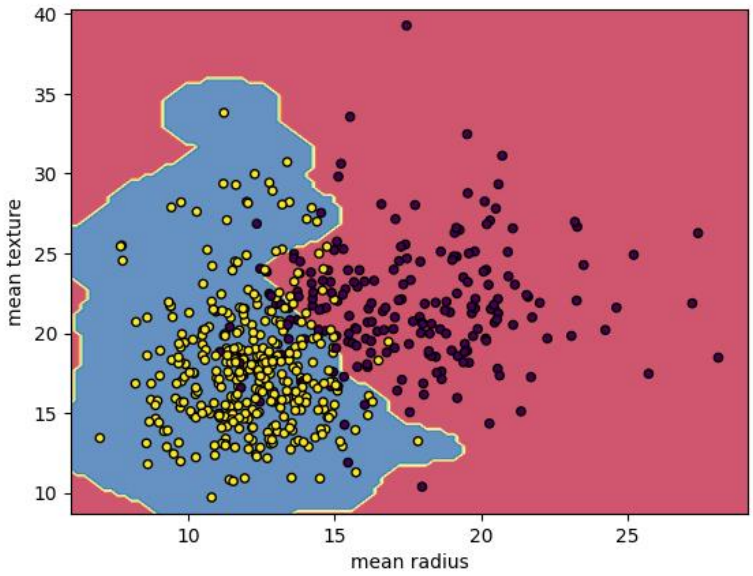
Lab #3 : kNN

Objectives : learn how to build and train a machine learning classifier in python using a Support Vector Machine (SVM) algorithm

Exercise #1

1. Load the breast cancer dataset from sklearn.datasets
2. Print the names of the features and the label
3. Separate input features and target variables.
4. Buil and train the SVM classifiers using RBF kernel.
5. Plot the scatter plot of the input features.
6. Plot the decision boundary.

You should get something like this :



Exercise #2

Objective : build (train and test) a multiclass classifier (3 classes), using Python and Scikitlean library.

You will develop two different classifiers to show the usage of two different kernel functions; Polynomial and RBF. The code should also calculate the accuracy and f1 scores to show the performance difference between the two selected kernel functions on the same dataset.

In this code, You will use the Iris flower dataset. That dataset contains three classes of 50 instances each, where each class refers to a type of Iris plant.

You should follow these steps :

1. Import the needed classes: svm, datasets, …
2. Load Iris dataset
3. separate features set from the target column (class label), and divide the data set to 80% for training, and 20% for testing
4. create two objects from SVM, to create two different classifiers; one with Polynomial kernel, and another one with RBF kernel
5. calculate the efficiency of the two models (test the two classifiers using the test data set)
6. calculate the accuracy and f1 scores for SVM with Polynomial kernel
7. In the same way, the accuracy and f1 scores for SVM with RBF kernel

Exercicse 3

You are given a dataset containing information about handwritten digits. Each image in the dataset is represented by an array of pixel values and is labeled with the corresponding digit it represents (0-9). The digits dataset consists of 8x8 pixel images of digits. The images attribute of the dataset stores 8x8 arrays of grayscale values for each image. Your task is to train a SVM model on this dataset to classify the digits.

Here are the steps to complete the exercise:

1. Import the needed classes: svm, datasets, …

2. Load the dataset: Use datasets.load_digits(). Split the dataset into training and testing sets.

3. visualize the first 4 images.

```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, label in zip(axes, digits.images, digits.target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
    ax.set_title("Training: %i" % label)
```

4. flatten the images.

5. Create a classifier: a support vector classifier with gamma = 0.001

6. Split the dataset and train the SVM model.

7. Predict the value of the digit on the test subset

8. visualize the first 4 test samples and show their predicted digit value in the title.

```
_, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
for ax, image, prediction in zip(axes, X_test, predicted):
    ax.set_axis_off()
    image = image.reshape(8, 8)
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation="nearest")
    ax.set_title(f"Prediction: {prediction}")
```

9. Build a text report showing the main classification metrics (use : metrics.classification_report).

10. Plot a confusion matrix of the true digit values and the predicted digit values. Use : metrics.ConfusionMatrixDisplay.from_predictions