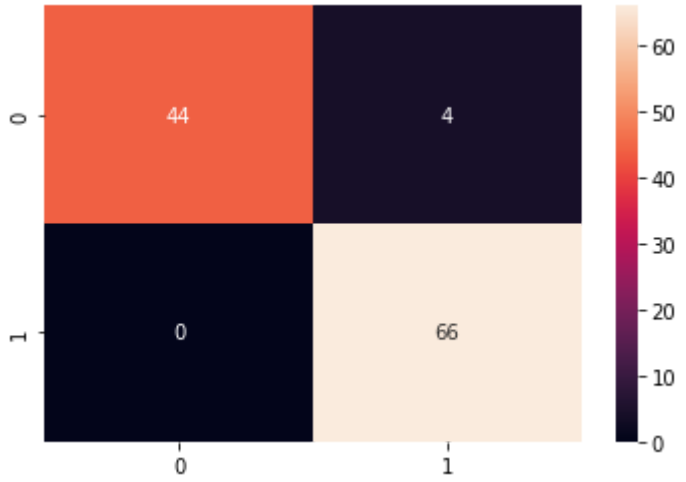
Out[47]:
```
SVC()
```

In [48]:
```
y_predict=svc_model.predict(new_test)
```

In [49]:
```
cmm=confusion_matrix(y_test,y_predict)
```

In [50]:
```
sns.heatmap(cmm,annot=True)
```

Out[50]:
```
<AxesSubplot:>
```



In [51]:
```
print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        48
           1       0.94      1.00      0.97        66

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

METHOD 2

USING SVM AND OPTIMIZING THE C AND GAMMA PARAMETER

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

In [53]:
```
from sklearn.model_selection import GridSearchCV
grid=GridSearchCV(SVC(),param_grid,refit=True,verbose=4)
grid
```

Out[53]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001],
                         'kernel': ['rbf']},
             verbose=4)
```

In [54]:
```
grid.fit(new_train,y_train)
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
[CV 1/5] END ........C=0.1, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.945 total time=   0.0s
[CV 3/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.912 total time=   0.0s
[CV 4/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.956 total time=   0.0s
[CV 5/5] END ........C=0.1, gamma=1, kernel=rbf;, score=0.934 total time=   0.0s
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.945 total time=   0.0s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.901 total time=   0.0s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.890 total time=   0.0s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.923 total time=   0.0s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.868 total time=   0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.648 total time=   0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.648 total time=   0.0s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.637 total time=   0.0s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.637 total time=   0.0s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.637 total time=   0.0s
[CV 1/5] END .........C=10, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END .........C=10, gamma=1, kernel=rbf;, score=0.967 total time=   0.0s
[CV 3/5] END .........C=10, gamma=1, kernel=rbf;, score=0.956 total time=   0.0s
[CV 4/5] END .........C=10, gamma=1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END .........C=10, gamma=1, kernel=rbf;, score=0.956 total time=   0.0s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.967 total time=   0.0s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.967 total time=   0.0s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.989 total time=   0.0s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.945 total time=   0.0s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.989 total time=   0.0s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.945 total time=   0.0s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.923 total time=   0.0s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.967 total time=   0.0s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.934 total time=   0.0s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.945 total time=   0.0s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.901 total time=   0.0s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.879 total time=   0.0s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.923 total time=   0.0s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.879 total time=   0.0s
[CV 1/5] END ........C=100, gamma=1, kernel=rbf;, score=0.956 total time=   0.0s
[CV 2/5] END ........C=100, gamma=1, kernel=rbf;, score=0.956 total time=   0.0s
[CV 3/5] END ........C=100, gamma=1, kernel=rbf;, score=0.945 total time=   0.0s
[CV 4/5] END ........C=100, gamma=1, kernel=rbf;, score=0.989 total time=   0.0s
[CV 5/5] END ........C=100, gamma=1, kernel=rbf;, score=0.967 total time=   0.0s
[CV 1/5] END ......C=100, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.967 total time=   0.0s
[CV 3/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.945 total time=   0.0s
[CV 4/5] END ......C=100, gamma=0.1, kernel=rbf;, score=1.000 total time=   0.0s
[CV 5/5] END ......C=100, gamma=0.1, kernel=rbf;, score=0.956 total time=   0.0s
[CV 1/5] END .....C=100, gamma=0.01, kernel=rbf;, score=1.000 total time=   0.0s
[CV 2/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.967 total time=   0.0s
[CV 3/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.967 total time=   0.0s
[CV 4/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.989 total time=   0.0s
[CV 5/5] END .....C=100, gamma=0.01, kernel=rbf;, score=0.945 total time=   0.0s
[CV 1/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.989 total time=   0.0s
[CV 2/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.945 total time=   0.0s
[CV 3/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.923 total time=   0.0s
[CV 4/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.967 total time=   0.0s
[CV 5/5] END ....C=100, gamma=0.001, kernel=rbf;, score=0.934 total time=   0.0s
```

Out[54]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001],
                         'kernel': ['rbf']},
             verbose=4)
```

In [55]:
```
grid.best_params_
```

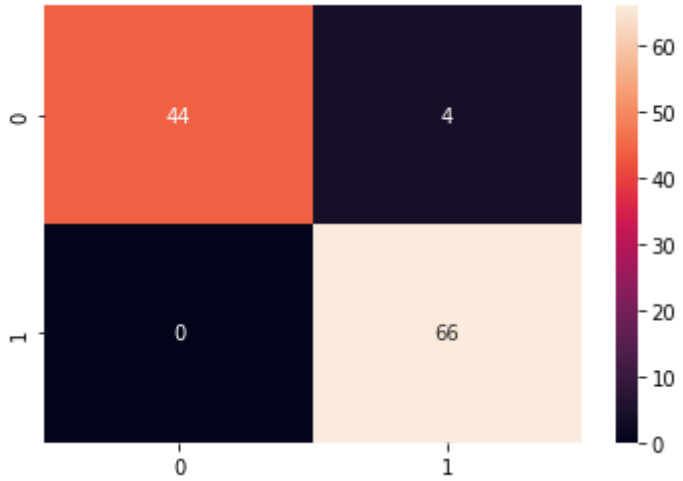
Out[55]:
```
{'C': 10, 'gamma': 1, 'kernel': 'rbf'}
```

In [56]:
```
y_predict1=grid.predict(new_test)
```

In [57]:
```
cmmm=confusion_matrix(y_test,y_predict1)
```

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

Out[58]:
```
<AxesSubplot:>
```



In [59]:
```
print(classification_report(y_test,y_predict1))
```

```
              precision    recall  f1-score   support

           0       1.00      0.92      0.96        48
           1       0.94      1.00      0.97        66

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

In [ ]:
```

```