Sequence 4 exercises

1. SVM Classifier: basic study **(homework)**

In this exercise, you will be using support vector machines (SVMs) with various 2D datasets. Experimenting with these datasets will help you gain an intuition of how SVMs work and how to use a Gaussian kernel with SVMs.

1.1 Linear SVM Classification [1]

All the code and data required for this part are on chamilo in the ***SVM Basic study*** directory.

NOTA 1: for the two first exercises, you are going to use a non-optimal implementation of the SVM algorithm. Then you will be asked to use the SVC function provided in the scikit-learn library.

NOTA 2: Most SVM software packages automatically add the extra feature $x0 = 1$ for you and automatically take care of learning the intercept term $\theta 0$. So when passing your training data to the SVM software, there is no need to add this extra feature $x0=1$ yourself

Let's begin with the 2D example dataset exo6*data1.mat*. Run the script **E*xos_SVM_student_code.py*** to plot the data. Comment the data repartition.

In this dataset, the positions of the positive examples (indicated with +) and the negative examples (indicated with o) suggest a natural separation indicated by the gap. However, notice that there is an outlier positive example + on the far left at about (0:1; 4:1). As part of this exercise, you will also see how this outlier affects the SVM decision boundary.

The purpose here is to study the influence of the C regularization parameter on the SVM classifier design and on the classification performances.

Run the code ***Exos_SVM_student_code.py***. How many samples in the data set and how many features for each sample? Run the code with first C=1 and then C=100. Comment the two obtained decision boundaries and conclude about the influence of the C value.

1.2 SVM with Gaussian Kernel [1]

Run **E*xos_SVM_student_code.py*** with the exo6*data2.mat* dataset in order to display the dataset. Obviously the dataset is not linearly separable with the provided features. This idea is then to classify the data by using an SVM classifier with a Gaussian Kernel. You can think about the Gaussian kernel as a similarity function that measures the "distance" between a pair of examples, $(x_{(i)}; x_{(j)})$. The Gaussian kernel is parameterized by a bandwidth parameter, σ, which determines how fast the similarity metric decreases as the examples are further apart.

Exos.SVM_student_code.py estimates the SVM classifier with a Gaussian kernel with parameters C=1 and σ=0.1 (generally good default hyper-parameter values) and displays the classification boundaries which are no more linear. So it is possible with an SVM to classify non linearly separable data.

1.3 Implementing SVC from sklean library

Considering *exo6data3.mat* dataset (it has been loaded and displayed in the *Exos_SVM_student_code.py* file), proceed to an SVM classification by using the SVC function of the sklearn library (look at the documentation). Compare the performances of a linear kernel and a Gaussian kernel ('rbf' kernel in the SVC function) SVM based classifier. Give the used parameter value(s). Compute the classification accuracy (use the *accuracy_function()*) and the precision and recall scores (use the *precision_score()* and *recall_score()* functions). What is the best model to use and what are the best parameter values?

2. PCA basic study **(homework)**

The aim of this exercise is to implement dimensionality reduction by using the Principal Component Analysis (PCA) algorithm and to study its impact on classification performances.

All the code and/or data required for this part are on chamilo in the ***PCA Basic study*** **directory**.

2.1 Basic example: iris data set

This data set is made of 150 samples of iris flowers belonging to three different balanced classes: setosa, versicolor and virginica. Each sample is described with a 4 dimension feature vector containing the length and width of sepal and the length and width of petal (cf figure 1)

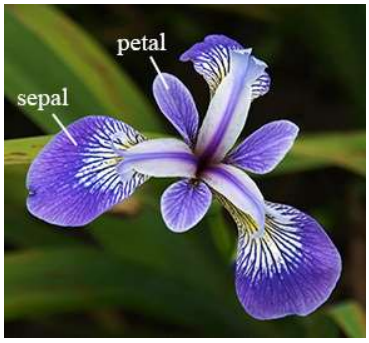


Figure 1: iris sample

The iris data set is a dataset that is available in the sklearn datasets library. Run the file ***Exos_PCA_student_code.py***. It loads the data and displays the samples considering first the sepal dimensions (dim 0 and 1) and then the petal dimensions (dim 2 and 3). Comment the data repartition in each case.

Complete the ***Exos_PCA_student_code.py*** file in order to proceed to a PCA. Before PCA, data have to be normalized (cf. *StandardScaler()* function). There are two ways of using the PCA algorithm: either the user defines the number of components to be kept or the user defines the % of the total variance to be kept.

How many components do you have when keeping 98% of the total variance?
What % of the total variance is kept when keeping the two highest components only?

2.2 PCA or not PCA before iris SVM based classification

Complete the ***Exos_PCA_student_code.py*** file in order to proceed to a linear SVM classification of the iris dataset with the whole features and then with the three and then the two main components only. Compare the obtained accuracies.

2.3 PCA: some counter-examples

Run ***Exos_PCA_student_code.py*** and look at the two proposed counter examples when using PCA is either useless or harmful.

3. Basic K-mean clustering **(homework)**

In this exercise, you will be guided in order to understand on a concrete example how k-means clustering is working. The whole detailed Python code is provided in the ***K-mean basis*** **directory**.

3.1 Running k-means clustering step by step on a 2D dataset [1]

The K-means algorithm is a method to automatically cluster similar data examples together. Concretely, you are given a training set $\{x^{(1)}; \ldots ; x^{(m)}\}$ (where $x^{(i)} \in R^n$), and want to group the data into a few cohesive clusters. The intuition behind K-means is an iterative procedure that starts by guessing the initial centroids, and then refines this guess by repeatedly assigning examples to their closest centroids and then re-computing the centroids based on the assignments. As the consequence, kmeans() is based on two assumptions:

- The "cluster center" is the arithmetic mean of all the points belonging to the cluster
- Each point is closer to its own cluster center than to other cluster centers

Exos_Kmeans_student_code.py is an implementation of this algorithm involving iteratively the two following functions: *findClosestCentroids()* and *computeCentroids().*The inner-loop of the algorithm repeatedly carries out two steps: (i) Assigning each training example x(i) to its closest centroid, and (ii) Re-computing the mean of each centroid using the points assigned to it.

Run ***Exos_Kmeans_student_code.py*** on the proposed 2D dataset in order to understand how it works. This is a direct illustration of what has been presented in the lecture. Plot the data before any clustering in order to check if K=3 seems to be a good value.

In the first run of *K_mean_algo*, the initial centroid points are manually selected to three arbitrary points that are [3 3; 6 2; 8 5]. Display the obtained final clustering.

In practice, since the clustering might depend on the initial centroid choice, it is better to choose them randomly. The function *kMeansInitCentroids()* is provided to randomly select the initial centroids among the provided dataset. Instead of manually initializing the centroids, run the *Kmean_algo* script by using *kMeansInitCentroids()* function for centroid initialization and run it several times. What do you notice?

More precisely, the *kMeansInitCentroids()* function might select the three following points [2.5437 0.9573; 1.4037 4.5753; 1.8298 4.5966] which are actually points of the dataset. Run the *Kmean_algo* script by using these new points as initial centroids and compare the obtained clustering with the previous one obtained with [3 3; 6 2; 8 5] as initial centroids. What happen?

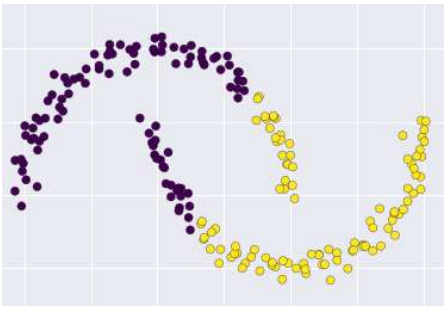
3.2 Using the Kmeans() function of the sklearn library

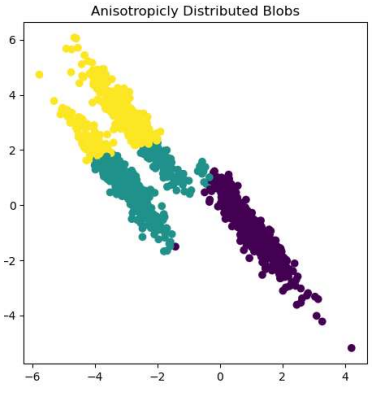
Run a Kmeans clustering on the same data but by using the *Kmeans()* function of the sklearn library (look at the related documentation in order to understand inputs and outputs). This function also returns the intra class variance of each cluster. For both initial centroids [3 3; 6 2; 8 5] and [2.5437 0.9573; 1.4037 4.5753; 1.8298 4.5966], compute the intra class variance of each cluster. Conclusion? Looking at the *KMeans()* function inputs, what is the solution to avoid the effect of a bad initialization?

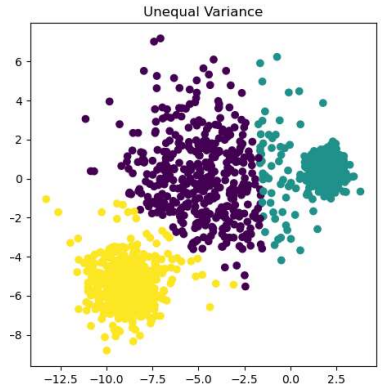
3.3 Kmeans algo: some counter-examples

The fundamental assumption of Kmeans is that points are closer to their own cluster center than to the others means and that the algorithm is not very efficient if the clusters have complicated geometries. In particular, the boundaries between k-means clusters are always linear.

Here are some cases where obviously kmeans clustering is not effective:







4. Predicting a breast cancer **(classwork)**

All the codes and/or data required for this part are on chamilo in the **breast cancer** directory.

4.1 Dataset description

The breast cancer Wisconsin diagnosis data set (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29) contains characteristics of the cell nuclei computed from a digitized image. The attribute information are:

- ID number
- 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.

Run the code ***Breast_cancer_students_code.py*** in order to load the data set. Go to the Spyder window entitled "Explorateur de variables" located above the console window and click on the cancer variable in order to explore all the variable contents. What are the names of the available features? What are the labels names and values? How many samples in the data set? How many positive and negative samples in the data set?

4.2 SVM classification

Complete the ***Breast_cancer_student_code.py*** file in order to split randomly the data set into two parts: 70% of the data for the training set and 30% for the test set.
Learn a linear and then a Gaussian SVM classifier with default parameters on the training set and report the accuracy, precision and recall scores computed on the test set. Comment the obtained results, which is the best model? Test also the impact of data normalization before classification. Fill in the following table with your results. Conclusion?

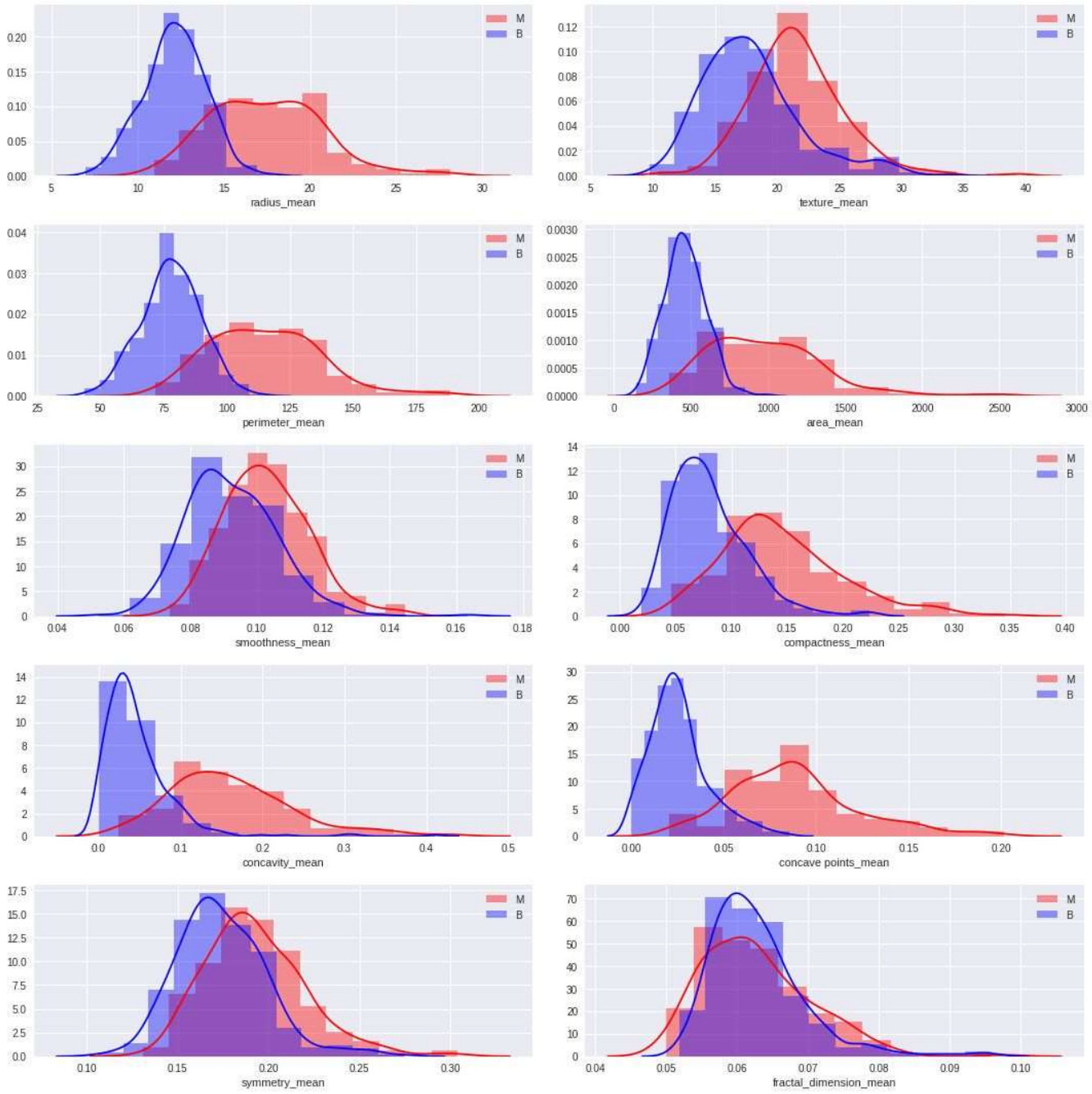
	Accuracy	**Precision**	**Recall**
Linear SVM			
Rbf SVM			
Linear SVM with data normalization			
Rbf SVM with data normalization			

4.3 Dataset analysis

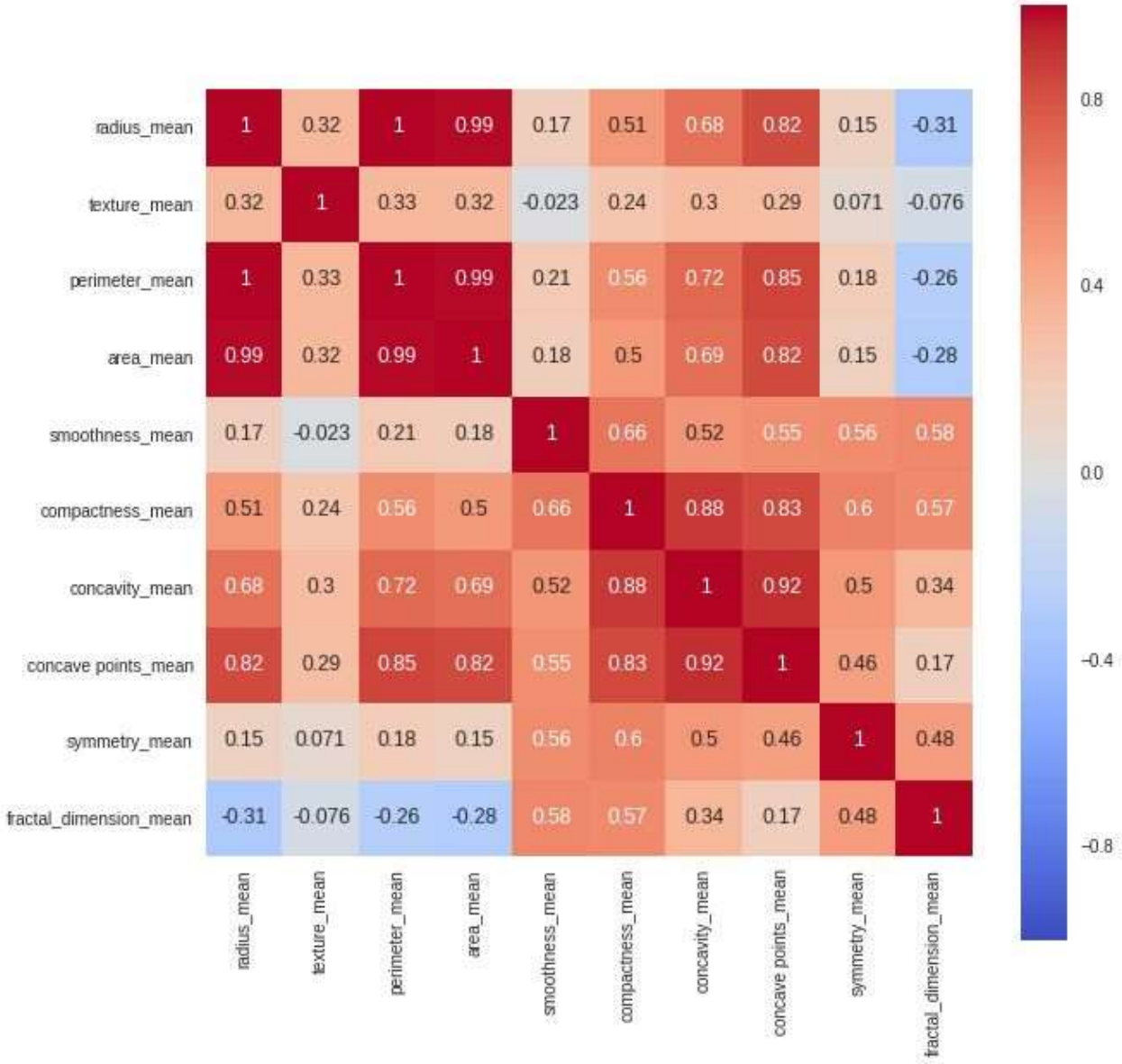
Until now, the classification has been done considering the whole data set (that is the 30 features). In order to have a deeper look at the data, we propose to display the 10 first features related to mean values. Three graphs are provided:

- Data feature distributions
- Data feature correlation
- Sample repartition according to each feature (blue points for Benign samples and red points for Malignant samples)

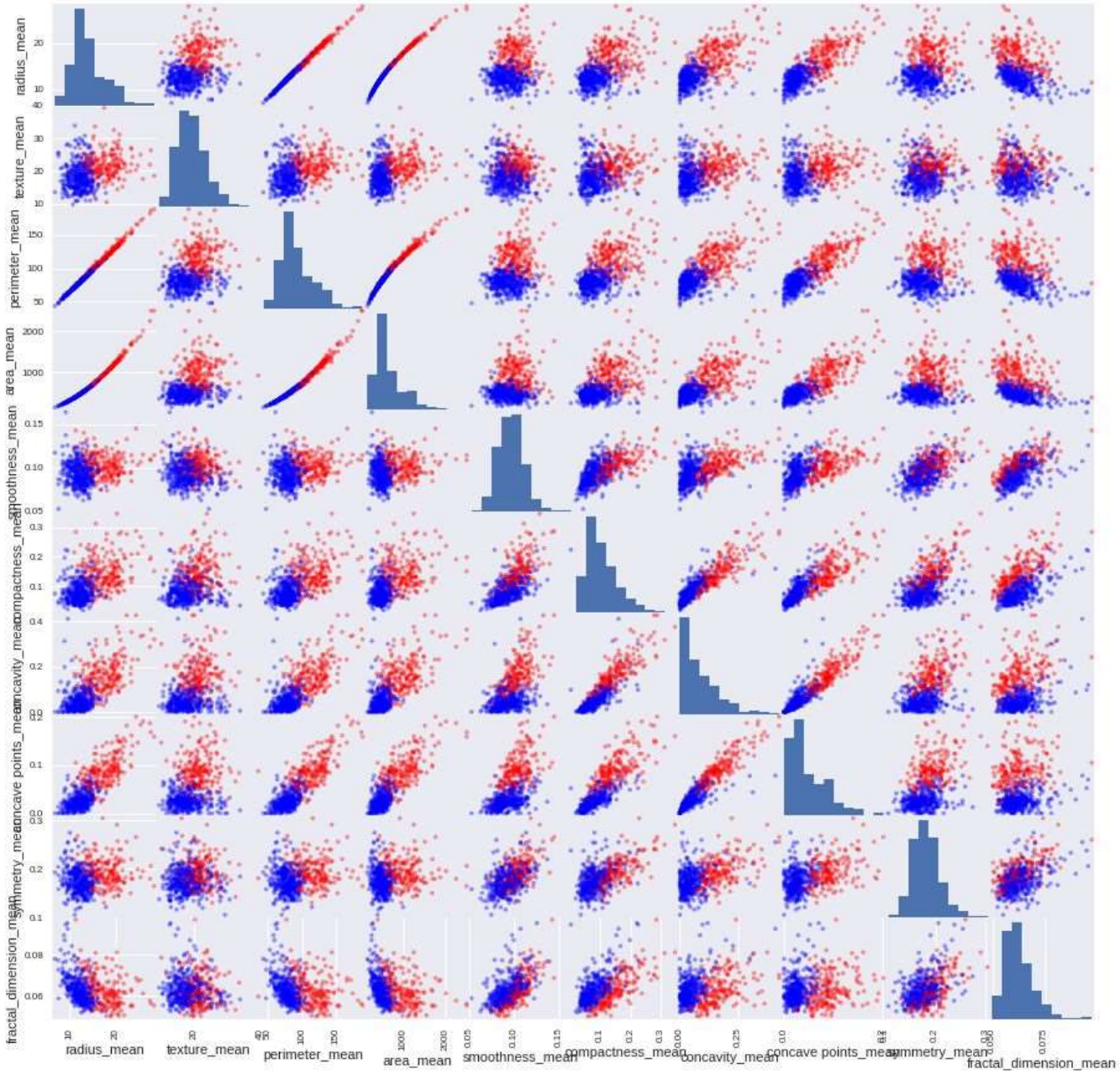
Recall here the name of the 10 first features. Analyze and comment each of the provided graphs.



Data distribution curves



Data correlation curves



Samples repartition according to each feature

4.4 Principal component analysis

Complete the ***Breast_cancer_student_code.py*** file in order to proceed to a PCA on the proposed data set. How many features are necessary to keep 99%, 95% and 90% of the total variance?

Proceed to a linear SVM classification after a PCA keeping successively 99%, 95%, 90% and 80% of the total information. Compute the accuracy, precision and recall scores in each case and compare the performances with the classification done before when considering the whole feature set (group all the figures in the following table). Comment the results.

% Variance	100	95	90	85	80
Nb Components					
Accuracy					
Precision					
Recall					

Proceed to PCA by keeping the two first main components only. Compute the classification performances and display the data set in the (component 1, component 2) 2D space.

5. Digit images clustering **(classroom work)**

The digit sklearn data set is made up of 1797 8x8 grey level images. Each image represents a hand written-digit from 0 to 9. Each image in the data set has its own label but in this exercise, the goal is to regroup all the samples related to a given digit by using the Kmeans clustering algorithm, so samples labels are not going to be used. The Kmeans algorithm in this case might be a good choice especially since the total number of clusters to be found is known (K=10).

5.1 Loading and visualizing the data

Run ***Digit_clustering_student_code.py*** in order to display some digit image samples. What are the data that are going to be used for clustering? What is the feature dimension?

5.2 Kmeans clustering

Run ***Digit_clustering_student_code.py*** to proceed to kmean clustering. The final digit centroids are displayed. Comment the obtained figure. Can you guess which digits are going to be difficult to distinguish?

Report and analyze the global accuracy and the confusion matrix.

5.3 Looking for the best K values

In this example, the K value is known but we are going to try to extract this value automatically using the K-elbow method. Complete the ***Digit_clustering_student_code.py*** file in order to proceed to kmean clustering with K from 1 to 30. For each segmentation, compute the total within variance. Plot the evolution of the computed variance w.r.t K. Comment the curve. If applying the k-elbow method, which K value is supposed to be the best? Conclusion.

[1] Machine Learning course Andrew Ng https://www.coursera.org/learn/machine-learning/home/welcome