Fatec Mogi Mirim

Projeto de IA - 1 Semestre de 2020

Bibliotecas

- Pandas
  - É uma lib Python focada em manipulação de dados
  - Site Oficial da Pandas (https://pandas.pydata.org/)
  - Para instalar a lib utilize o comando abaixo:

```
pip install pandas
```

- NumPy
  - É uma lib Python de codigo aberto focado em processamento computacional numerico
  - Site Oficial da NumPy (https://numpy.org/)
  - Para instalar a lib utilize o comando abaixo:

```
pip install numpy
```

- Matplotlib
  - Lib Python focada em criação de gráficos
  - Site Oficial da Matplotlib (https://matplotlib.org/)
  - Para instalar a lib utilize o comando abaixo:

```
pip install matplotlib
```

- Seaborn
  - Lib Python baseada em Matplotlib focada em visualização de gráficos estatisticos
  - Site Oficial da Seaborn (https://matplotlib.org/)
  - Para instalar a lib utilize o comando abaixo:

```
pip install seaborn
```

- Scikit-learn (Sklearn)
  - Lib Python que oferece uma gama de ferramentas para fazer analise de dados
  - Site Oficial do Scikit-learn (https://scikit-learn.org/stable/)
  - Para instalar a lib utilize o comando abaixo:

```
pip install sklearn
```

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

Importando os Dados Base

In [7]:
```
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
cancer
cancer.keys()
print(cancer['DESCR'])
print(cancer['target_names'])
print(cancer['target'])
print(cancer['feature_names'])
print(cancer['data'])
```

```
.. _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
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, fiel
d
        10 is Radius SE, field 20 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 extracti
on
     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 techni
ques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
     163-171.
['malignant' 'benign']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 0 1 0 1 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 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 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 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 0 1 1 0 1 0 1 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 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 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 0 0 1 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 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 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]
['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']
[[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]]
```

Organizando a Tabela de forma visivel

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

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

Out[8]:

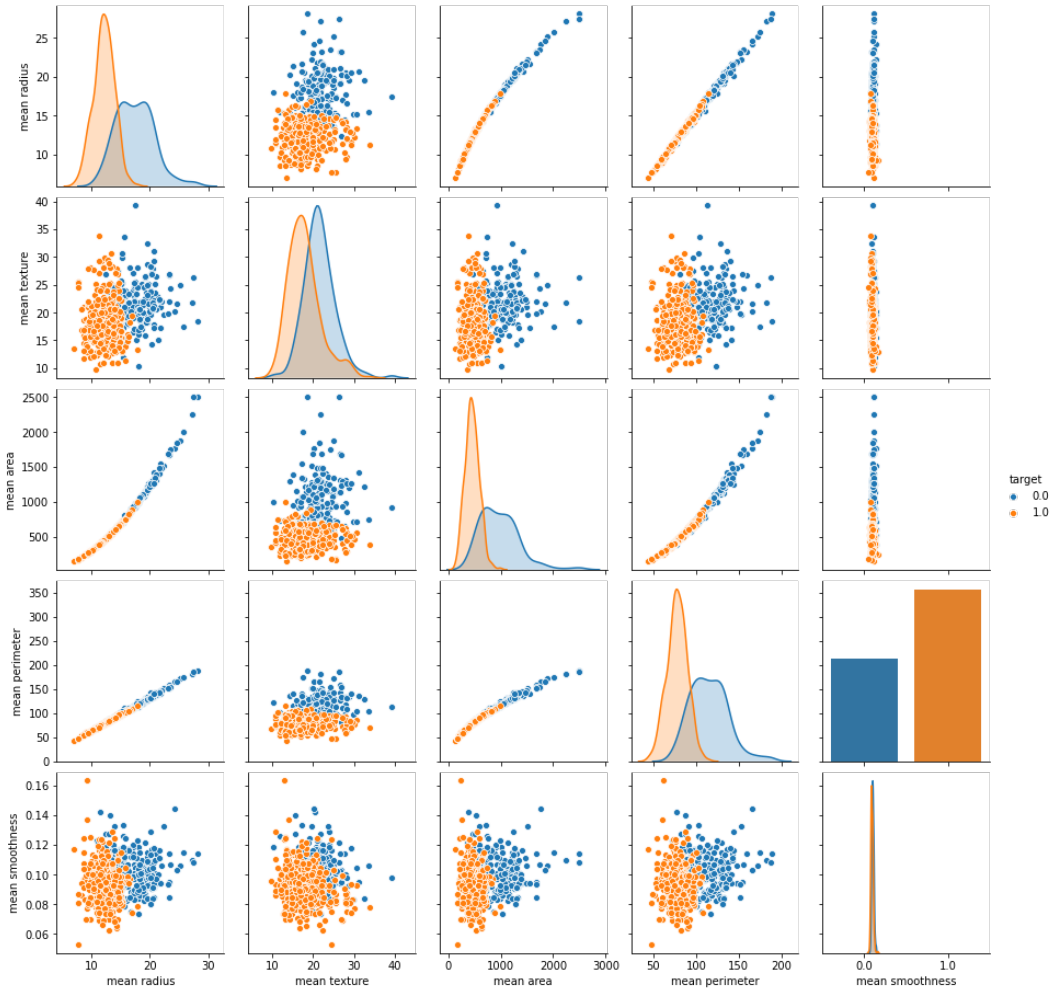
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	m fra dimen
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05

5 rows × 31 columns

Visualizar os dados com a Lib MatPlot

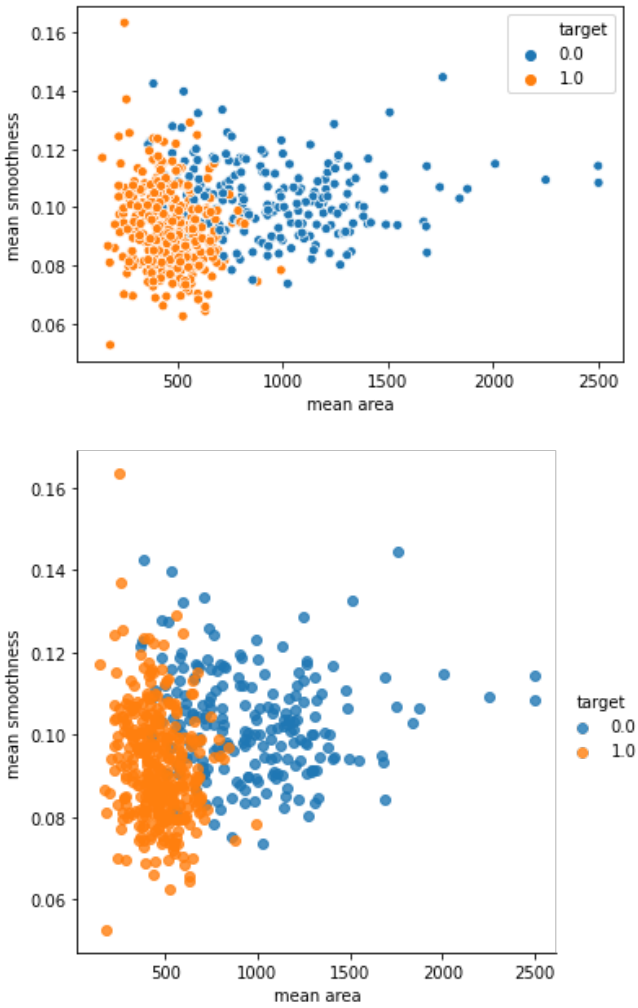
In [17]:
```
sns.pairplot(df_cancer, hue = 'target', vars = ['mean radius', 'mean texture
', 'mean area', 'mean perimeter', 'mean smoothness'] )
sns.countplot(df_cancer['target'], label = "Count")
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4d5ee6feb8>



In [13]:
```
sns.scatterplot(x = 'mean area', y = 'mean smoothness', hue = 'target', data
= df_cancer)
sns.lmplot('mean area', 'mean smoothness', hue ='target', data = df_cancer,
fit_reg=False)
```

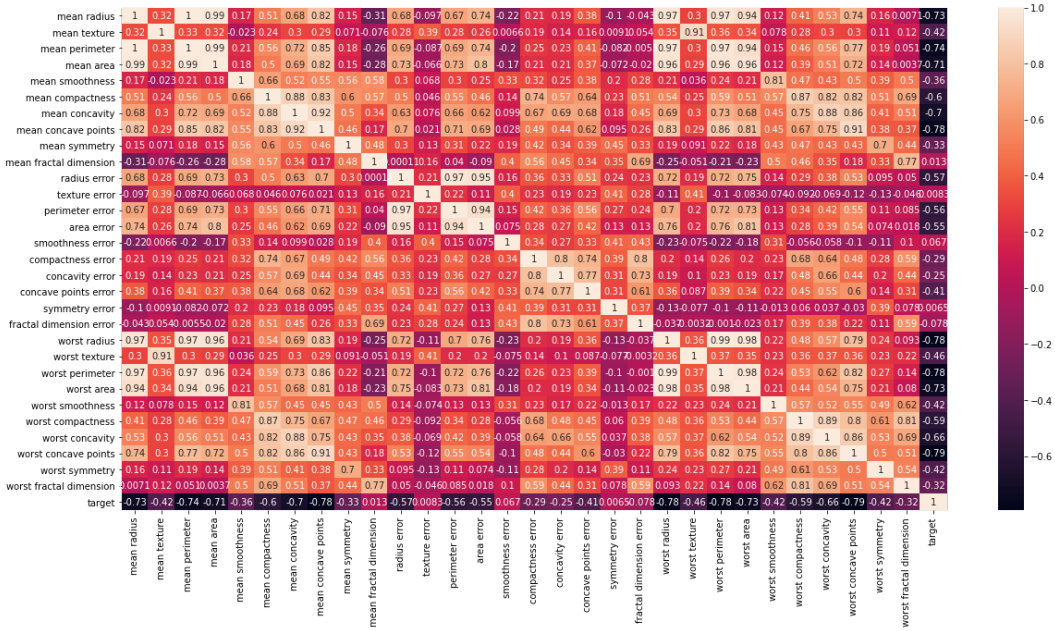
Out[13]: <seaborn.axisgrid.FacetGrid at 0x7f4d647f4470>



Checando a correlação entre as variaveis

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

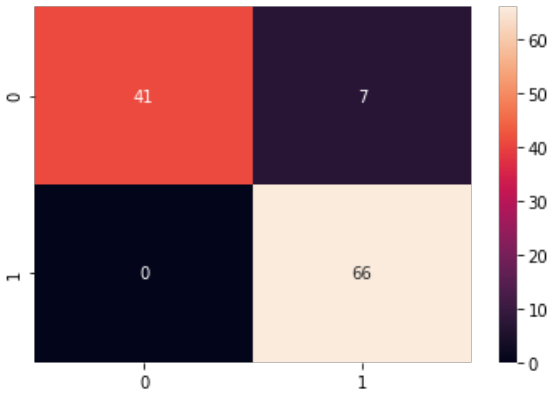
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4d5e943d68>



Treinando e avaliando o modelo

In [23]:
```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

X = df_cancer.drop(['target'],axis=1)
y = df_cancer['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20,
random_state=5)

X_train.shape
X_test.shape
y_train.shape
y_test.shape

svc_model = SVC()
svc_model.fit(X_train, y_train)
# Avaliando o modelo
y_predict = svc_model.predict(X_test)
cm = confusion_matrix(y_test, y_predict)

sns.heatmap(cm, annot=True)

print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.85      0.92        48
         1.0       0.90      1.00      0.95        66

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



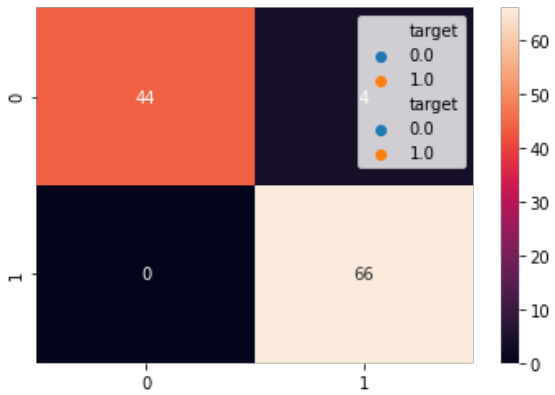
1º Melhoramento do Modelo

Realizando a Normalização de dados através do dimensionamento de recursos (Tipo de normalização baseada em Uni) que classifica os valores dentro do intervalo [0,1]. Tem por base a seguinte equação:

```
X '= (X - Xmin) / (Xmáx - Xmin)
```

In [26]:
```
min_train = X_train.min()
range_train = (X_train - min_train).max()
X_train_scaled = (X_train - min_train)/range_train

sns.scatterplot(x = X_train['mean area'], y = X_train['mean smoothness'], hu
e = y_train)
sns.scatterplot(x = X_train_scaled['mean area'], y = X_train_scaled['mean sm
oothness'], hue = y_train)

min_test = X_test.min()
range_test = (X_test - min_test).max()
X_test_scaled = (X_test - min_test)/range_test

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")

print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.92      0.96        48
         1.0       0.94      1.00      0.97        66

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



2º Melhoramento do Modelo

- Parâmetro C: controla o trade-off entre a classificação de pontos de treinamento corretamente e com um limite de decisão suave:
  - O pequeno C (solto) reduz o custo (penalidade) da classificação incorreta (margem suave)
  - Grande C (estrito) eleva o custo da classificação incorreta (margem bruta), forçando o modelo para explicar os dados de entrada mais restritos e potencialmente sobrepostos
- Parâmetro gama: controla até que ponto a influência de um único conjunto de treinamento atinge
  - Grande gama: alcance próximo (pontos de dados mais próximos têm alto peso)
  - Gama pequena: longo alcance (mais solução de generalização)

In [30]:
```
from sklearn.model_selection import GridSearchCV
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001], 'kerne
l': ['rbf']}

grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=4)
grid.fit(X_train_scaled,y_train)

grid.best_params_
grid.best_estimator_

grid_predictions = grid.predict(X_test_scaled)
cm = confusion_matrix(y_test, grid_predictions)
```

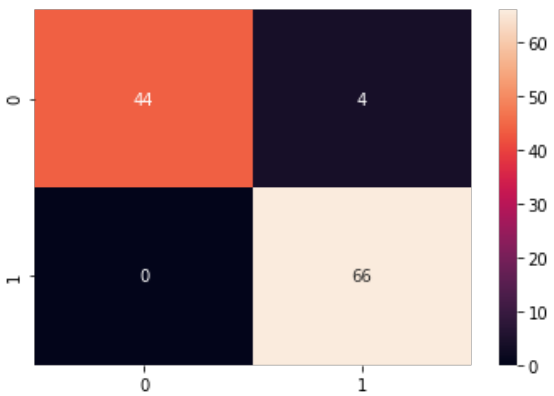
```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
[CV] C=0.1, gamma=1, kernel=rbf .....................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=0.1, gamma=1, kernel=rbf .....................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=0.1, gamma=1, kernel=rbf .....................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.912, total=   0.0s
[CV] C=0.1, gamma=1, kernel=rbf .....................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=0.1, gamma=1, kernel=rbf .....................................
[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.934, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ...................................
[CV] ......... C=0.1, gamma=0.1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ...................................
[CV] ......... C=0.1, gamma=0.1, kernel=rbf, score=0.901, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ...................................
[CV] ......... C=0.1, gamma=0.1, kernel=rbf, score=0.890, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ...................................
[CV] ......... C=0.1, gamma=0.1, kernel=rbf, score=0.923, total=   0.0s
[CV] C=0.1, gamma=0.1, kernel=rbf ...................................
[CV] ......... C=0.1, gamma=0.1, kernel=rbf, score=0.868, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ..................................
[CV] ........ C=0.1, gamma=0.01, kernel=rbf, score=0.648, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ..................................
[CV] ........ C=0.1, gamma=0.01, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ..................................

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[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:    0.1s remaining:    0.
0s
```

```
[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ..................................
[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.01, kernel=rbf ..................................
[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf .................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.648, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf .................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf .................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf .................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=0.1, gamma=0.001, kernel=rbf .................................
[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf .......................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf .......................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf .......................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf .......................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=1, gamma=1, kernel=rbf .......................................
[CV] ............ C=1, gamma=1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf .....................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.989, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf .....................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf .....................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.923, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf .....................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=1, gamma=0.1, kernel=rbf .....................................
[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.934, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.945, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.901, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.879, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.923, total=   0.0s
[CV] C=1, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.868, total=   0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.648, total=   0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=1, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.637, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=10, gamma=1, kernel=rbf ......................................
[CV] ........... C=10, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf ....................................
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=1.000, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf ....................................
```

```
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf .....................................
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.967, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf .....................................
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.989, total=   0.0s
[CV] C=10, gamma=0.1, kernel=rbf .....................................
[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=10, gamma=0.01, kernel=rbf, score=0.989, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=10, gamma=0.01, kernel=rbf, score=0.945, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=10, gamma=0.01, kernel=rbf, score=0.923, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=10, gamma=0.01, kernel=rbf, score=0.967, total=   0.0s
[CV] C=10, gamma=0.01, kernel=rbf ....................................
[CV] ......... C=10, gamma=0.01, kernel=rbf, score=0.934, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=10, gamma=0.001, kernel=rbf, score=0.945, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=10, gamma=0.001, kernel=rbf, score=0.901, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=10, gamma=0.001, kernel=rbf, score=0.879, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=10, gamma=0.001, kernel=rbf, score=0.923, total=   0.0s
[CV] C=10, gamma=0.001, kernel=rbf ...................................
[CV] ........ C=10, gamma=0.001, kernel=rbf, score=0.879, total=   0.0s
[CV] C=100, gamma=1, kernel=rbf ......................................
[CV] .......... C=100, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=100, gamma=1, kernel=rbf ......................................
[CV] .......... C=100, gamma=1, kernel=rbf, score=0.956, total=   0.0s
[CV] C=100, gamma=1, kernel=rbf ......................................
[CV] .......... C=100, gamma=1, kernel=rbf, score=0.945, total=   0.0s
[CV] C=100, gamma=1, kernel=rbf ......................................
[Parallel(n_jobs=1)]: Done  80 out of  80 | elapsed:    1.3s finished
```

In [31]:
```
sns.heatmap(cm, annot=True)

print(classification_report(y_test,grid_predictions))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.92      0.96        48
         1.0       0.94      1.00      0.97        66

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



In [ ]: