Support Vector Machines with Python

Welcome to the Support Vector Machines with Python Lecture Notebook! Remember to refer to the video lecture for the full background information on the code here!

Import Libraries

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

Get the Data

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

In [2]:
```
from sklearn.datasets import load_breast_cancer
```

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

The data set is presented in a dictionary form:

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

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

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

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

```
.. _breast_cancer_dataset:

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

**Data Set Characteristics:**

    :Number of Instances: 569

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

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        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.
```

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

Out[6]:
```
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')
```

Set up DataFrame

In [7]:
```
df_feat = pd.DataFrame(cancer['data'],columns=cancer['feature_names'])
pd.set_option('display.max_columns', None) # to display all the columns in df
df_feat
```

Out[7]:

	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	sym
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	1.0950	0.9053	8.589	153.40	0.006399	0.04904	0.05373	0.01587	0.
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.01860	0.01340	0.
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	0.7456	0.7869	4.585	94.03	0.006150	0.04006	0.03832	0.02058	0.
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	0.4956	1.1560	3.445	27.23	0.009110	0.07458	0.05661	0.01867	0.
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	0.7572	0.7813	5.438	94.44	0.011490	0.02461	0.05688	0.01885	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	1.1760	1.2560	7.673	158.70	0.010300	0.02891	0.05198	0.02454	0.
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	0.7655	2.4630	5.203	99.04	0.005769	0.02423	0.03950	0.01678	0.
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	0.4564	1.0750	3.425	48.55	0.005903	0.03731	0.04730	0.01557	0.
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	0.7260	1.5950	5.772	86.22	0.006522	0.06158	0.07117	0.01664	0.
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	0.3857	1.4280	2.548	19.15	0.007189	0.00466	0.00000	0.00000	0.

569 rows × 30 columns

In [8]:
```
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
```

In [9]:
```
df_target = pd.DataFrame(cancer['target'],columns=['Cancer'])
df_target
```

Out[9]:

	Cancer
0	0
1	0
2	0
3	0
4	0
...	...
564	0
565	0
566	0
567	0
568	1

569 rows × 1 columns

Train Test Split

In [10]:
```
from sklearn.model_selection import train_test_split
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(df_feat, np.ravel(df_target), test_size=0.30, random_state=101)
```

Train the Support Vector Classifier

In [12]:
```
from sklearn.svm import SVC
```

In [13]:
```
model = SVC()
```

In [14]:
```
model.fit(X_train,y_train)
```

Out[14]:
```
SVC()
```

Predictions and Evaluations

Now let's predict using the trained model.

In [15]:
```
predictions = model.predict(X_test)
```

In [16]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [17]:
```
print(confusion_matrix(y_test,predictions))
```

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

In [18]:
```
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
```

Woah! Notice that we are classifying everything into a single class! This means our model needs to have it parameters adjusted (it may also help to normalize the data).

We can search for parameters using a GridSearch!

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.

In [19]:
```
param_grid = {'C': [0.1,1, 10, 100, 1000], 'gamma': [1,0.1,0.01,0.001,0.0001], 'kernel': ['rbf']}
```

In [20]:
```
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).

In [21]:
```
grid = GridSearchCV(SVC(),param_grid,refit=True,verbose=3)
```

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.

In [22]:
```
# May take awhile!
grid.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END ....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV 2/5] END ....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV 3/5] END ....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV 4/5] END ....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV 5/5] END ....................C=0.1, gamma=1, kernel=rbf; total time=   0.0s
[CV 1/5] END ..................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 2/5] END ..................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 3/5] END ..................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 4/5] END ..................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 5/5] END ..................C=0.1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 1/5] END .................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 2/5] END .................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 3/5] END .................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 4/5] END .................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 5/5] END .................C=0.1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 1/5] END ................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 2/5] END ................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 3/5] END ................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 4/5] END ................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 5/5] END ................C=0.1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 1/5] END ...............C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 2/5] END ...............C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 3/5] END ...............C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 4/5] END ...............C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 5/5] END ...............C=0.1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 1/5] END ......................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV 2/5] END ......................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV 3/5] END ......................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV 4/5] END ......................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV 5/5] END ......................C=1, gamma=1, kernel=rbf; total time=   0.0s
[CV 1/5] END ....................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 2/5] END ....................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 3/5] END ....................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 4/5] END ....................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 5/5] END ....................C=1, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 1/5] END ...................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 2/5] END ...................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 3/5] END ...................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 4/5] END ...................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 5/5] END ...................C=1, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 1/5] END ..................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 2/5] END ..................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 3/5] END ..................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 4/5] END ..................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 5/5] END ..................C=1, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 1/5] END .................C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 2/5] END .................C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 3/5] END .................C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 4/5] END .................C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 5/5] END .................C=1, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 1/5] END .....................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV 2/5] END .....................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV 3/5] END .....................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV 4/5] END .....................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV 5/5] END .....................C=10, gamma=1, kernel=rbf; total time=   0.0s
[CV 1/5] END ...................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 2/5] END ...................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 3/5] END ...................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 4/5] END ...................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 5/5] END ...................C=10, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 1/5] END ..................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 2/5] END ..................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 3/5] END ..................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 4/5] END ..................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 5/5] END ..................C=10, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 1/5] END .................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 2/5] END .................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 3/5] END .................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 4/5] END .................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 5/5] END .................C=10, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 1/5] END ................C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 2/5] END ................C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 3/5] END ................C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 4/5] END ................C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 5/5] END ................C=10, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 1/5] END ....................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV 2/5] END ....................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV 3/5] END ....................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV 4/5] END ....................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV 5/5] END ....................C=100, gamma=1, kernel=rbf; total time=   0.0s
[CV 1/5] END ..................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 2/5] END ..................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 3/5] END ..................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 4/5] END ..................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 5/5] END ..................C=100, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 1/5] END .................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 2/5] END .................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 3/5] END .................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 4/5] END .................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 5/5] END .................C=100, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 1/5] END ................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 2/5] END ................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 3/5] END ................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 4/5] END ................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 5/5] END ................C=100, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 1/5] END ...............C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 2/5] END ...............C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 3/5] END ...............C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 4/5] END ...............C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 5/5] END ...............C=100, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 1/5] END ...................C=1000, gamma=1, kernel=rbf; total time=   0.0s
[CV 2/5] END ...................C=1000, gamma=1, kernel=rbf; total time=   0.0s
[CV 3/5] END ...................C=1000, gamma=1, kernel=rbf; total time=   0.0s
[CV 4/5] END ...................C=1000, gamma=1, kernel=rbf; total time=   0.0s
[CV 5/5] END ...................C=1000, gamma=1, kernel=rbf; total time=   0.0s
[CV 1/5] END .................C=1000, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 2/5] END .................C=1000, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 3/5] END .................C=1000, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 4/5] END .................C=1000, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 5/5] END .................C=1000, gamma=0.1, kernel=rbf; total time=   0.0s
[CV 1/5] END ................C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 2/5] END ................C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 3/5] END ................C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 4/5] END ................C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 5/5] END ................C=1000, gamma=0.01, kernel=rbf; total time=   0.0s
[CV 1/5] END ...............C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 2/5] END ...............C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 3/5] END ...............C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 4/5] END ...............C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 5/5] END ...............C=1000, gamma=0.001, kernel=rbf; total time=   0.0s
[CV 1/5] END ..............C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 2/5] END ..............C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 3/5] END ..............C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 4/5] END ..............C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
[CV 5/5] END ..............C=1000, gamma=0.0001, kernel=rbf; total time=   0.0s
```

Out[22]:
```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10, 100, 1000],
                         'gamma': [1, 0.1, 0.01, 0.001, 0.0001],
                         'kernel': ['rbf']},
             verbose=3)
```

You can inspect the best parameters found by GridSearchCV in the best*params* attribute, and the best estimator in the best_estimator_ attribute:

In [23]:
```
grid.best_params_
```

Out[23]:
```
{'C': 1, 'gamma': 0.0001, 'kernel': 'rbf'}
```

In [24]:
```
grid.best_estimator_
```

Out[24]:
```
SVC(C=1, gamma=0.0001)
```

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

In [25]:
```
grid_predictions = grid.predict(X_test)
```

In [26]:
```
print(confusion_matrix(y_test,grid_predictions))
```

```
[[ 59   7]
 [  4 101]]
```

In [27]:
```
print(classification_report(y_test,grid_predictions))
```

```
              precision    recall  f1-score   support

           0       0.94      0.89      0.91        66
           1       0.94      0.96      0.95       105

    accuracy                           0.94       171
   macro avg       0.94      0.93      0.93       171
weighted avg       0.94      0.94      0.94       171
```

Great job!