Support Vector Machines

August 28, 2023

---

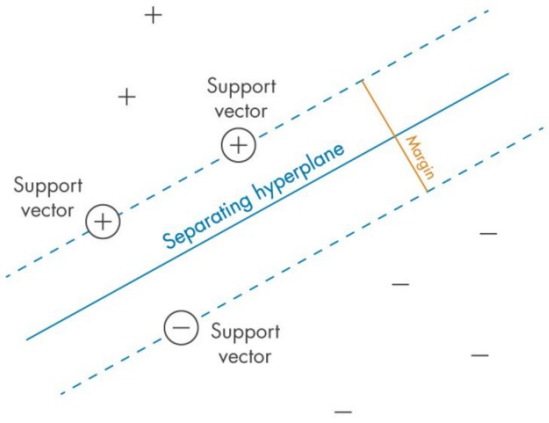
1 Support Vector Machines with Python

1.1 What is Support Vector Machine (SVM)?

Support vector machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression tasks. At times, SVM for classification is termed as support vector classification (SVC) and SVM for regression is termed as support vector regression (SVR). In this post, we will learn about SVM classifier.

The main idea behind SVM classifier is to find a hyperplane that maximally separates the data points of different classes. In other words, we are looking for the largest margin between the data points belonging to two classes. The hyperplane is selected such that it maximizes the distance between the hyperplane and the closest data points from each class, which are called support vectors. These points have a direct impact on the position and orientation of the hyperplane.

The rationale behind having decision boundaries with large margins is that they tend to have a lower generalization error, whereas models with small margins are more prone to overfitting. Hence, SVM classifier is also termed as maximum margin classifier, meaning that it finds the line or hyperplane that has the largest distance to the nearest training data points (support vectors) of any class. The following picture represents the same.



Given labeled training data (for supervised learning), the SVM classification algorithm outputs an optimal hyperplane which categorizes new data examples into different classes. This hyperplane is then used to make predictions on new data points.

Let's take an example to understand Support Vector Machine concepts in a better manner. Say you have been asked to predict whether a customer will churn or not and you have all their past transaction records as well as demographic information. After exploring the data, you've found that there's not much difference between the average transaction amount of customers who churned and those who didn't. You also found that most of the customers who churned live relatively far from the city center. Based on these findings, you decided to use Support Vector Machine classification algorithm to build your prediction model. Model trained using SVM classification algorithm will be able to classify the customers as high risk (churned) or otherwise.

1.2 Three hyperparameters that results in different SVM models

There are some key concepts that are important to understand when working with SVMs. First, the data points that are closest to the hyperplane are called support vectors. Second, when the data is not linearly separable, SVMs use a technique called kernel trick, which maps the original features into a higher-dimensional feature space where the data is linearly separable. In this new feature space, we can draw a hyperplane that separates the two classes. Third, when working with Python SKlearn SVC algorithm, there are three hyperparameters that results in different SVM models (hypothesis): C, gamma and kernel function.

1.2.1 C hyperparameters

- C is a regularization parameter that controls overfitting. It balances maximizing the margin and minimizing training error.
- Smaller C values lead to simpler models with larger margins. This can cause underfitting but the model generalizes better.
- Larger C values lead to more complex models with better training accuracy. This can cause overfitting and poor generalization.
- The choice of C depends on the problem. Use smaller C for noisy data where generalization is more important. Use larger C for clean data where training accuracy is critical.
- Start with a smaller C and gradually increase until desired level of accuracy is reached. But don't make C too large or you'll overfit.
- C controls the tradeoff between overfitting and underfitting. The optimal C minimizes overfitting while maximizing accuracy.
- Finding the right C requires experimenting with different values and evaluating performance on training and validation sets.
- The goal is to pick a C value that results in good performance on unseen data, not just training data. So balance training accuracy vs generalization.

1.2.2 Gamma hyperparameters

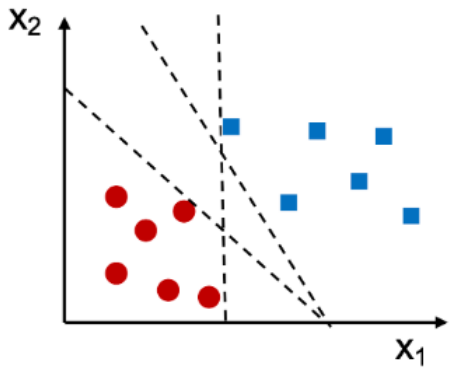
When training a support vector machine (SVM) model using Sklearn SVC algorithm, the gamma hyperparameter can take on two special values: 'scale', 'auto' or 'float'. When gamma is set to 'scale', it means that the value of gamma is calculated as 1 / (number of features * variance in data). This ensures that all features are given equal importance in the model and produces consistent results no matter how many features are used.

1.2.3 Kernel hyperparameter

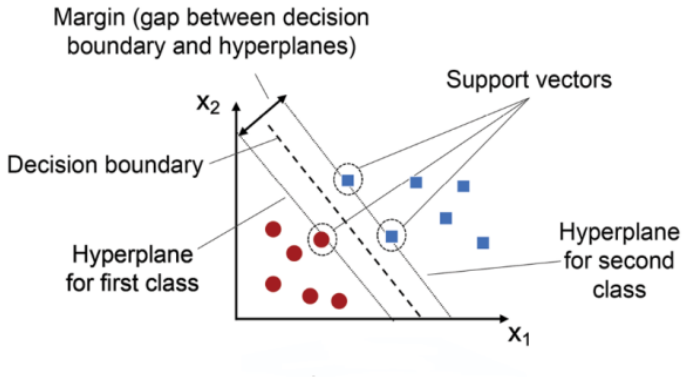
SVM kernel is a mathematical function that is used to map the data points from one space into another, usually higher dimensional space. When training a support vector machine (SVM) model using Sklearn SVC algorithm, the kernel hyperparameter can take on several values: 'linear', 'poly', 'rbf' and 'sigmoid'.

- **When kernel is set to 'linear'**, it means that the model will use a linear boundary for classification and regression. This is the simplest type of SVM and works best when data are linearly separable.
- **When kernel is set to 'poly'**, it means that the model will use polynomial functions of degree higher than 1 for classification. This type of SVM is more suitable for complex non-linear datasets.
- **When kernel is set to 'rbf'**, it means that the model will use radial basis funcitons for classification or regression. RBF kernels are capable of dealing with complex multi-class datasets and have good generalization performance with noisy data points.
- **When kernel is set to 'sigmoid'**, it means that the model will apply sigmoid functions instead of RBFs for classification or regression tasks. Sigmoid kernels tend to be less sensitive than RBFs with respect to outliers in data but may not generalize as well unless their parameters are tuned properly.

As an example, let's say we have a dataset with two features (x1 and x2) and two classes (0 and 1). We can visualize this data by plotting it in a two-dimensional space, with each point colored according to its class label. Look at the diagram below.



In the above case, we can see that there are different straight lines that can perfectly separate the two classes. However, we can still find a decision boundary that does a pretty good job. This boundary is generated by Support Vector Machine algorithm. Using SVM algorithm, as mentioned above, training the model represents finding the hyperplane (dashed line in the picture below) which separates the data belonging to two different classes by maximum or largest margin. And, the points closest to this hyperplane are called support vectors. Note this in the diagram given below.



The blue square points represent one class and the red dots represent another class. The black line is the decision boundary learned by an SVM. As you can see, the SVM has placed the boundary in such a way as to maximize the margin between the two classes.

1.3 Import Libraries

[89]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

1.4 Get the Data

We'll use the built in breast cancer dataset from Scikit Learn. We can get with the load function:

[90]:
```
from sklearn.datasets import load_breast_cancer
```

[91]:
```
cancer = load_breast_cancer()
```

The data set is presented in a dictionary form:

[92]:
```
cancer.keys()
```

[92]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
'filename', 'data_module'])
```

We can grab information and arrays out of this dictionary to set up our data frame and understanding of the features:

[93]:
```
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
        worst/largest values) of these features were computed for each image,
        resulting in 30 features. For instance, field 0 is Mean Radius, field
        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 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.
```

[94]:
```
cancer['feature_names']
```

[94]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

1.5 Set up DataFrame

[95]:
```
df_feat = pd.DataFrame(cancer['data'],columns=cancer['feature_names'])
df_feat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 30 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
```

```
 4   mean smoothness          569 non-null    float64
 5   mean compactness         569 non-null    float64
 6   mean concavity           569 non-null    float64
 7   mean concave points      569 non-null    float64
 8   mean symmetry            569 non-null    float64
 9   mean fractal dimension   569 non-null    float64
 10  radius error             569 non-null    float64
 11  texture error            569 non-null    float64
 12  perimeter error          569 non-null    float64
 13  area error               569 non-null    float64
 14  smoothness error         569 non-null    float64
 15  compactness error        569 non-null    float64
 16  concavity error          569 non-null    float64
 17  concave points error     569 non-null    float64
 18  symmetry error           569 non-null    float64
 19  fractal dimension error  569 non-null    float64
 20  worst radius             569 non-null    float64
 21  worst texture            569 non-null    float64
 22  worst perimeter          569 non-null    float64
 23  worst area               569 non-null    float64
 24  worst smoothness         569 non-null    float64
 25  worst compactness        569 non-null    float64
 26  worst concavity          569 non-null    float64
 27  worst concave points     569 non-null    float64
 28  worst symmetry           569 non-null    float64
 29  worst fractal dimension  569 non-null    float64
dtypes: float64(30)
memory usage: 133.5 KB
```

[96]:
```
df_feat
```

[96]:

	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	
..	…	…	…	…	…	
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	\
0	0.27760	0.30010	0.14710	0.2419	
1	0.07864	0.08690	0.07017	0.1812	

```
2            0.15990         0.19740              0.12790         0.2069
3            0.28390         0.24140              0.10520         0.2597
4            0.13280         0.19800              0.10430         0.1809
..               …               …                    …              …
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 radius  worst texture  \
0                   0.07871  …        25.380          17.33
1                   0.05667  …        24.990          23.41
2                   0.05999  …        23.570          25.53
3                   0.09744  …        14.910          26.50
4                   0.05883  …        22.540          16.67
..                      …  …           …              …
564                 0.05623  …        25.450          26.40
565                 0.05533  …        23.690          38.25
566                 0.05648  …        18.980          34.12
567                 0.07016  …        25.740          39.42
568                 0.05884  …         9.456          30.37

     worst perimeter  worst area  worst smoothness  worst compactness  \
0              184.60      2019.0           0.16220            0.66560
1              158.80      1956.0           0.12380            0.18660
2              152.50      1709.0           0.14440            0.42450
3               98.87       567.7           0.20980            0.86630
4              152.20      1575.0           0.13740            0.20500
..                …           …                …                 …
564            166.10      2027.0           0.14100            0.21130
565            155.00      1731.0           0.11660            0.19220
566            126.70      1124.0           0.11390            0.30940
567            184.60      1821.0           0.16500            0.86810
568             59.16       268.6           0.08996            0.06444

     worst concavity  worst concave points  worst symmetry  \
0             0.7119                0.2654          0.4601
1             0.2416                0.1860          0.2750
2             0.4504                0.2430          0.3613
3             0.6869                0.2575          0.6638
4             0.4000                0.1625          0.2364
..               …                     …               …
564           0.4107                0.2216          0.2060
565           0.3215                0.1628          0.2572
566           0.3403                0.1418          0.2218
567           0.9387                0.2650          0.4087
```

```
568           0.0000                0.0000          0.2871

     worst fractal dimension
0                    0.11890
1                    0.08902
2                    0.08758
3                    0.17300
4                    0.07678
..                       …
564                  0.07115
565                  0.06637
566                  0.07820
567                  0.12400
568                  0.07039

[569 rows x 30 columns]
```

[97]:
```
cancer['target']
```

[97]:
```
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, 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, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       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, 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])
```

[98]:
```
df_target = pd.DataFrame(cancer['target'],columns=['Cancer'])
```

[99]:
```
df_target
```

[99]:
```
     Cancer
0         0
1         0
2         0
3         0
4         0
..      …
564       0
565       0
566       0
567       0
568       1

[569 rows x 1 columns]
```

Now let's actually check out the dataframe!

[100]:
```
df_feat.head()
```

[100]:
```
   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 radius  worst texture  worst perimeter  \
0                 0.07871  …         25.38          17.33           184.60
1                 0.05667  …         24.99          23.41           158.80
2                 0.05999  …         23.57          25.53           152.50
3                 0.09744  …         14.91          26.50            98.87
4                 0.05883  …         22.54          16.67           152.20

   worst area  worst smoothness  worst compactness  worst concavity  \
0      2019.0            0.1622             0.6656           0.7119
1      1956.0            0.1238             0.1866           0.2416
2      1709.0            0.1444             0.4245           0.4504
3       567.7            0.2098             0.8663           0.6869
4      1575.0            0.1374             0.2050           0.4000
```

```
   worst concave points  worst symmetry  worst fractal dimension
0                0.2654          0.4601                  0.11890
1                0.1860          0.2750                  0.08902
2                0.2430          0.3613                  0.08758
3                0.2575          0.6638                  0.17300
4                0.1625          0.2364                  0.07678

[5 rows x 30 columns]
```

2 Exploratory Data Analysis

We'll skip the Data Viz part for this lecture since there are so many features that are hard to interpret if you don't have domain knowledge of cancer or tumor cells. In your project you will have more to visualize for the data.

2.1 Train Test Split

[101]:
```
from sklearn.model_selection import train_test_split
```

[102]:
```
X= df_feat
y= cancer['target']
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.30,␣
 ↪random_state=101)
```

3 Train the Support Vector Classifier

[103]:
```
from sklearn.svm import SVC
```

[104]:
```
model = SVC()
```

[105]:
```
model.fit(X_train,y_train)
```

[105]:
```
SVC()
```

3.1 Predictions and Evaluations

Now let's predict using the trained model.

[106]:
```
predictions = model.predict(X_test)
```

[107]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

[108]:
```
print(confusion_matrix(y_test,predictions))
```

```
[[ 56  10]
 [  3 102]]
```

[109]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.95      0.85      0.90        66
           1       0.91      0.97      0.94       105

    accuracy                           0.92       171
   macro avg       0.93      0.91      0.92       171
weighted avg       0.93      0.92      0.92       171
```

We can adjust different parameters so that our model performs better (it may also help to normalize the data).

We can search for parameters using a GridSearch!

4 Gridsearch

Finding the right parameters (like what C or gamma values to use) is a tricky task! But luckily, we can be a little lazy and just try a bunch of combinations and see what works best! This idea of creating a 'grid' of parameters and just trying out all the possible combinations is called a Gridsearch, this method is common enough that Scikit-learn has this functionality built in with GridSearchCV! The CV stands for cross-validation which is the

GridSearchCV takes a dictionary that describes the parameters that should be tried and a model to train. The grid of parameters is defined as a dictionary, where the keys are the parameters and the values are the settings to be tested.

[119]:
```
param_grid = {'C': [0.1,1,5, 10, 50, 100, 1000], 'gamma': [10,1,0.1,0.01,0.
↪001,0.0001], 'kernel': ['rbf','linear']}
```

[120]:
```
from sklearn.model_selection import GridSearchCV
```

One of the great things about GridSearchCV is that it is a meta-estimator. It takes an estimator like SVC, and creates a new estimator, that behaves exactly the same - in this case, like a classifier. You should add refit=True and choose verbose to whatever number you want, higher the number, the more verbose (verbose just means the text output describing the process).

[121]:
```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=5)
```

What fit does is a bit more involved then usual. First, it runs the same loop with cross-validation, to find the best parameter combination. Once it has the best combination, it runs fit again on all data passed to fit (without cross-validation), to built a single new model using the best parameter setting.

[122]:
```
# May take awhile!
grid.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 84 candidates, totalling 420 fits
[CV 1/5] END …C=0.1, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
```

```
[CV 2/5] END …C=0.1, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=10, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=0.1, gamma=10, kernel=linear;, score=0.950 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=10, kernel=linear;, score=0.925 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=10, kernel=linear;, score=0.988 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=10, kernel=linear;, score=0.937 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=10, kernel=linear;, score=0.962 total time=   0.0s
[CV 1/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=0.1, gamma=1, kernel=linear;, score=0.950 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=1, kernel=linear;, score=0.925 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=1, kernel=linear;, score=0.988 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=1, kernel=linear;, score=0.937 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=1, kernel=linear;, score=0.962 total time=   0.0s
[CV 1/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.950 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.925 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.988 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.937 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=0.1, kernel=linear;, score=0.962 total time=   0.0s
[CV 1/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.637 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=0.625 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.950 total time=   0.0s
[CV 2/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.925 total time=   0.0s
[CV 3/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.988 total time=   0.0s
[CV 4/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.937 total time=   0.0s
[CV 5/5] END ..C=0.1, gamma=0.01, kernel=linear;, score=0.962 total time=   0.0s
[CV 1/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.637 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=0.625 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=0.001, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.950 total time=   0.0s
[CV 2/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.925 total time=   0.0s
[CV 3/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.988 total time=   0.0s
[CV 4/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.937 total time=   0.0s
```

```
[CV 5/5] END .C=0.1, gamma=0.001, kernel=linear;, score=0.962 total time=   0.0s
[CV 1/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.887 total time=   0.0s
[CV 2/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.938 total time=   0.0s
[CV 3/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.963 total time=   0.0s
[CV 4/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.962 total time=   0.0s
[CV 5/5] END …C=0.1, gamma=0.0001, kernel=rbf;, score=0.886 total time=   0.0s
[CV 1/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.950 total time=   0.0s
[CV 2/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.925 total time=   0.0s
[CV 3/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.988 total time=   0.0s
[CV 4/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.937 total time=   0.0s
[CV 5/5] END C=0.1, gamma=0.0001, kernel=linear;, score=0.962 total time=   0.0s
[CV 1/5] END …C=1, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=1, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=1, gamma=10, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=1, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=1, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=1, gamma=10, kernel=linear;, score=0.950 total time=   0.2s
[CV 2/5] END …C=1, gamma=10, kernel=linear;, score=0.938 total time=   0.2s
[CV 3/5] END …C=1, gamma=10, kernel=linear;, score=1.000 total time=   0.3s
[CV 4/5] END …C=1, gamma=10, kernel=linear;, score=0.937 total time=   0.2s
[CV 5/5] END …C=1, gamma=10, kernel=linear;, score=0.987 total time=   0.1s
[CV 1/5] END …C=1, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=1, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=1, gamma=1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=1, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=1, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=1, gamma=1, kernel=linear;, score=0.950 total time=   0.2s
[CV 2/5] END …C=1, gamma=1, kernel=linear;, score=0.938 total time=   0.1s
[CV 3/5] END …C=1, gamma=1, kernel=linear;, score=1.000 total time=   0.2s
[CV 4/5] END …C=1, gamma=1, kernel=linear;, score=0.937 total time=   0.2s
[CV 5/5] END …C=1, gamma=1, kernel=linear;, score=0.987 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=1, gamma=0.1, kernel=linear;, score=0.950 total time=   0.2s
[CV 2/5] END …C=1, gamma=0.1, kernel=linear;, score=0.938 total time=   0.1s
[CV 3/5] END …C=1, gamma=0.1, kernel=linear;, score=1.000 total time=   0.2s
[CV 4/5] END …C=1, gamma=0.1, kernel=linear;, score=0.937 total time=   0.2s
[CV 5/5] END …C=1, gamma=0.1, kernel=linear;, score=0.987 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=1, gamma=0.01, kernel=linear;, score=0.950 total time=   0.2s
[CV 2/5] END …C=1, gamma=0.01, kernel=linear;, score=0.938 total time=   0.1s
```

```
[CV 3/5] END …C=1, gamma=0.01, kernel=linear;, score=1.000 total time=   0.2s
[CV 4/5] END …C=1, gamma=0.01, kernel=linear;, score=0.937 total time=   0.2s
[CV 5/5] END …C=1, gamma=0.01, kernel=linear;, score=0.987 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.900 total time=   0.0s
[CV 2/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.925 total time=   0.0s
[CV 4/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.962 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 1/5] END …C=1, gamma=0.001, kernel=linear;, score=0.950 total time=   0.2s
[CV 2/5] END …C=1, gamma=0.001, kernel=linear;, score=0.938 total time=   0.1s
[CV 3/5] END …C=1, gamma=0.001, kernel=linear;, score=1.000 total time=   0.2s
[CV 4/5] END …C=1, gamma=0.001, kernel=linear;, score=0.937 total time=   0.2s
[CV 5/5] END …C=1, gamma=0.001, kernel=linear;, score=0.987 total time=   0.1s
[CV 1/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 2/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.950 total time=   0.0s
[CV 3/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.975 total time=   0.0s
[CV 4/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.962 total time=   0.0s
[CV 5/5] END …C=1, gamma=0.0001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 1/5] END ..C=1, gamma=0.0001, kernel=linear;, score=0.950 total time=   0.2s
[CV 2/5] END ..C=1, gamma=0.0001, kernel=linear;, score=0.938 total time=   0.1s
[CV 3/5] END ..C=1, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.2s
[CV 4/5] END ..C=1, gamma=0.0001, kernel=linear;, score=0.937 total time=   0.2s
[CV 5/5] END ..C=1, gamma=0.0001, kernel=linear;, score=0.987 total time=   0.1s
[CV 1/5] END …C=5, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=5, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=5, gamma=10, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=5, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=5, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=5, gamma=10, kernel=linear;, score=0.950 total time=   1.1s
[CV 2/5] END …C=5, gamma=10, kernel=linear;, score=0.938 total time=   0.4s
[CV 3/5] END …C=5, gamma=10, kernel=linear;, score=1.000 total time=   1.0s
[CV 4/5] END …C=5, gamma=10, kernel=linear;, score=0.962 total time=   1.2s
[CV 5/5] END …C=5, gamma=10, kernel=linear;, score=0.987 total time=   1.8s
[CV 1/5] END …C=5, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=5, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=5, gamma=1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=5, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=5, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=5, gamma=1, kernel=linear;, score=0.950 total time=   1.1s
[CV 2/5] END …C=5, gamma=1, kernel=linear;, score=0.938 total time=   0.4s
[CV 3/5] END …C=5, gamma=1, kernel=linear;, score=1.000 total time=   1.0s
[CV 4/5] END …C=5, gamma=1, kernel=linear;, score=0.962 total time=   1.2s
[CV 5/5] END …C=5, gamma=1, kernel=linear;, score=0.987 total time=   1.8s
[CV 1/5] END …C=5, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=5, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=5, gamma=0.1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=5, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=5, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
```

```
[CV 1/5] END …C=5, gamma=0.1, kernel=linear;, score=0.950 total time=   1.1s
[CV 2/5] END …C=5, gamma=0.1, kernel=linear;, score=0.938 total time=   0.4s
[CV 3/5] END …C=5, gamma=0.1, kernel=linear;, score=1.000 total time=   1.0s
[CV 4/5] END …C=5, gamma=0.1, kernel=linear;, score=0.962 total time=   1.2s
[CV 5/5] END …C=5, gamma=0.1, kernel=linear;, score=0.987 total time=   1.8s
[CV 1/5] END …C=5, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=5, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=5, gamma=0.01, kernel=rbf;, score=0.613 total time=   0.0s
[CV 4/5] END …C=5, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=5, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=5, gamma=0.01, kernel=linear;, score=0.950 total time=   1.1s
[CV 2/5] END …C=5, gamma=0.01, kernel=linear;, score=0.938 total time=   0.4s
[CV 3/5] END …C=5, gamma=0.01, kernel=linear;, score=1.000 total time=   1.0s
[CV 4/5] END …C=5, gamma=0.01, kernel=linear;, score=0.962 total time=   1.2s
[CV 5/5] END …C=5, gamma=0.01, kernel=linear;, score=0.987 total time=   1.8s
[CV 1/5] END …C=5, gamma=0.001, kernel=rbf;, score=0.875 total time=   0.0s
[CV 2/5] END …C=5, gamma=0.001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=5, gamma=0.001, kernel=rbf;, score=0.900 total time=   0.0s
[CV 4/5] END …C=5, gamma=0.001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 5/5] END …C=5, gamma=0.001, kernel=rbf;, score=0.924 total time=   0.0s
[CV 1/5] END …C=5, gamma=0.001, kernel=linear;, score=0.950 total time=   1.2s
[CV 2/5] END …C=5, gamma=0.001, kernel=linear;, score=0.938 total time=   0.4s
[CV 3/5] END …C=5, gamma=0.001, kernel=linear;, score=1.000 total time=   1.0s
[CV 4/5] END …C=5, gamma=0.001, kernel=linear;, score=0.962 total time=   1.2s
[CV 5/5] END …C=5, gamma=0.001, kernel=linear;, score=0.987 total time=   1.8s
[CV 1/5] END …C=5, gamma=0.0001, kernel=rbf;, score=0.925 total time=   0.0s
[CV 2/5] END …C=5, gamma=0.0001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=5, gamma=0.0001, kernel=rbf;, score=0.975 total time=   0.0s
[CV 4/5] END …C=5, gamma=0.0001, kernel=rbf;, score=0.962 total time=   0.0s
[CV 5/5] END …C=5, gamma=0.0001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 1/5] END ..C=5, gamma=0.0001, kernel=linear;, score=0.950 total time=   1.1s
[CV 2/5] END ..C=5, gamma=0.0001, kernel=linear;, score=0.938 total time=   0.4s
[CV 3/5] END ..C=5, gamma=0.0001, kernel=linear;, score=1.000 total time=   1.0s
[CV 4/5] END ..C=5, gamma=0.0001, kernel=linear;, score=0.962 total time=   1.2s
[CV 5/5] END ..C=5, gamma=0.0001, kernel=linear;, score=0.987 total time=   1.8s
[CV 1/5] END …C=10, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=10, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=10, gamma=10, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=10, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=10, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=10, gamma=10, kernel=linear;, score=0.938 total time=   1.7s
[CV 2/5] END …C=10, gamma=10, kernel=linear;, score=0.938 total time=   0.5s
[CV 3/5] END …C=10, gamma=10, kernel=linear;, score=1.000 total time=   0.8s
[CV 4/5] END …C=10, gamma=10, kernel=linear;, score=0.949 total time=   2.2s
[CV 5/5] END …C=10, gamma=10, kernel=linear;, score=0.987 total time=   1.6s
[CV 1/5] END …C=10, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=10, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=10, gamma=1, kernel=rbf;, score=0.625 total time=   0.0s
```

```
[CV 4/5] END …C=10, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=10, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=10, gamma=1, kernel=linear;, score=0.938 total time=   1.7s
[CV 2/5] END …C=10, gamma=1, kernel=linear;, score=0.938 total time=   0.5s
[CV 3/5] END …C=10, gamma=1, kernel=linear;, score=1.000 total time=   0.8s
[CV 4/5] END …C=10, gamma=1, kernel=linear;, score=0.949 total time=   2.0s
[CV 5/5] END …C=10, gamma=1, kernel=linear;, score=0.987 total time=   1.6s
[CV 1/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=10, gamma=0.1, kernel=linear;, score=0.938 total time=   1.7s
[CV 2/5] END …C=10, gamma=0.1, kernel=linear;, score=0.938 total time=   0.5s
[CV 3/5] END …C=10, gamma=0.1, kernel=linear;, score=1.000 total time=   0.8s
[CV 4/5] END …C=10, gamma=0.1, kernel=linear;, score=0.949 total time=   2.0s
[CV 5/5] END …C=10, gamma=0.1, kernel=linear;, score=0.987 total time=   1.7s
[CV 1/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.613 total time=   0.0s
[CV 4/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=10, gamma=0.01, kernel=linear;, score=0.938 total time=   1.7s
[CV 2/5] END …C=10, gamma=0.01, kernel=linear;, score=0.938 total time=   0.5s
[CV 3/5] END …C=10, gamma=0.01, kernel=linear;, score=1.000 total time=   0.8s
[CV 4/5] END …C=10, gamma=0.01, kernel=linear;, score=0.949 total time=   2.1s
[CV 5/5] END …C=10, gamma=0.01, kernel=linear;, score=0.987 total time=   1.6s
[CV 1/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.887 total time=   0.0s
[CV 2/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.900 total time=   0.0s
[CV 4/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.001, kernel=rbf;, score=0.924 total time=   0.0s
[CV 1/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.938 total time=   1.7s
[CV 2/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.938 total time=   0.5s
[CV 3/5] END ..C=10, gamma=0.001, kernel=linear;, score=1.000 total time=   0.8s
[CV 4/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.949 total time=   2.0s
[CV 5/5] END ..C=10, gamma=0.001, kernel=linear;, score=0.987 total time=   1.7s
[CV 1/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.950 total time=   0.0s
[CV 2/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.975 total time=   0.0s
[CV 4/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.949 total time=   0.0s
[CV 5/5] END …C=10, gamma=0.0001, kernel=rbf;, score=0.949 total time=   0.0s
[CV 1/5] END .C=10, gamma=0.0001, kernel=linear;, score=0.938 total time=   1.7s
[CV 2/5] END .C=10, gamma=0.0001, kernel=linear;, score=0.938 total time=   0.5s
[CV 3/5] END .C=10, gamma=0.0001, kernel=linear;, score=1.000 total time=   0.8s
[CV 4/5] END .C=10, gamma=0.0001, kernel=linear;, score=0.949 total time=   2.0s
[CV 5/5] END .C=10, gamma=0.0001, kernel=linear;, score=0.987 total time=   1.6s
[CV 1/5] END …C=50, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
```

```
[CV 2/5] END …C=50, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=50, gamma=10, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=50, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=50, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=50, gamma=10, kernel=linear;, score=0.950 total time=   1.3s
[CV 2/5] END …C=50, gamma=10, kernel=linear;, score=0.950 total time=   0.4s
[CV 3/5] END …C=50, gamma=10, kernel=linear;, score=1.000 total time=   2.2s
[CV 4/5] END …C=50, gamma=10, kernel=linear;, score=0.962 total time=   5.3s
[CV 5/5] END …C=50, gamma=10, kernel=linear;, score=0.987 total time=   3.0s
[CV 1/5] END …C=50, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=50, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=50, gamma=1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=50, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=50, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=50, gamma=1, kernel=linear;, score=0.950 total time=   1.3s
[CV 2/5] END …C=50, gamma=1, kernel=linear;, score=0.950 total time=   0.4s
[CV 3/5] END …C=50, gamma=1, kernel=linear;, score=1.000 total time=   2.2s
[CV 4/5] END …C=50, gamma=1, kernel=linear;, score=0.962 total time=   5.2s
[CV 5/5] END …C=50, gamma=1, kernel=linear;, score=0.987 total time=   3.1s
[CV 1/5] END …C=50, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=50, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=50, gamma=0.1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=50, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=50, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=50, gamma=0.1, kernel=linear;, score=0.950 total time=   1.3s
[CV 2/5] END …C=50, gamma=0.1, kernel=linear;, score=0.950 total time=   0.4s
[CV 3/5] END …C=50, gamma=0.1, kernel=linear;, score=1.000 total time=   2.2s
[CV 4/5] END …C=50, gamma=0.1, kernel=linear;, score=0.962 total time=   5.3s
[CV 5/5] END …C=50, gamma=0.1, kernel=linear;, score=0.987 total time=   3.1s
[CV 1/5] END …C=50, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=50, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=50, gamma=0.01, kernel=rbf;, score=0.613 total time=   0.0s
[CV 4/5] END …C=50, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=50, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=50, gamma=0.01, kernel=linear;, score=0.950 total time=   1.3s
[CV 2/5] END …C=50, gamma=0.01, kernel=linear;, score=0.950 total time=   0.4s
[CV 3/5] END …C=50, gamma=0.01, kernel=linear;, score=1.000 total time=   2.2s
[CV 4/5] END …C=50, gamma=0.01, kernel=linear;, score=0.962 total time=   5.2s
[CV 5/5] END …C=50, gamma=0.01, kernel=linear;, score=0.987 total time=   3.0s
[CV 1/5] END …C=50, gamma=0.001, kernel=rbf;, score=0.887 total time=   0.0s
[CV 2/5] END …C=50, gamma=0.001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=50, gamma=0.001, kernel=rbf;, score=0.900 total time=   0.0s
[CV 4/5] END …C=50, gamma=0.001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 5/5] END …C=50, gamma=0.001, kernel=rbf;, score=0.924 total time=   0.0s
[CV 1/5] END ..C=50, gamma=0.001, kernel=linear;, score=0.950 total time=   1.3s
[CV 2/5] END ..C=50, gamma=0.001, kernel=linear;, score=0.950 total time=   0.4s
[CV 3/5] END ..C=50, gamma=0.001, kernel=linear;, score=1.000 total time=   2.2s
[CV 4/5] END ..C=50, gamma=0.001, kernel=linear;, score=0.962 total time=   5.3s
```

```
[CV 5/5] END ..C=50, gamma=0.001, kernel=linear;, score=0.987 total time=   3.1s
[CV 1/5] END …C=50, gamma=0.0001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 2/5] END …C=50, gamma=0.0001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=50, gamma=0.0001, kernel=rbf;, score=0.963 total time=   0.0s
[CV 4/5] END …C=50, gamma=0.0001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 5/5] END …C=50, gamma=0.0001, kernel=rbf;, score=0.949 total time=   0.0s
[CV 1/5] END .C=50, gamma=0.0001, kernel=linear;, score=0.950 total time=   1.3s
[CV 2/5] END .C=50, gamma=0.0001, kernel=linear;, score=0.950 total time=   0.4s
[CV 3/5] END .C=50, gamma=0.0001, kernel=linear;, score=1.000 total time=   2.2s
[CV 4/5] END .C=50, gamma=0.0001, kernel=linear;, score=0.962 total time=   5.3s
[CV 5/5] END .C=50, gamma=0.0001, kernel=linear;, score=0.987 total time=   3.1s
[CV 1/5] END …C=100, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=100, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=100, gamma=10, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=100, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=100, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=100, gamma=10, kernel=linear;, score=0.950 total time=   1.4s
[CV 2/5] END …C=100, gamma=10, kernel=linear;, score=0.950 total time=   0.5s
[CV 3/5] END …C=100, gamma=10, kernel=linear;, score=1.000 total time=   2.5s
[CV 4/5] END …C=100, gamma=10, kernel=linear;, score=0.949 total time=   4.9s
[CV 5/5] END …C=100, gamma=10, kernel=linear;, score=0.987 total time=   2.7s
[CV 1/5] END …C=100, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=100, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=100, gamma=1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=100, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=100, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=100, gamma=1, kernel=linear;, score=0.950 total time=   1.4s
[CV 2/5] END …C=100, gamma=1, kernel=linear;, score=0.950 total time=   0.5s
[CV 3/5] END …C=100, gamma=1, kernel=linear;, score=1.000 total time=   2.5s
[CV 4/5] END …C=100, gamma=1, kernel=linear;, score=0.949 total time=   4.9s
[CV 5/5] END …C=100, gamma=1, kernel=linear;, score=0.987 total time=   2.7s
[CV 1/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=100, gamma=0.1, kernel=linear;, score=0.950 total time=   1.4s
[CV 2/5] END …C=100, gamma=0.1, kernel=linear;, score=0.950 total time=   0.5s
[CV 3/5] END …C=100, gamma=0.1, kernel=linear;, score=1.000 total time=   2.6s
[CV 4/5] END …C=100, gamma=0.1, kernel=linear;, score=0.949 total time=   4.9s
[CV 5/5] END …C=100, gamma=0.1, kernel=linear;, score=0.987 total time=   2.7s
[CV 1/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.613 total time=   0.0s
[CV 4/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.950 total time=   1.3s
[CV 2/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.950 total time=   0.5s
```

```
[CV 3/5] END ..C=100, gamma=0.01, kernel=linear;, score=1.000 total time=   2.5s
[CV 4/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.949 total time=   4.9s
[CV 5/5] END ..C=100, gamma=0.01, kernel=linear;, score=0.987 total time=   2.7s
[CV 1/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.887 total time=   0.0s
[CV 2/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.900 total time=   0.0s
[CV 4/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.001, kernel=rbf;, score=0.924 total time=   0.0s
[CV 1/5] END .C=100, gamma=0.001, kernel=linear;, score=0.950 total time=   1.4s
[CV 2/5] END .C=100, gamma=0.001, kernel=linear;, score=0.950 total time=   0.5s
[CV 3/5] END .C=100, gamma=0.001, kernel=linear;, score=1.000 total time=   2.5s
[CV 4/5] END .C=100, gamma=0.001, kernel=linear;, score=0.949 total time=   5.0s
[CV 5/5] END .C=100, gamma=0.001, kernel=linear;, score=0.987 total time=   2.8s
[CV 1/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.925 total time=   0.0s
[CV 2/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.975 total time=   0.0s
[CV 4/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 5/5] END …C=100, gamma=0.0001, kernel=rbf;, score=0.949 total time=   0.0s
[CV 1/5] END C=100, gamma=0.0001, kernel=linear;, score=0.950 total time=   1.4s
[CV 2/5] END C=100, gamma=0.0001, kernel=linear;, score=0.950 total time=   0.5s
[CV 3/5] END C=100, gamma=0.0001, kernel=linear;, score=1.000 total time=   2.5s
[CV 4/5] END C=100, gamma=0.0001, kernel=linear;, score=0.949 total time=   4.9s
[CV 5/5] END C=100, gamma=0.0001, kernel=linear;, score=0.987 total time=   2.7s
[CV 1/5] END …C=1000, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=1000, gamma=10, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=1000, gamma=10, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=1000, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=1000, gamma=10, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=1000, gamma=10, kernel=linear;, score=0.950 total time=   3.9s
[CV 2/5] END …C=1000, gamma=10, kernel=linear;, score=0.950 total time=   0.9s
[CV 3/5] END …C=1000, gamma=10, kernel=linear;, score=1.000 total time=   1.2s
[CV 4/5] END …C=1000, gamma=10, kernel=linear;, score=0.949 total time=   4.5s
[CV 5/5] END …C=1000, gamma=10, kernel=linear;, score=0.987 total time=   3.8s
[CV 1/5] END …C=1000, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=1000, gamma=1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=1000, gamma=1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=1000, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=1000, gamma=1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END …C=1000, gamma=1, kernel=linear;, score=0.950 total time=   3.9s
[CV 2/5] END …C=1000, gamma=1, kernel=linear;, score=0.950 total time=   0.9s
[CV 3/5] END …C=1000, gamma=1, kernel=linear;, score=1.000 total time=   1.2s
[CV 4/5] END …C=1000, gamma=1, kernel=linear;, score=0.949 total time=   4.5s
[CV 5/5] END …C=1000, gamma=1, kernel=linear;, score=0.987 total time=   3.8s
[CV 1/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.625 total time=   0.0s
[CV 4/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=1000, gamma=0.1, kernel=rbf;, score=0.633 total time=   0.0s
```

```
[CV 1/5] END ..C=1000, gamma=0.1, kernel=linear;, score=0.950 total time=   3.9s
[CV 2/5] END ..C=1000, gamma=0.1, kernel=linear;, score=0.950 total time=   0.9s
[CV 3/5] END ..C=1000, gamma=0.1, kernel=linear;, score=1.000 total time=   1.2s
[CV 4/5] END ..C=1000, gamma=0.1, kernel=linear;, score=0.949 total time=   4.5s
[CV 5/5] END ..C=1000, gamma=0.1, kernel=linear;, score=0.987 total time=   3.8s
[CV 1/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 2/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.637 total time=   0.0s
[CV 3/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.613 total time=   0.0s
[CV 4/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 5/5] END …C=1000, gamma=0.01, kernel=rbf;, score=0.633 total time=   0.0s
[CV 1/5] END .C=1000, gamma=0.01, kernel=linear;, score=0.950 total time=   3.9s
[CV 2/5] END .C=1000, gamma=0.01, kernel=linear;, score=0.950 total time=   0.9s
[CV 3/5] END .C=1000, gamma=0.01, kernel=linear;, score=1.000 total time=   1.2s
[CV 4/5] END .C=1000, gamma=0.01, kernel=linear;, score=0.949 total time=   4.5s
[CV 5/5] END .C=1000, gamma=0.01, kernel=linear;, score=0.987 total time=   3.8s
[CV 1/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.887 total time=   0.0s
[CV 2/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.900 total time=   0.0s
[CV 4/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.937 total time=   0.0s
[CV 5/5] END …C=1000, gamma=0.001, kernel=rbf;, score=0.924 total time=   0.0s
[CV 1/5] END C=1000, gamma=0.001, kernel=linear;, score=0.950 total time=   3.8s
[CV 2/5] END C=1000, gamma=0.001, kernel=linear;, score=0.950 total time=   0.9s
[CV 3/5] END C=1000, gamma=0.001, kernel=linear;, score=1.000 total time=   1.2s
[CV 4/5] END C=1000, gamma=0.001, kernel=linear;, score=0.949 total time=   4.5s
[CV 5/5] END C=1000, gamma=0.001, kernel=linear;, score=0.987 total time=   3.8s
[CV 1/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.938 total time=   0.0s
[CV 2/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.912 total time=   0.0s
[CV 3/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.963 total time=   0.0s
[CV 4/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.924 total time=   0.0s
[CV 5/5] END ..C=1000, gamma=0.0001, kernel=rbf;, score=0.962 total time=   0.0s
[CV 1/5] END C=1000, gamma=0.0001, kernel=linear;, score=0.950 total time=
3.8s
[CV 2/5] END C=1000, gamma=0.0001, kernel=linear;, score=0.950 total time=
0.9s
[CV 3/5] END C=1000, gamma=0.0001, kernel=linear;, score=1.000 total time=
1.2s
[CV 4/5] END C=1000, gamma=0.0001, kernel=linear;, score=0.949 total time=
4.6s
[CV 5/5] END C=1000, gamma=0.0001, kernel=linear;, score=0.987 total time=
3.9s
```

[122]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 5, 10, 50, 100, 1000],
                         'gamma': [10, 1, 0.1, 0.01, 0.001, 0.0001],
                         'kernel': ['rbf', 'linear']},
             verbose=5)
```

You can inspect the best parameters found by GridSearchCV in the **best_params_** attribute,

and the best estimator in the **best_estimator_** attribute:

[123]:
```
grid.best_params_
```

[123]:
```
{'C': 50, 'gamma': 10, 'kernel': 'linear'}
```

[124]:
```
grid.best_estimator_
```

[124]:
```
SVC(C=50, gamma=10, kernel='linear')
```

Then you can re-run predictions on this grid object just like you would with a normal model.

[125]:
```
grid_predictions = grid.predict(X_test)
```

[126]:
```
print(confusion_matrix(y_test,grid_predictions))
```

```
[[ 60   6]
 [  3 102]]
```

[127]:
```
print(classification_report(y_test,grid_predictions))
```

```
              precision    recall  f1-score   support

           0       0.95      0.91      0.93        66
           1       0.94      0.97      0.96       105

    accuracy                           0.95       171
   macro avg       0.95      0.94      0.94       171
weighted avg       0.95      0.95      0.95       171
```

5 Great job!