Week 7 Lab (Support Vector Machine)

COSC 3337 Dr. Rizk

About The Data

We'll be using the Breast Cancer Wisconsin (Diagnostic) Data Set from kaggle for this lab, but feel free to follow along with your own dataset. The dataset contains a total of 32 columns, with following attribute information:

1) ID number

2) Diagnosis (M = malignant, B = benign)

3‑32)

Ten real‑valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)

b) texture (standard deviation of gray‑scale values)

c) perimeter

d) area

e) smoothness (local variation in radius lengths)

f) compactness (perimeter^2 / area ‑ 1.0)

g) concavity (severity of concave portions of the contour)

h) concave points (number of concave portions of the contour)

i) symmetry

j) fractal dimension ("coastline approximation" ‑ 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

Our goal will be to predict the diagnosis (benign or malignant).

Exploratory Data Analysis

Let's begin by importing some necessary libraries that we'll be using to explore the data.

In [1]:**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

In [2]:**from** matplotlib **import** rcParams

rcParams['figure.figsize'] **=** 15, 5

sns**.**set\_style('darkgrid')

Our first step is to load the data into a pandas DataFrame.

In [3]:breast\_cancer\_df **=** pd**.**read\_csv('data.csv')

breast\_cancer\_df**.**head()

Out[3]:

id diagnosis radius\_mean texture\_mean perimeter\_mean area\_mean smoothness\_mean compactness\_mean concavi

0 842302 M 17.99 10.38 122.80 1001.0 0.11840 0.27760 1 842517 M 20.57 17.77 132.90 1326.0 0.08474 0.07864 2 84300903 M 19.69 21.25 130.00 1203.0 0.10960 0.15990 3 84348301 M 11.42 20.38 77.58 386.1 0.14250 0.28390 4 84358402 M 20.29 14.34 135.10 1297.0 0.10030 0.13280

5 rows × 33 columns

There's an odd column "Unnamed: 32", which we'll go ahead and drop since it's full of NaN values. We also won't need the id label, so we can drop that as well.

In [4]:breast\_cancer\_df**.**drop(labels**=**['Unnamed: 32', 'id'], axis**=**1, inplace**=True**)

Since a lot of the features in this dataset can be hard to interpret without domain knowledge of cancer or tumor cells, we'll just do a few visualizations here, but feel free to explore as much as you'd like before constructing a model.

In [5]:breast\_cancer\_df**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 569 entries, 0 to 568

Data columns (total 31 columns):

 #   Column                   Non­Null Count  Dtype

­­­  ­­­­­­                   ­­­­­­­­­­­­­­  ­­­­­

 0   diagnosis                569 non­null    object

 1   radius\_mean              569 non­null    float64

 2   texture\_mean             569 non­null    float64

 3   perimeter\_mean           569 non­null    float64

 4   area\_mean                569 non­null    float64

 5   smoothness\_mean          569 non­null    float64

 6   compactness\_mean         569 non­null    float64

 7   concavity\_mean           569 non­null    float64

 8   concave points\_mean      569 non­null    float64

 9   symmetry\_mean            569 non­null    float64

 10  fractal\_dimension\_mean   569 non­null    float64

 11  radius\_se                569 non­null    float64

 12  texture\_se               569 non­null    float64

 13  perimeter\_se             569 non­null    float64

 14  area\_se                  569 non­null    float64

 15  smoothness\_se            569 non­null    float64

 16  compactness\_se           569 non­null    float64

 17  concavity\_se             569 non­null    float64

 18  concave points\_se        569 non­null    float64

 19  symmetry\_se              569 non­null    float64

 20  fractal\_dimension\_se     569 non­null    float64

 21  radius\_worst             569 non­null    float64

 22  texture\_worst            569 non­null    float64

 23  perimeter\_worst          569 non­null    float64

 24  area\_worst               569 non­null    float64

 25  smoothness\_worst         569 non­null    float64

 26  compactness\_worst        569 non­null    float64

 27  concavity\_worst          569 non­null    float64

 28  concave points\_worst     569 non­null    float64

 29  symmetry\_worst           569 non­null    float64

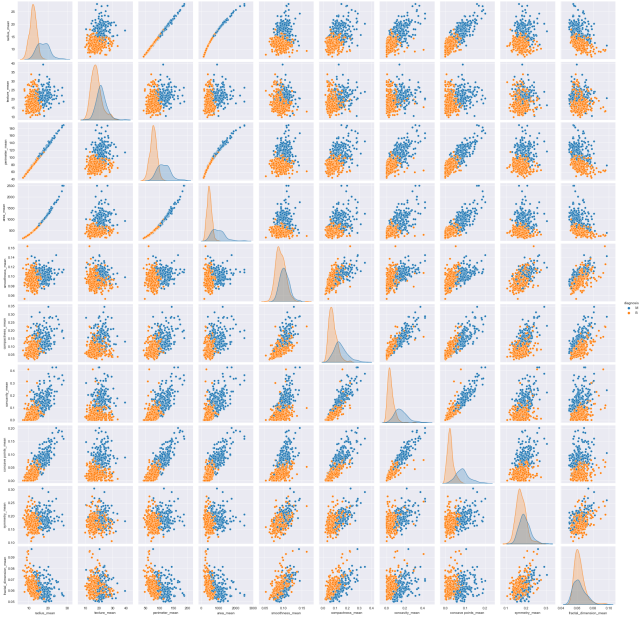
 30  fractal\_dimension\_worst  569 non­null    float64

dtypes: float64(30), object(1)

memory usage: 137.9+ KB

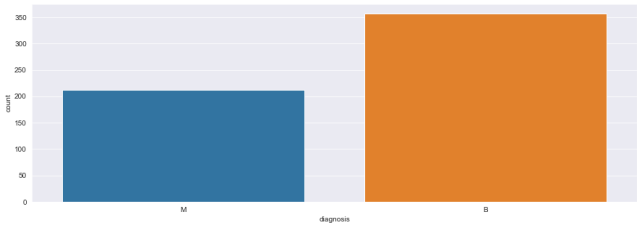
calling .info() we see that there are no missing values in this dataset.

In [6]: sns**.**pairplot(breast\_cancer\_df, hue**=**'diagnosis', vars**=**['radius\_mean', 'texture\_mean', 'perimeter\_mean', 'area\_                                      'smoothness\_mean', 'compactness\_mean', 'concavity\_mean',                                      'concave points\_mean', 'symmetry\_mean', 'fractal\_dimension\_mean']) plt**.**show()

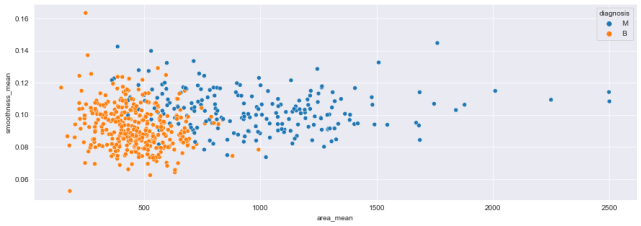
There seems to be pretty good distinction between the diagnosis (blue & orange) in most of the atributes above.

In [7]: sns**.**countplot(x**=**breast\_cancer\_df['diagnosis'])

plt**.**show()

Majority of our data observations are of the benign class.

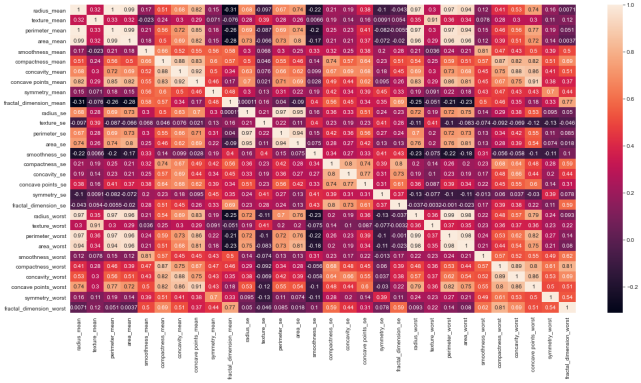
In [8]: sns**.**scatterplot(x **=** 'area\_mean', y **=** 'smoothness\_mean', hue **=** 'diagnosis', data **=** breast\_cancer\_df) plt**.**show()

area\_mean could be a good predictor wheather malignant or benign since there is pretty good separation here. Most benign (orange) have area\_mean of around 500 or lower.

In [9]:plt**.**figure(figsize**=**(20,10))

sns**.**heatmap(breast\_cancer\_df**.**corr(), annot**=True**)

plt**.**show()

Some strong correlations are present. (very bright squares for example)

Pre‑Processing

Let's go ahead and scale our data before training and creating our model

In [10]:**from** sklearn.preprocessing **import** StandardScaler

*# all columns except 'Outcome'*

X **=** breast\_cancer\_df**.**drop('diagnosis', axis**=**1)

y **=** breast\_cancer\_df['diagnosis']

*# create our scaler object*

scaler **=** StandardScaler()

*# use our scaler object to transform/scale our data and save it into X\_scaled*

X\_scaled **=** scaler**.**fit\_transform(X)

*# reassign X to a new DataFrame using the X\_scaled values.*

X **=** pd**.**DataFrame(data**=**X\_scaled, columns**=**X**.**columns)

In [11]:X**.**head()

Out[11]:

radius\_mean texture\_mean perimeter\_mean area\_mean smoothness\_mean compactness\_mean concavity\_meanconcave points\_mean

0 1.097064 ‑2.073335 1.269934 0.984375 1.568466 3.283515 2.652874 2.532475 1 1.829821 ‑0.353632 1.685955 1.908708 ‑0.826962 ‑0.487072 ‑0.023846 0.548144 2 1.579888 0.456187 1.566503 1.558884 0.942210 1.052926 1.363478 2.037231 3 ‑0.768909 0.253732 ‑0.592687 ‑0.764464 3.283553 3.402909 1.915897 1.451707 4 1.750297 ‑1.151816 1.776573 1.826229 0.280372 0.539340 1.371011 1.428493

5 rows × 30 columns

Creating our Model

We're now ready to begin creating and training our model. We first need to split our data into training and testing sets. This can be done using sklearn's train\_test\_split(X, y, test\_size) function. This function takes in your features (X), the target variable (y), and the test\_size you'd like (Generally a test size of around 0.3 is good enough). It will then return a tuple of X\_train, X\_test, y\_train, y\_test sets for us. We will train our model on the training set and then use the test set to evaluate the model.

In [12]:**from** sklearn.model\_selection **import** train\_test\_split

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, test\_size**=**0.3, random\_state**=**101)

In [13]:**from** sklearn.svm **import** SVC

*# instantiate the model with default parameters*

model **=** SVC()

*# fit/train*

model**.**fit(X\_train,y\_train)

Out[13]:

SVC()

Model Evaluation

Now that we've finished training, we can make predictions off of the test data and evaluate our model's performance using the corresponding test data.

In [14]:predictions **=** model**.**predict(X\_test)

In [15]:**from** sklearn.metrics **import** confusion\_matrix

cm **=** confusion\_matrix(y\_test, predictions)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])

Confusion matrix

 [[104   1]

 [  3  63]]

True Positives(TP) =  104

True Negatives(TN) =  63

False Positives(FP) =  1

False Negatives(FN) =  3

In [16]:**from** sklearn.metrics **import** classification\_report

print(classification\_report(y\_test,predictions))

              precision    recall  f1­score   support

           B       0.97      0.99      0.98       105

           M       0.98      0.95      0.97        66

    accuracy                           0.98       171

   macro avg       0.98      0.97      0.98       171

weighted avg       0.98      0.98      0.98       171

Hyperparameter Tuning

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.

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. Let's go ahead and try a few different parameters to see which of them is the best set to use.

In [17]:param\_grid **=** {'C': [0.1,1, 10, 100, 1000], 'gamma': [1,0.1,0.01,0.001,0.0001], 'kernel': ['rbf']}

You should add refit=True and choose verbose to whatever number you want. The higher the number, the more verbose.(verbose just means the text output describing the process).

In [19]:**from** sklearn.model\_selection **import** GridSearchCV

grid **=** GridSearchCV(SVC(),param\_grid,refit**=True**,verbose**=**3)

What fit does is a bit more involved than 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 build a single new model using the best parameter setting.

Note: This process may take a while. The more parameters we test, the longer it may take since it has to try all different combinations inorder to find the best set.

In [20]:grid**.**fit(X\_train,y\_train)

Fitting 5 folds for each of 25 candidates, totalling 125 fits

[CV] C=0.1, gamma=1, kernel=rbf ......................................

[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=0.1, gamma=1, kernel=rbf ......................................

[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=0.1, gamma=1, kernel=rbf ......................................

[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.625, total=   0.0s

[CV] C=0.1, gamma=1, kernel=rbf ......................................

[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=0.1, gamma=1, kernel=rbf ......................................

[CV] .......... C=0.1, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=0.1, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.925, total=   0.0s

[CV] C=0.1, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.950, total=   0.0s

[CV] C=0.1, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.900, total=   0.0s

[CV] C=0.1, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.962, total=   0.0s

[CV] C=0.1, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=0.1, gamma=0.1, kernel=rbf, score=0.949, total=   0.0s

[CV] C=0.1, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.912, total=   0.0s

[CV] C=0.1, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.963, total=   0.0s

[CV] C=0.1, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.975, total=   0.0s

[CV] C=0.1, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.987, total=   0.0s

[CV] C=0.1, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=0.1, gamma=0.01, kernel=rbf, score=0.962, total=   0.0s

[CV] C=0.1, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.688, total=   0.0s

[CV] C=0.1, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.688, total=   0.0s

[CV] C=0.1, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.688, total=   0.0s

[CV] C=0.1, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.684, total=   0.0s

[CV] C=0.1, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=0.1, gamma=0.001, kernel=rbf, score=0.709, total=   0.0s

[CV] C=0.1, gamma=0.0001, kernel=rbf .................................

[CV] ..... C=0.1, gamma=0.0001, kernel=rbf, score=0.637, total=   0.0s

[CV] C=0.1, gamma=0.0001, kernel=rbf .................................

[CV] ..... C=0.1, gamma=0.0001, kernel=rbf, score=0.637, total=   0.0s

[CV] C=0.1, gamma=0.0001, kernel=rbf .................................

[CV] ..... C=0.1, gamma=0.0001, kernel=rbf, score=0.625, total=   0.0s

[CV] C=0.1, gamma=0.0001, kernel=rbf .................................

[CV] ..... C=0.1, gamma=0.0001, kernel=rbf, score=0.633, total=   0.0s

[CV] C=0.1, gamma=0.0001, kernel=rbf .................................

[Parallel(n\_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[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

[CV] ..... C=0.1, gamma=0.0001, kernel=rbf, score=0.633, total=   0.0s

[CV] C=1, gamma=1, kernel=rbf ........................................

[CV] ............ C=1, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=1, gamma=1, kernel=rbf ........................................

[CV] ............ C=1, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=1, gamma=1, kernel=rbf ........................................

[CV] ............ C=1, gamma=1, kernel=rbf, score=0.625, total=   0.0s

[CV] C=1, gamma=1, kernel=rbf ........................................

[CV] ............ C=1, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=1, gamma=1, kernel=rbf ........................................

[CV] ............ C=1, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=1, gamma=0.1, kernel=rbf ......................................

[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.950, total=   0.0s

[CV] C=1, gamma=0.1, kernel=rbf ......................................

[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.950, total=   0.0s

[CV] C=1, gamma=0.1, kernel=rbf ......................................

[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.963, total=   0.0s

[CV] C=1, gamma=0.1, kernel=rbf ......................................

[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1, gamma=0.1, kernel=rbf ......................................

[CV] .......... C=1, gamma=0.1, kernel=rbf, score=0.987, total=   0.0s

[CV] C=1, gamma=0.01, kernel=rbf .....................................

[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.950, total=   0.0s

[CV] C=1, gamma=0.01, kernel=rbf .....................................

[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.988, total=   0.0s

[CV] C=1, gamma=0.01, kernel=rbf .....................................

[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.988, total=   0.0s

[CV] C=1, gamma=0.01, kernel=rbf .....................................

[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.987, total=   0.0s

[CV] C=1, gamma=0.01, kernel=rbf .....................................

[CV] ......... C=1, gamma=0.01, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1, gamma=0.001, kernel=rbf ....................................

[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.912, total=   0.0s

[CV] C=1, gamma=0.001, kernel=rbf ....................................

[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.963, total=   0.0s

[CV] C=1, gamma=0.001, kernel=rbf ....................................

[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1, gamma=0.001, kernel=rbf ....................................

[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.987, total=   0.0s

[CV] C=1, gamma=0.001, kernel=rbf ....................................

[CV] ........ C=1, gamma=0.001, kernel=rbf, score=0.962, total=   0.0s

[CV] C=1, gamma=0.0001, kernel=rbf ...................................

[CV] ....... C=1, gamma=0.0001, kernel=rbf, score=0.688, total=   0.0s

[CV] C=1, gamma=0.0001, kernel=rbf ...................................

[CV] ....... C=1, gamma=0.0001, kernel=rbf, score=0.700, total=   0.0s

[CV] C=1, gamma=0.0001, kernel=rbf ...................................

[CV] ....... C=1, gamma=0.0001, kernel=rbf, score=0.700, total=   0.0s

[CV] C=1, gamma=0.0001, kernel=rbf ...................................

[CV] ....... C=1, gamma=0.0001, kernel=rbf, score=0.696, total=   0.0s

[CV] C=1, gamma=0.0001, kernel=rbf ...................................

[CV] ....... C=1, gamma=0.0001, kernel=rbf, score=0.709, total=   0.0s

[CV] C=10, gamma=1, kernel=rbf .......................................

[CV] ........... C=10, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=10, gamma=1, kernel=rbf .......................................

[CV] ........... C=10, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=10, gamma=1, kernel=rbf .......................................

[CV] ........... C=10, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=10, gamma=1, kernel=rbf .......................................

[CV] ........... C=10, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=10, gamma=1, kernel=rbf .......................................

[CV] ........... C=10, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=10, gamma=0.1, kernel=rbf .....................................

[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.950, total=   0.0s

[CV] C=10, gamma=0.1, kernel=rbf .....................................

[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.963, total=   0.0s

[CV] C=10, gamma=0.1, kernel=rbf .....................................

[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.963, total=   0.0s

[CV] C=10, gamma=0.1, kernel=rbf .....................................

[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.987, total=   0.0s

[CV] C=10, gamma=0.1, kernel=rbf .....................................

[CV] ......... C=10, gamma=0.1, kernel=rbf, score=0.987, total=   0.0s

[CV] C=10, gamma=0.01, kernel=rbf ....................................

[CV] ........ C=10, gamma=0.01, kernel=rbf, score=0.963, total=   0.0s

[CV] C=10, gamma=0.01, kernel=rbf ....................................

[CV] ........ C=10, gamma=0.01, kernel=rbf, score=0.975, total=   0.0s

[CV] C=10, gamma=0.01, kernel=rbf ....................................

[CV] ........ C=10, gamma=0.01, kernel=rbf, score=1.000, total=   0.0s

[CV] C=10, gamma=0.01, kernel=rbf ....................................

[CV] ........ C=10, gamma=0.01, kernel=rbf, score=0.975, total=   0.0s

[CV] C=10, gamma=0.01, kernel=rbf ....................................

[CV] ........ C=10, gamma=0.01, kernel=rbf, score=0.987, total=   0.0s

[CV] C=10, gamma=0.001, kernel=rbf ...................................

[CV] ....... C=10, gamma=0.001, kernel=rbf, score=0.950, total=   0.0s

[CV] C=10, gamma=0.001, kernel=rbf ...................................

[CV] ....... C=10, gamma=0.001, kernel=rbf, score=0.988, total=   0.0s

[CV] C=10, gamma=0.001, kernel=rbf ...................................

[CV] ....... C=10, gamma=0.001, kernel=rbf, score=1.000, total=   0.0s

[CV] C=10, gamma=0.001, kernel=rbf ...................................

[CV] ....... C=10, gamma=0.001, kernel=rbf, score=0.987, total=   0.0s

[CV] C=10, gamma=0.001, kernel=rbf ...................................

[CV] ....... C=10, gamma=0.001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=10, gamma=0.0001, kernel=rbf ..................................

[CV] ...... C=10, gamma=0.0001, kernel=rbf, score=0.912, total=   0.0s

[CV] C=10, gamma=0.0001, kernel=rbf ..................................

[CV] ...... C=10, gamma=0.0001, kernel=rbf, score=0.963, total=   0.0s

[CV] C=10, gamma=0.0001, kernel=rbf ..................................

[CV] ...... C=10, gamma=0.0001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=10, gamma=0.0001, kernel=rbf ..................................

[CV] ...... C=10, gamma=0.0001, kernel=rbf, score=0.987, total=   0.0s

[CV] C=10, gamma=0.0001, kernel=rbf ..................................

[CV] ...... C=10, gamma=0.0001, kernel=rbf, score=0.949, total=   0.0s

[CV] C=100, gamma=1, kernel=rbf ......................................

[CV] .......... C=100, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=100, gamma=1, kernel=rbf ......................................

[CV] .......... C=100, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=100, gamma=1, kernel=rbf ......................................

[CV] .......... C=100, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=100, gamma=1, kernel=rbf ......................................

[CV] .......... C=100, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=100, gamma=1, kernel=rbf ......................................

[CV] .......... C=100, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=100, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=100, gamma=0.1, kernel=rbf, score=0.950, total=   0.0s

[CV] C=100, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=100, gamma=0.1, kernel=rbf, score=0.963, total=   0.0s

[CV] C=100, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=100, gamma=0.1, kernel=rbf, score=0.975, total=   0.0s

[CV] C=100, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=100, gamma=0.1, kernel=rbf, score=0.975, total=   0.0s

[CV] C=100, gamma=0.1, kernel=rbf ....................................

[CV] ........ C=100, gamma=0.1, kernel=rbf, score=0.987, total=   0.0s

[CV] C=100, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=100, gamma=0.01, kernel=rbf, score=0.975, total=   0.0s

[CV] C=100, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=100, gamma=0.01, kernel=rbf, score=0.938, total=   0.0s

[CV] C=100, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=100, gamma=0.01, kernel=rbf, score=0.975, total=   0.0s

[CV] C=100, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=100, gamma=0.01, kernel=rbf, score=0.949, total=   0.0s

[CV] C=100, gamma=0.01, kernel=rbf ...................................

[CV] ....... C=100, gamma=0.01, kernel=rbf, score=0.975, total=   0.0s

[CV] C=100, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=100, gamma=0.001, kernel=rbf, score=0.950, total=   0.0s

[CV] C=100, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=100, gamma=0.001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=100, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=100, gamma=0.001, kernel=rbf, score=1.000, total=   0.0s

[CV] C=100, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=100, gamma=0.001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=100, gamma=0.001, kernel=rbf ..................................

[CV] ...... C=100, gamma=0.001, kernel=rbf, score=0.987, total=   0.0s

[CV] C=100, gamma=0.0001, kernel=rbf .................................

[CV] ..... C=100, gamma=0.0001, kernel=rbf, score=0.950, total=   0.0s

[CV] C=100, gamma=0.0001, kernel=rbf .................................

[CV] ..... C=100, gamma=0.0001, kernel=rbf, score=0.988, total=   0.0s

[CV] C=100, gamma=0.0001, kernel=rbf .................................

[CV] ..... C=100, gamma=0.0001, kernel=rbf, score=1.000, total=   0.0s

[CV] C=100, gamma=0.0001, kernel=rbf .................................

[CV] ..... C=100, gamma=0.0001, kernel=rbf, score=0.987, total=   0.0s

[CV] C=100, gamma=0.0001, kernel=rbf .................................

[CV] ..... C=100, gamma=0.0001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1000, gamma=1, kernel=rbf .....................................

[CV] ......... C=1000, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=1000, gamma=1, kernel=rbf .....................................

[CV] ......... C=1000, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=1000, gamma=1, kernel=rbf .....................................

[CV] ......... C=1000, gamma=1, kernel=rbf, score=0.637, total=   0.0s

[CV] C=1000, gamma=1, kernel=rbf .....................................

[CV] ......... C=1000, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=1000, gamma=1, kernel=rbf .....................................

[CV] ......... C=1000, gamma=1, kernel=rbf, score=0.633, total=   0.0s

[CV] C=1000, gamma=0.1, kernel=rbf ...................................

[CV] ....... C=1000, gamma=0.1, kernel=rbf, score=0.950, total=   0.0s

[CV] C=1000, gamma=0.1, kernel=rbf ...................................

[CV] ....... C=1000, gamma=0.1, kernel=rbf, score=0.963, total=   0.0s

[CV] C=1000, gamma=0.1, kernel=rbf ...................................

[CV] ....... C=1000, gamma=0.1, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1000, gamma=0.1, kernel=rbf ...................................

[CV] ....... C=1000, gamma=0.1, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1000, gamma=0.1, kernel=rbf ...................................

[CV] ....... C=1000, gamma=0.1, kernel=rbf, score=0.987, total=   0.0s

[CV] C=1000, gamma=0.01, kernel=rbf ..................................

[CV] ...... C=1000, gamma=0.01, kernel=rbf, score=0.950, total=   0.0s

[CV] C=1000, gamma=0.01, kernel=rbf ..................................

[CV] ...... C=1000, gamma=0.01, kernel=rbf, score=0.950, total=   0.0s

[CV] C=1000, gamma=0.01, kernel=rbf ..................................

[CV] ...... C=1000, gamma=0.01, kernel=rbf, score=0.950, total=   0.0s

[CV] C=1000, gamma=0.01, kernel=rbf ..................................

[CV] ...... C=1000, gamma=0.01, kernel=rbf, score=0.937, total=   0.0s

[CV] C=1000, gamma=0.01, kernel=rbf ..................................

[CV] ...... C=1000, gamma=0.01, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1000, gamma=0.001, kernel=rbf .................................

[CV] ..... C=1000, gamma=0.001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1000, gamma=0.001, kernel=rbf .................................

[CV] ..... C=1000, gamma=0.001, kernel=rbf, score=0.963, total=   0.0s

[CV] C=1000, gamma=0.001, kernel=rbf .................................

[CV] ..... C=1000, gamma=0.001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1000, gamma=0.001, kernel=rbf .................................

[CV] ..... C=1000, gamma=0.001, kernel=rbf, score=0.949, total=   0.0s

[CV] C=1000, gamma=0.001, kernel=rbf .................................

[CV] ..... C=1000, gamma=0.001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1000, gamma=0.0001, kernel=rbf ................................

[CV] .... C=1000, gamma=0.0001, kernel=rbf, score=0.950, total=   0.0s

[CV] C=1000, gamma=0.0001, kernel=rbf ................................

[CV] .... C=1000, gamma=0.0001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1000, gamma=0.0001, kernel=rbf ................................

[CV] .... C=1000, gamma=0.0001, kernel=rbf, score=1.000, total=   0.0s

[CV] C=1000, gamma=0.0001, kernel=rbf ................................

[CV] .... C=1000, gamma=0.0001, kernel=rbf, score=0.975, total=   0.0s

[CV] C=1000, gamma=0.0001, kernel=rbf ................................

[CV] .... C=1000, gamma=0.0001, kernel=rbf, score=0.987, total=   0.0s

[Parallel(n\_jobs=1)]: Done 125 out of 125 | elapsed:    1.0s finished

Out[20]:

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 using the bestparams attribute, and the best estimator using the best\_estimator\_ attribute. Here we see that the best set of parameters from the ones we specified are 10 for c value, 0.01 for gamma, and 'rbf' for the kernel.

In [21]:grid**.**best\_params\_

Out[21]:

{'C': 10, 'gamma': 0.01, 'kernel': 'rbf'}

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

In [23]:grid\_predictions **=** grid**.**predict(X\_test)

In [24]:print(confusion\_matrix(y\_test,grid\_predictions))

print(classification\_report(y\_test,grid\_predictions))

[[105   0]

 [  2  64]]

              precision    recall  f1­score   support

           B       0.98      1.00      0.99       105

           M       1.00      0.97      0.98        66

    accuracy                           0.99       171

   macro avg       0.99      0.98      0.99       171

weighted avg       0.99      0.99      0.99       171

Nice! We got a slightly better improvement using these parameters, though our original accuracy was already very good. Keep this grid search in mind when you need to do hyperparameter tuning. It can save you a lot of time.

Congrats!  You now know how to use SVM and hyperparameter tuning in sklearn. Try using this on your own dataset and refer back to this lecture if you get stuck.