UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

# DATA MINING

## Rapport sur le TP1

# «CLUSTERING»

## Encadrante :

**Zaineb CHELLY DAGDIA**

## Etudiant :

**Toufik GUENANE 21807955**

Ce rapport consiste de deux parties qui vont traiter une partie essentielle du Data Mining, celle du Data Preprocessing et puis du Data Clustering qu'on va voir dans la deuxième partie de ce rapport, aussi on verra la première partie qui va expliquer comment utiliser un outil de Data Mining qui est Weka pour le clustering.

# Part 1 : WEKA « CLUSTERING »

## 1-1 Introduction :

Sur cette première partie, on va travailler sur l'outil de Data Mining WEKA pour faire du Clustering sur l'un des dataset les plus connus celui décrivant les IRIS. Il est décrit par quatre attributs observables et la classe de la fleur.

Le but est d'induire la classe d'Iris connaissant les attributs numériques observables et cela à partir du Clustering.

a1. sepal length (cm)

a2. sepal width in cm

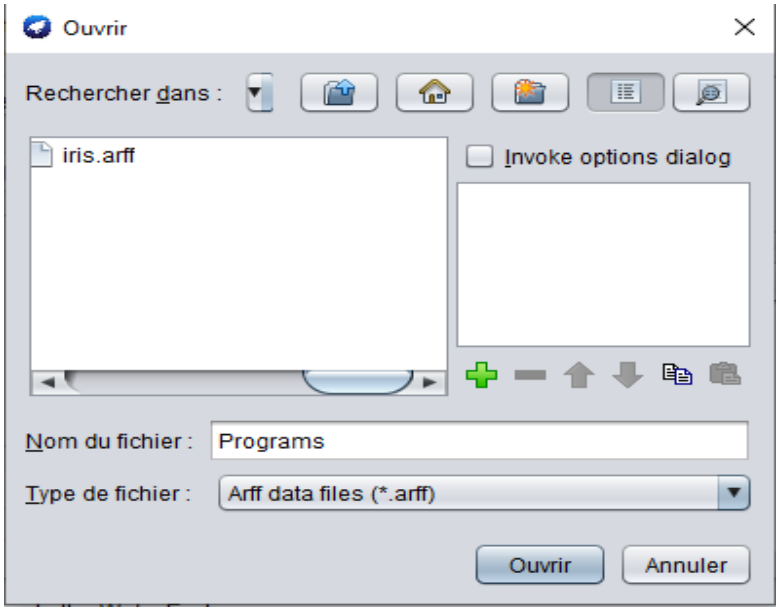
a3. petal length in cm

a4. petal width in cm

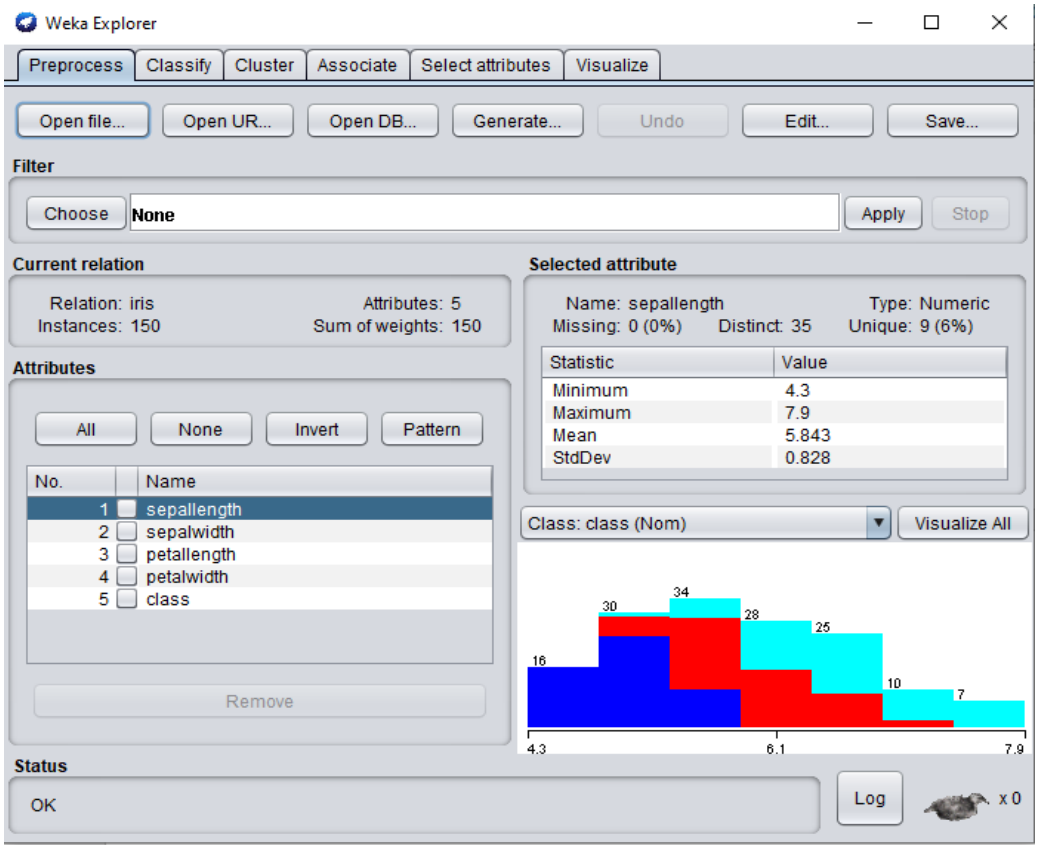
L'attribut label désigne la classe possède trois modalités : « Iris-Setosa », « IrisVersicolour » et « Iris-Virginica »

## 1-2 Importation des données sur Weka :

Pour cela, dans la fenêtre « Explorer » de Weka, ouvrir le fichier iris.arff.



Apres l'importation on se retrouve avec :




A partir de là, on peut voir que :

- On a trois classe différentes avec les trois couleurs juste en face (Bleu, rouge et turquoise) ;
- On a 5 attributs pour ce dataset avec 150 instances ;
- On a zéro valeur manquantes sur tous les instances de l'attribut 'sepallength' avec tous les statistiques de cette attribut comme MIN value, MAX value, Mean(moyenne) et StdDev (écart type). (Pareil pour les autres attributs) ;
- On a aussi une distribution gaussienne sur tous les attributs.

A partir de là, on a des un dataset sans valeurs manquantes avec une bonne distribution gaussienne, on pourra par la suite passer à la deuxième étape du clustering sur Weka.

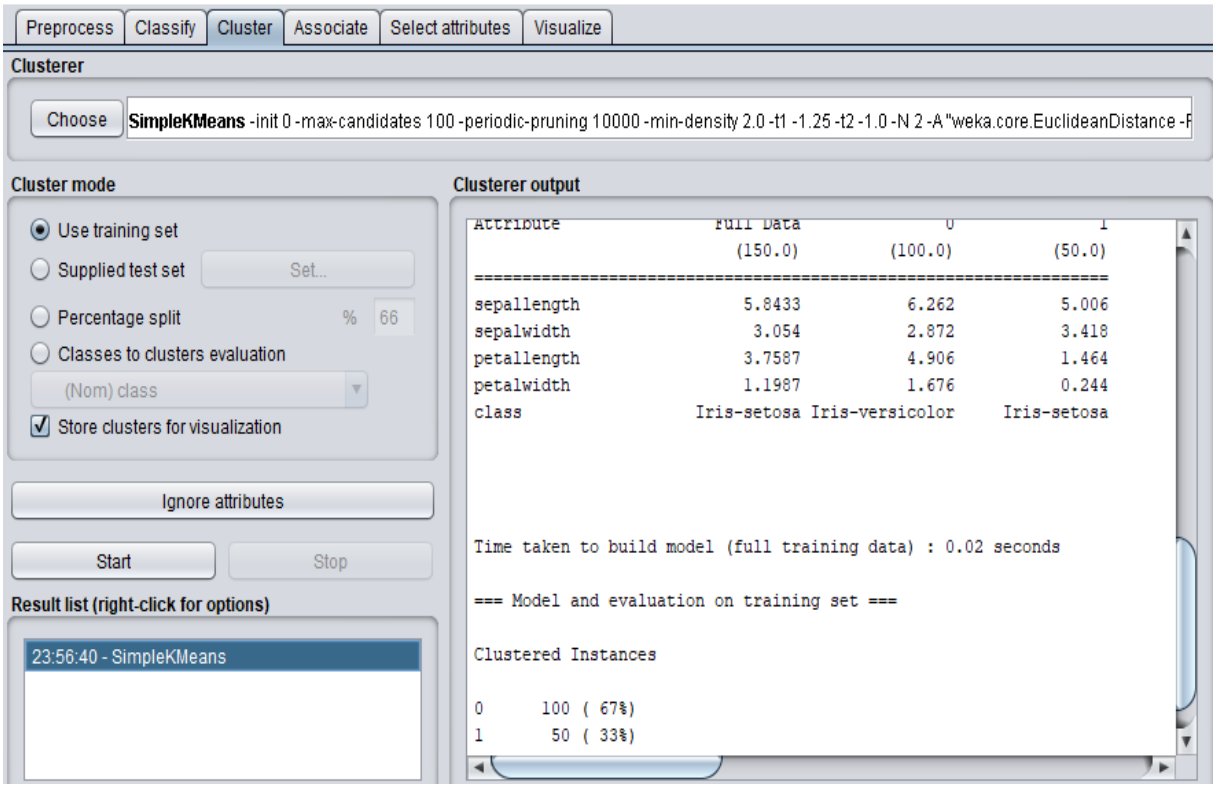
## 1-3 Clustering des données sur Weka :

Sur cette étude, on sera amené à utiliser trois algorithmes différents de Clustering, qui sont K-Means, EM et X-Means.

**1-3-1 K-Means :** On va lancer l'algorithme de KMeans avec les paramètres par défaut, puis on va prendre par défaut K=3, et on va faire un clustering sur les données qu'on de deux manières, la première avec l'attribut Class et la deuxième Sans Class.

**Default KMeans :**

On clique sur la fenêtre « Cluster », puis on choisit le SimpleKmeans sans rien touché dans les paramètres, et on clique sur Start :




On observe que l'algorithme a réussi à faire deux clusters : cluster0 qui contient 100 (67%) instances et cluster1 qui contient 50 (33%) instances, aussi on a :

Number of iterations: 7 //nombres d'itérations

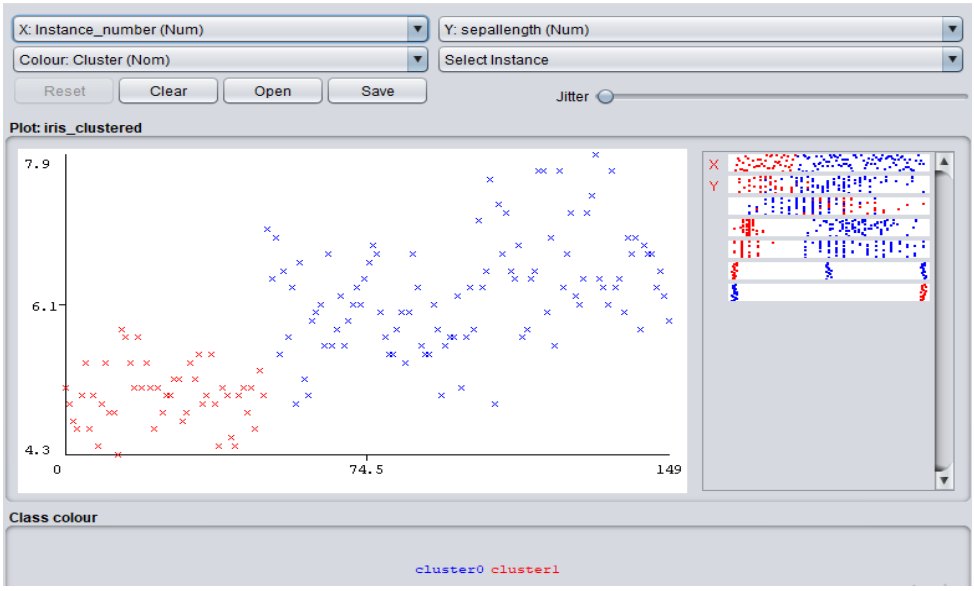
Within **cluster sum of squared errors : 62.1436882815797** // un cluster qui a une petite somme de carrés est plus compact qu'un cluster qui a une grande somme de carrés.

Initial starting points (random) : //Points initiaux prédites par le modèle de clustering

Cluster 0: 6.1,2.9,4.7,1.4,Iris-versicolor

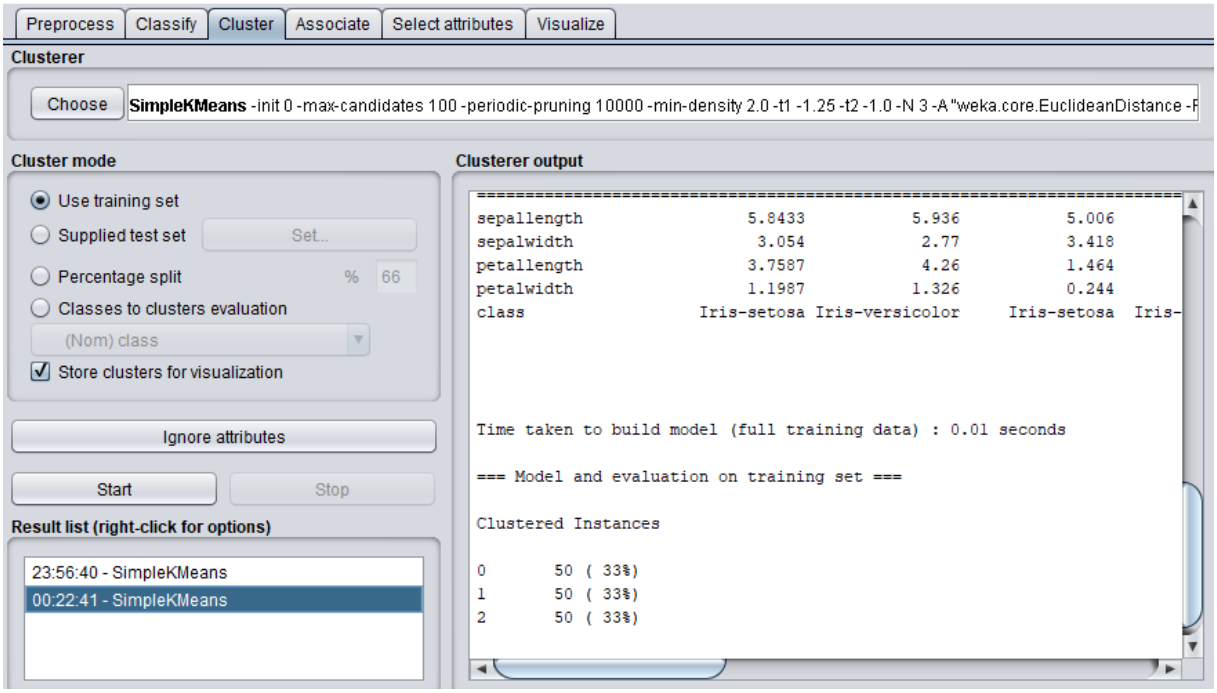
Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor

Par la suite on affiche le graphe de clustering pour voir les différents clusters :



**KMeans K=3 et avec class attribut :**

On clique sur la fenêtre « Cluster », puis on choisit le SimpleKmeans et on prend le nombre de clusters K=3, et on clique sur Start :




On observe que l'algorithme a réussi à faire trois clusters : cluster0 qui contient 50 (33%) instances, cluster1 qui contient 50 (33%) instances et cluster2 qui contient 50 (33%) instances, aussi on a :

Number of iterations: 3 //nombre iterations de l'algorithme

Within **cluster sum of squared errors: 7.817456892309574**

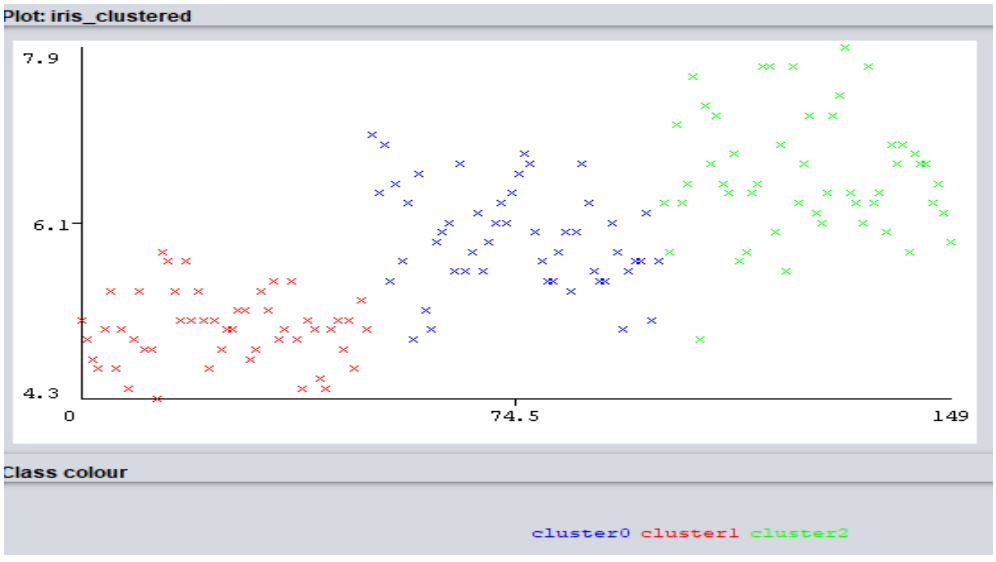
Initial starting points (random):// Points initiaux prédites par le modèle de clustering

Cluster 0: 6.1,2.9,4.7,1.4,Iris-versicolor

Cluster 1: 6.2,2.9,4.3,1.3,Iris-versicolor

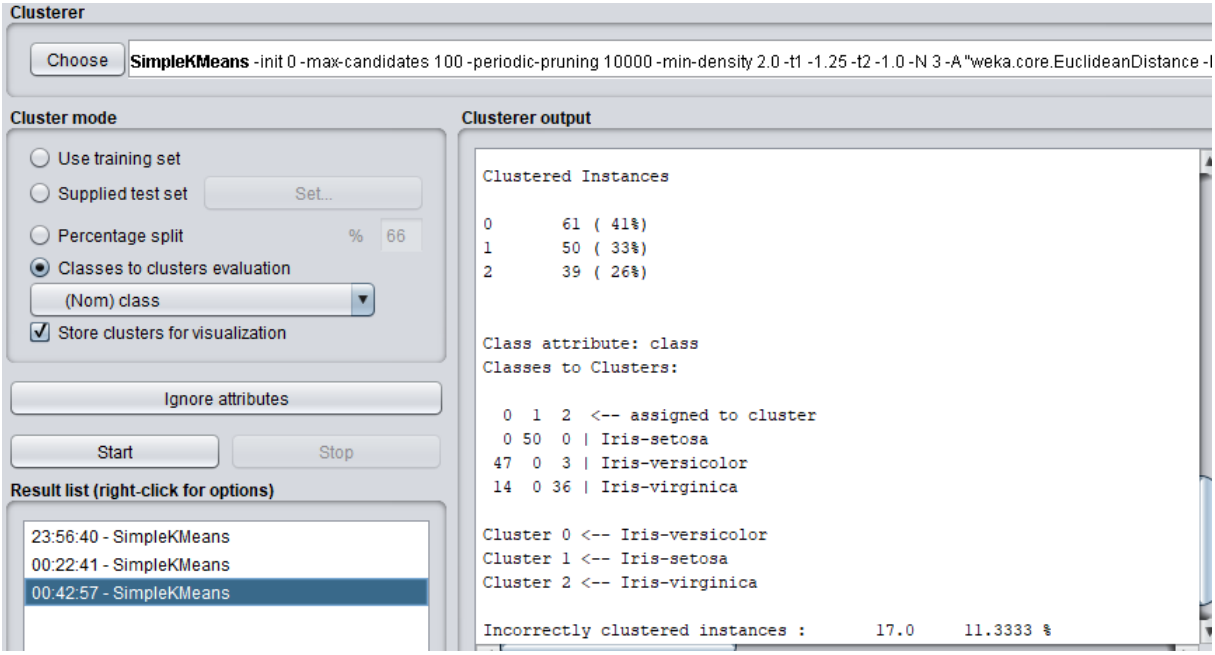
Cluster 2: 6.9,3.1,5.1,2.3,Iris-virginica

Par la suite on affiche le graphe de clustering pour voir les différents clusters :



**KMeans K=3 et sans class attribut :**

On clique sur la fenêtre « Cluster », puis on choisit le SimpleKmeans et on prend le nombre de clusters K=3 et on prend l'attribut class du dataset, et on clique sur Start :




On observe que l'algorithme a réussi à faire trois clusters : cluster0 qui contient 61 (41%) instances, cluster1 qui contient 50 (33%) instances et cluster2 qui contient 39 (26%) instances, aussi on a :

Number of iterations: 6

Within **cluster sum of squared errors: 6.998114004826762**

Initial starting points (random):

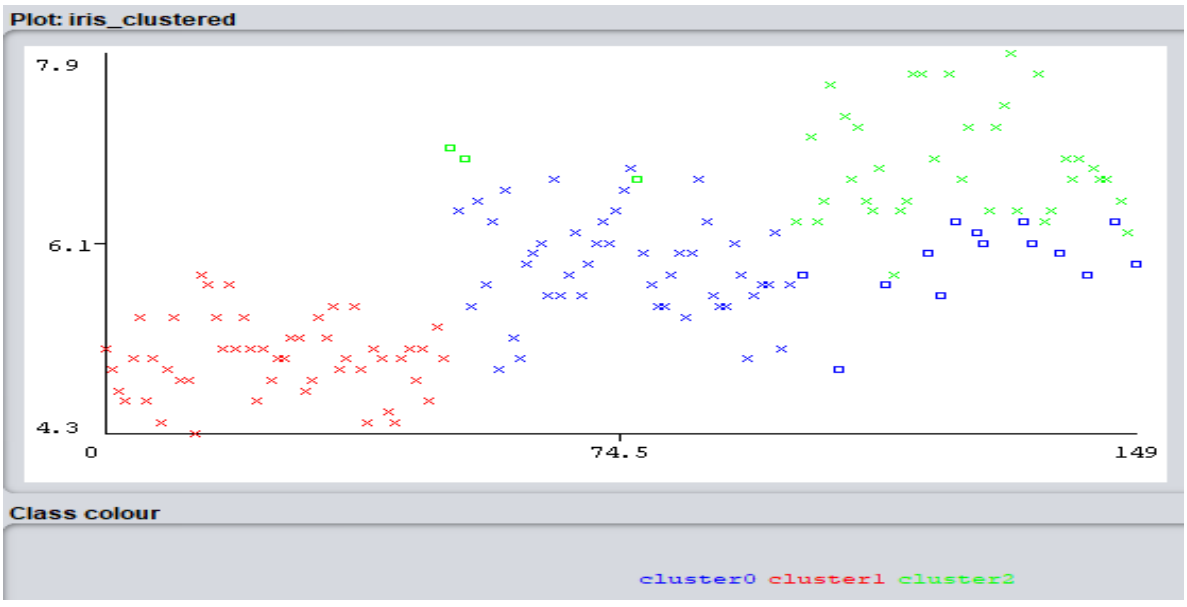
Cluster 0: 6.1,2.9,4.7,1.4

Cluster 1: 6.2,2.9,4.3,1.3

Cluster 2: 6.9,3.1,5.1,2.3

**Incorrectly clustered instances : 17.0 11.3333 %** //Le nombre d'instances mal clustered

Par la suite on affiche le graphe de clustering pour voir les différents clusters :

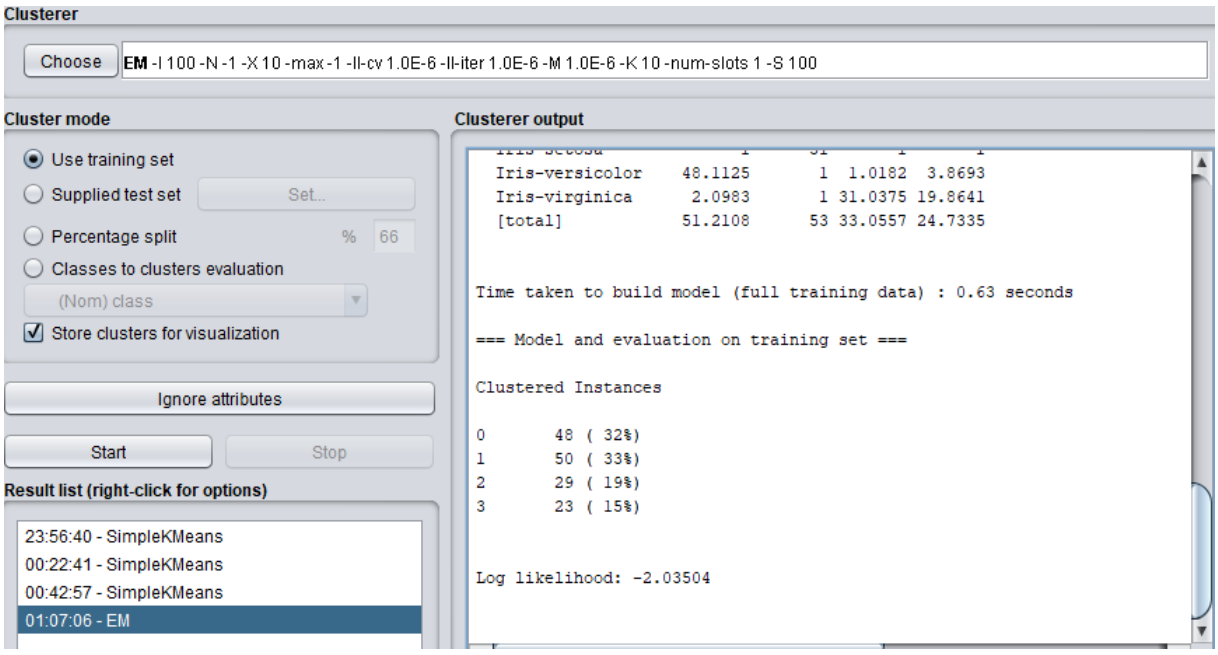


Contrairement aux deux premiers graphes qu'on a visualisés, on aperçoit qu'en plus d'avoir des croix qui représentent les bonnes classifications du modèle par rapport au label Class, on a des carrés qui représentent les mauvaises valeurs incorrectly clustered par rapport à son label initial de classification, ou on a trouvé 17 instances mal groupé.

**1-3-2 EM :** On va lancer l'algorithme de EM avec les paramètres par défaut, puis on va tuner un peu avec le nombre de clusters=3 et on va faire en deux cas avec Class et sans Class, afin de comparer ces résultats avec celles de KMeans effectués juste au-dessus.

**Default EM :**

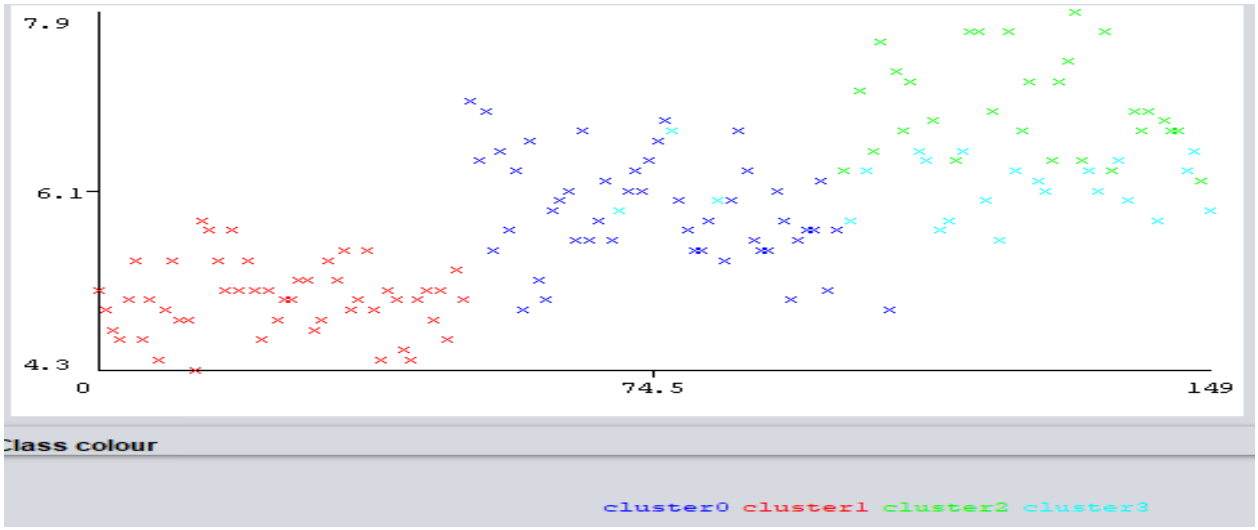
On clique sur la fenêtre « Cluster », puis on choisit le EM, et on clique sur Start :




On observe que l'algorithme a réussi à faire quatre clusters : cluster0 qui contient 48 (32%) instances, cluster1 qui contient 50 (33%) instances, cluster2 qui contient 29 (19%) instances et cluster3 qui contient 23 (15%) instances, aussi on a :

Log likelihood : -2.03504 // log probabilité : il s'agit de la probabilité de chaque observation étant donné l'étiquette de cluster attribuée. Si vous prenez le log de ceci, des valeurs négatives apparaissent naturellement, car les vraisemblances(likelihood) sont supposées être dans [0;1]. (Plus le log likelihood est grand plus il est bon).

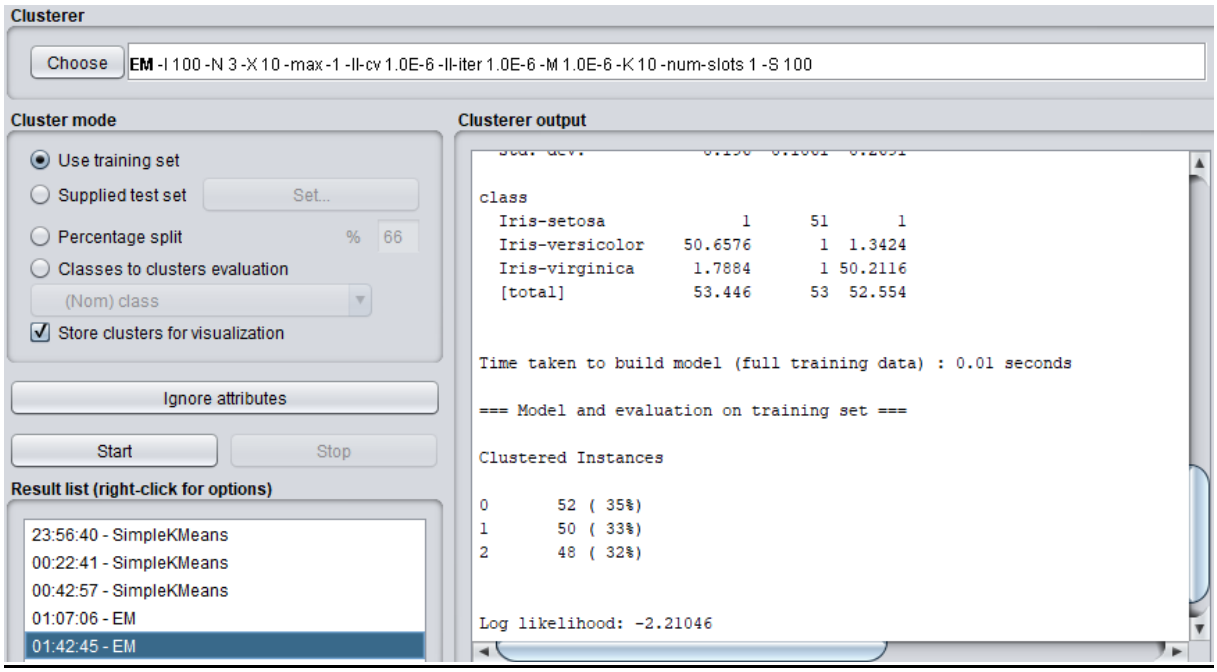
Par la suite on affiche le graphe de clustering pour voir les différents clusters :



**Default EM VS Classes réelles :** à partir des résultats trouvés de ce model EM (paramètres par default), on peut voir qu'il y'a une différence sur le nombre de clusters crées par rapport aux autres classes déjà existantes ou on a trois. Ici, dans le cas de EM on se retrouve avec 4 clusters.

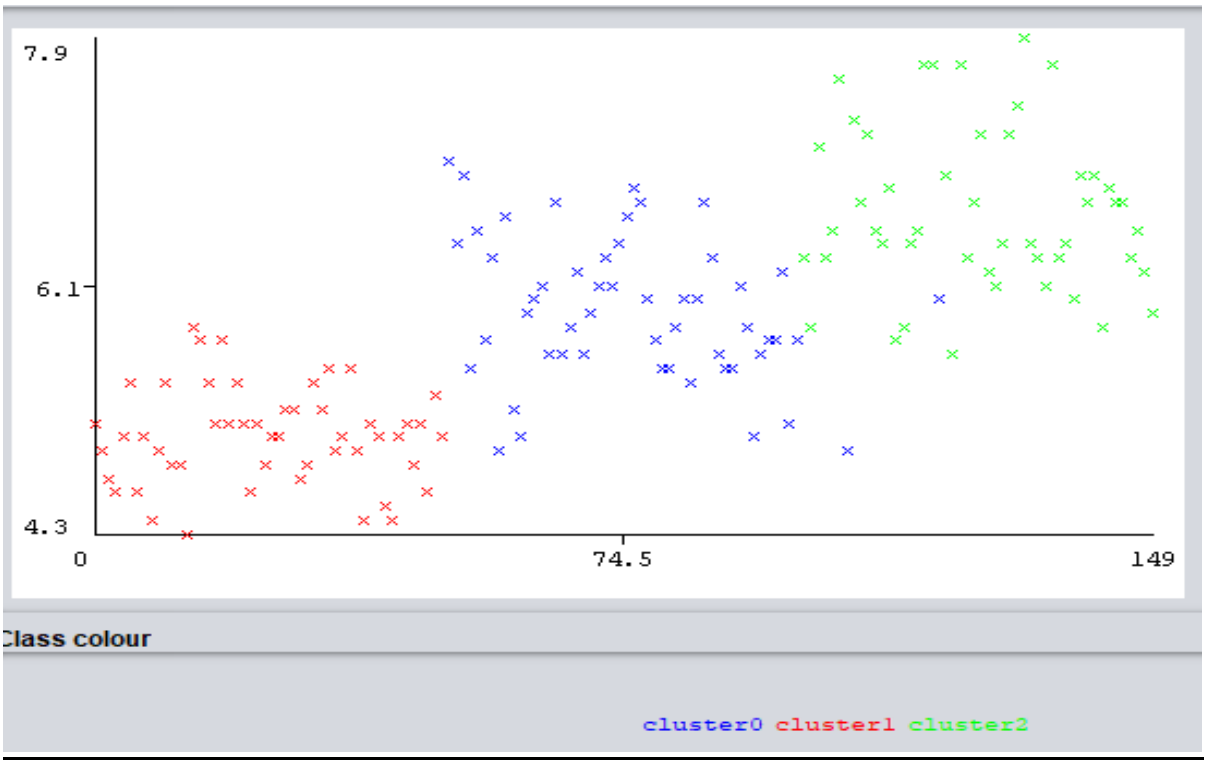
**EM with (number of clusters=3 and with class label):**

On clique sur la fenêtre « Cluster », puis on choisit le EM, et on prend le nombre de clusters=3 au lieu de -1(default pour cross validation), et on clique sur Start :



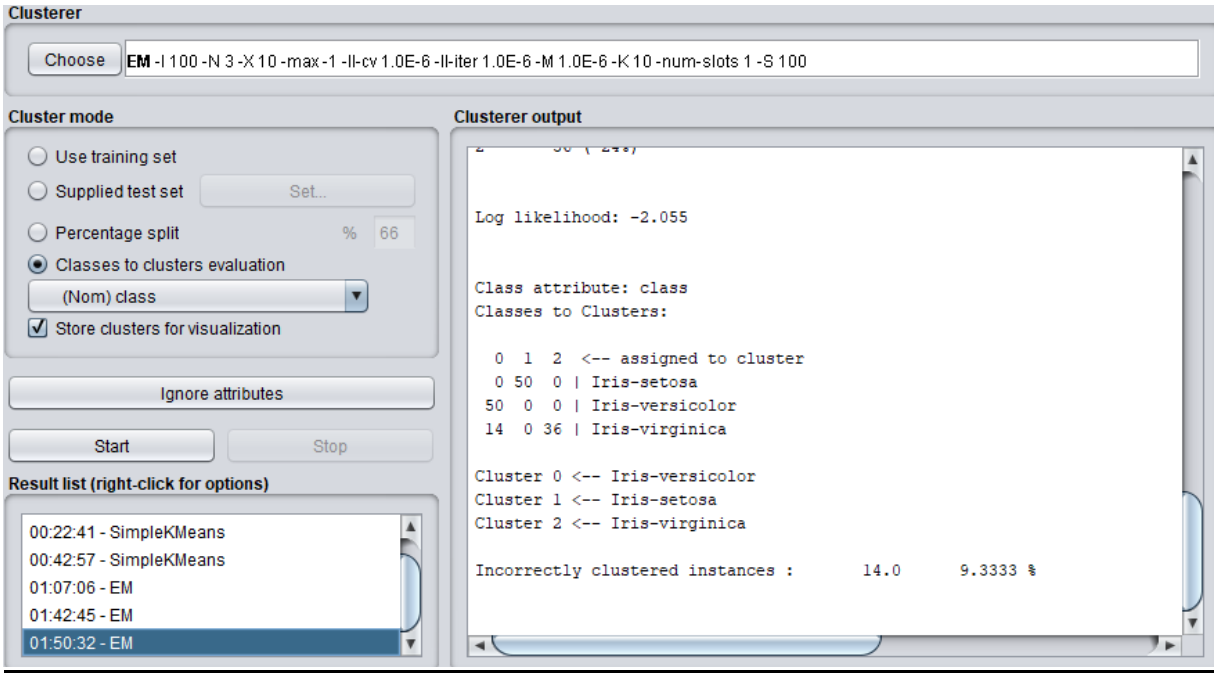

On observe que l'algorithme a réussi à faire quatre clusters : cluster0 qui contient 52 (35%) instances, cluster1 qui contient 50 (33%) instances et cluster2 qui contient 48 (32%), aussi on a : Log likelihood : -2.21046

Par la suite on affiche le graphe de clustering pour voir les différents clusters :



**EM with (number of clusters=3 and with no class label):**

On clique sur la fenêtre « Cluster », puis on choisit le EM, et on prend le nombre de clusters=3 au lieu de -1(default pour cross validation), on ne prend pas cette fois le label class pour ce modèle, et on clique sur Start :




On observe que l'algorithme a réussi à faire quatre clusters : cluster0 qui contient 64 (43%) instances, cluster1 qui contient 50 (33%) instances et cluster2 qui contient 36 (24%), aussi on a :

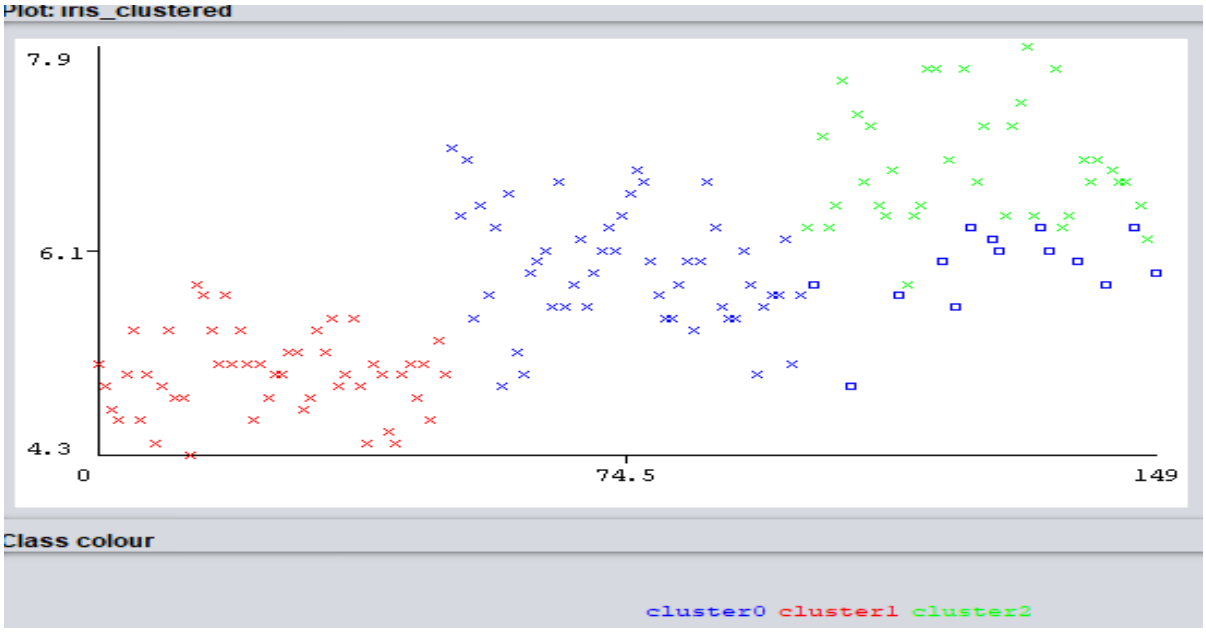
Clustered Instances

cluster0 64 ( 43%)
cluster1 50 ( 33%)
cluster2 36 ( 24%)

Log likelihood: -2.055

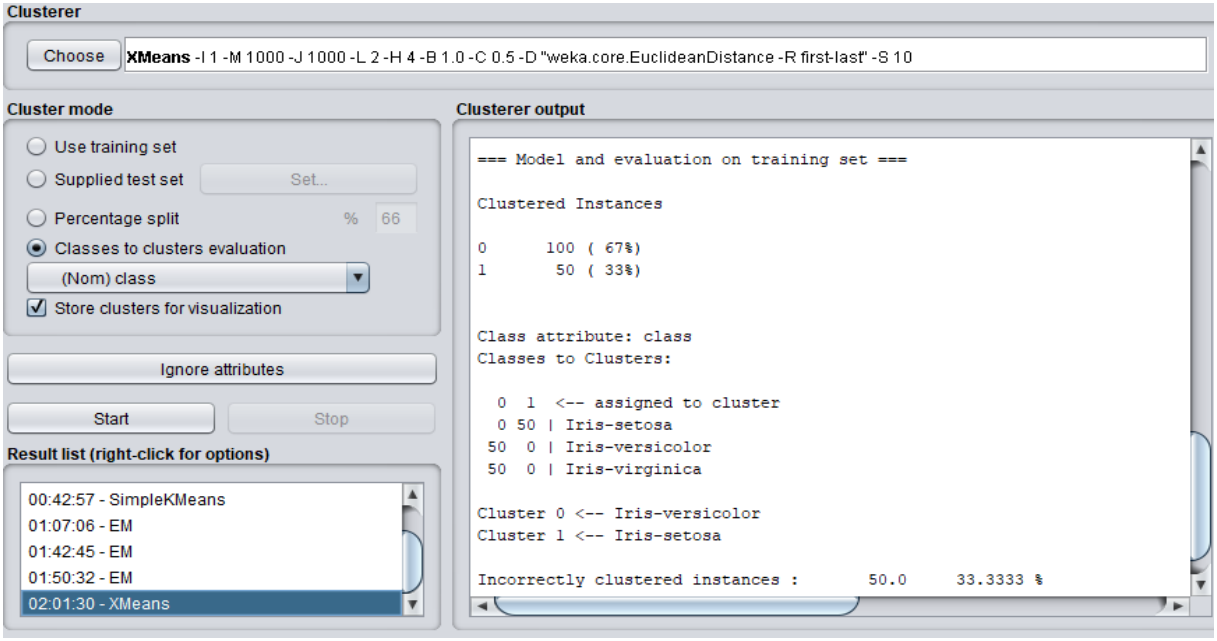
Incorrectly clustered instances: 14.0 9.3333 %

Par la suite on affiche le graphe de clustering pour voir les différents clusters :



Contrairement aux deux premiers graphes qu'on a visualisés, on aperçoit qu'en plus d'avoir des croix qui représentent les bonnes classifications du modèle par rapport au label Class, on a des carrés qui représentent les mauvaises valeurs incorrectly clustered par rapport à son label initial de classification, ou on a trouvé 14 instances mal groupé.

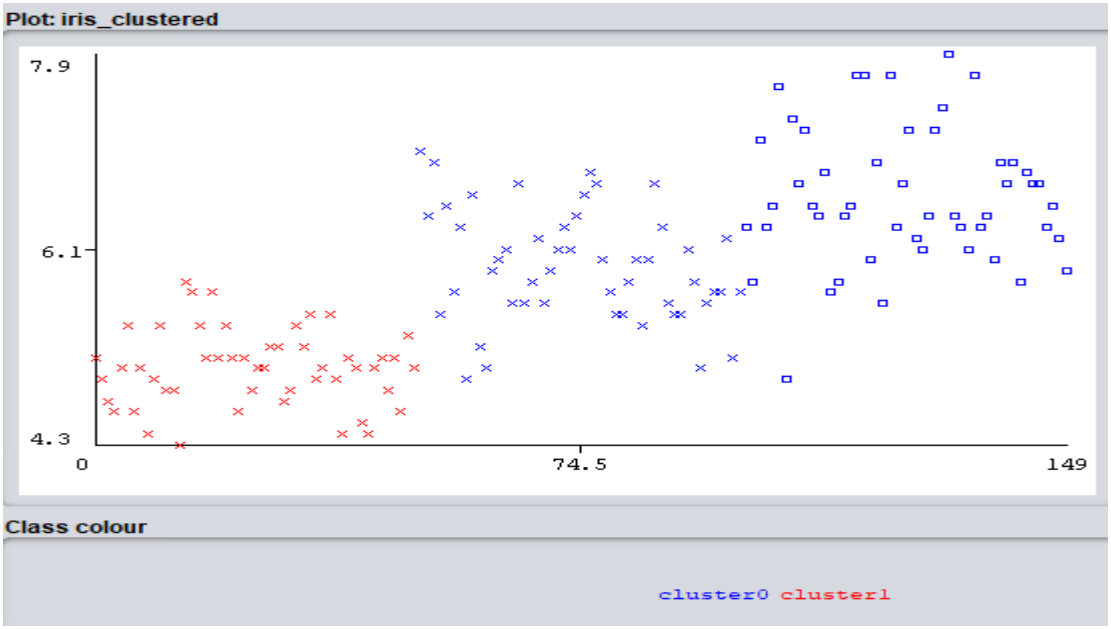
**1-3-3 XMeans :** On va lancer l'algorithme de XMeans avec les paramètres par défaut, qui sensé trouver le nombre de classes automatiquement.




On observe que l'algorithme a réussi à faire deux clusters : cluster0 qui contient 100 (67%) instances, cluster1 qui contient 50 (33%), aussi on a :

Clustered Instances
cluster0 100 ( 67%)
cluster1 50 ( 33%)
Incorrectly clustered instances : 50.0 33.3333 %

Par la suite on affiche le graphe de clustering pour voir les différents clusters :



Sur ce graphe qu'on a visualisé, on aperçoit qu'en plus d'avoir des croix qui représentent les bonnes classifications du modèle par rapport au label Class, on a des carrés qui représentent les mauvaises valeurs incorrectly clustered par rapport à son label initial de classification, ou on a trouvé 50 instances mal groupé.

## 1-4 Comparaisons des résultats des algos de Clustering sur Weka :

**Kmeans(Default) Vs EM(Default) Vs Xmeans(Default) :** Comparaisons des trois algos avec des paramètres par default.

On voit à partir de cette expérience, que les trois algos ont donné des résultats inattendus, surtout pour Xmeans qui est sensé de retourner le bon nombre de clusters automatiquement, il retourne 2 clusters identiques à ceux de Kmeans par default ; aussi on Em par default qui à créer 4 clusters au lieu de 2.

**Kmeans (K=3 et avec Class) Vs EM (K=3 et avec Class) :** Comparaisons des deux algos avec des paramètres par k=3 et avec le label Class.

On voit à partir de cette expérience, que les deux algos ont donné des résultats presque identiques, ou on remarque que Kmeans à donner exactement la bonne clustering des classes réelles par au EM qui a bien clusterer mais a mal groupé deux instances dans le premier cluster0 ou on voit 52 instances au lieu de 50 comme les données réelles des classes.

**Kmeans (K=3 et sans Class) Vs EM (K=3 et sans Class) Vs Xmeans(Default) :** Comparaisons des trois algos avec des paramètres par K=3 et sans le label Class pour Kmeans et EM, et Xmeans reste par default.

On voit à partir de cette expérience, que les deux algos Kmeans et EM ont donné des résultats presque identiques, ou on remarque que EM prend légèrement l'avance sur Kmeans sur au niveaux de la précision de clustering, ou on voit que Kmeans a mal groupé 17 (11.33%) instances et que EM a juste mal groupé 14 (9.33%) instances. On a aussi Xmeans qui devait donner automatiquement le meilleur des résultats, mais il passe complétement à coté déjà avec deux clusters avec un taux d'erreurs de 1/3 ou il a mal groupé 50 instances.

# Part 2 : Clustering with Scikit-learn

## 2-1 Introduction:

L'une des maladies les plus diagnostiqué chez les femmes dans le monde est le cancer du sein. Une femme sur 9 sera atteinte de ce cancer au cours de sa vie et 1 femme sur 27 en mourra. Le plus souvent, le cancer du sein survient après 50 ans. Le taux de survie 5 ans après le diagnostic varie de 80 % à 90 %, selon l'âge et le type de cancer. Afin de prévenir les risques de ces maladies et pour mieux se protéger, on doit comprendre et apprendre comment les détecter d'une manière efficace et sécurisé. Cette recherche vise à identifier les facteurs de risque de maladie cancer de sein les plus pertinents ainsi qu'à prédire le risque global à l'aide des algorithmes de clustering qui ont pour but de grouper et prédire si un groupe de patients présente un risque de développer une future maladie cancéreuse ou pas.

# 2-2 Préparation et pre-processing des données :

## 2-2-1 Breast Cancer Wisconsin Dataset :

L'étude de la maladie de cancer de sein va se faire sur le dataset **Breast Cancer Wisconsin (Original) Data Set** qui se retrouve dans le repertoire de UCI Machine Learning.

## 2-2-1-1 Information sur le Dataset :

Dans ce dataset, on retrouve des données qui sont sur 699 instances et 11 attributs, et parmi ces attributs :

Attribut Information :

-- 1. ID : est l'identifiant d'une instance (Sample code number)

-- 2. Clump_Thickness: Épaisseur de touffe prend des valeurs entre 1 - 10

-- 3. Uniformity_of_Cell_Size: Uniformité de la taille des cellules prend des valeurs entre 1 - 10

-- 4. Uniformity_of_Cell_Shape: Uniformité de la forme des cellules prend des valeurs entre 1 - 10

-- 5. Marginal_Adhesion:Adhérence marginale prend des valeurs entre 1 - 10

-- 6. Single_Epithelial_Cell_Size: Taille de cellule épithéliale unique prend des valeurs entre 1 - 10

-- 7. Bare_Nuclei: Un noyau dans une préparation cytologique qui est pratiquement dépourvu de cytoplasme qui prend des valeurs entre 1 - 10

-- 8. Bland_Chromatin:Chromatine fade qui prend des valeurs entre 1 - 10

-- 9. Normal_Nucleoli:Normal Nucléoles qui prend des valeurs entre 1 - 10

-- 10. Mitoses: 1 - 10

-- 11. Class: (2 for benign, 4 for malignant)

## 2-2-2 Data Understanding:

Avant d'effectuer l'étape de data pre-processing, on doit d'abord comprendre les données, cela peut nous aider à mieux appliquer nos connaissances en data pre-processing, car on a pu comprendre nos données et pu éventuellement trouver des anomalies dans nos données qu'on va traiter par la suite dans la partie de data pre-processing.

Cette étape de data understanding se compose de deux étapes, mais avant de se lancer on a remarqué que le dataset avait des valeurs manquantes ou on aperçoit qu'elles étaient remplacées par '?' au lieu de NaN(null).

On était donc obligés de transformés ces ' ? ' en NaN avant de commencé la compréhension des données, afin de réaliser les différents calculs :

***Transformation de '?' vers NaN et detection des valeurs manquantes***

```
breast_DF = breast_DF.replace('?',np.nan)
breast_DF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 699 entries, 0 to 698
Data columns (total 11 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   ID                           699 non-null    int64
 1   Clump_Thickness              699 non-null    int64
 2   Uniformity_of_Cell_Size      699 non-null    int64
 3   Uniformity_of_Cell_Shape     699 non-null    int64
 4   Marginal_Adhesion            699 non-null    int64
 5   Single_Epithelial_Cell_Size  699 non-null    int64
 6   Bare_Nuclei                  683 non-null    object
 7   Bland_Chromatin              699 non-null    int64
 8   Normal_Nucleoli              699 non-null    int64
 9   Mitoses                      699 non-null    int64
 10  Class                        699 non-null    int64
dtypes: int64(10), object(1)
memory usage: 60.2+ KB
```

Figure 1 – Transformation des ' ?' en NaN

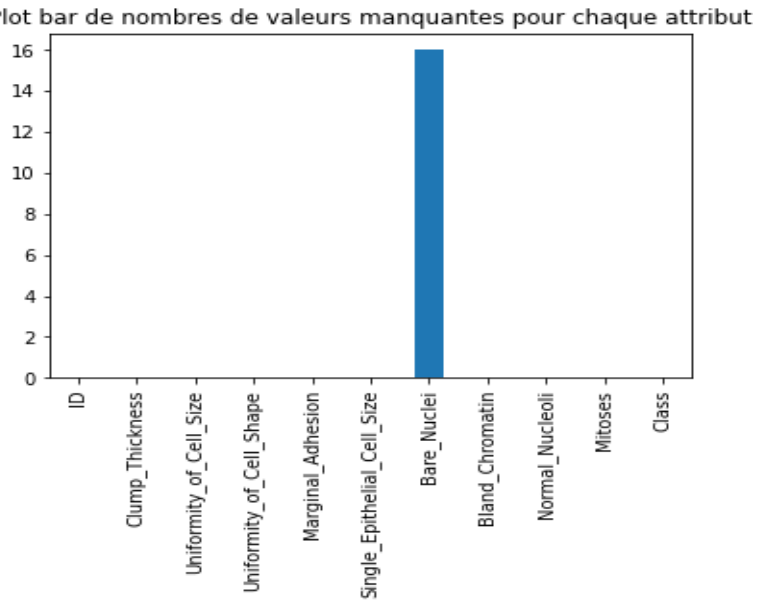



Figure 2 –Plot bar des nombres de valeurs manquantes pour chaque attribut

D'après ces deux figures, on aperçoit qu'on a des données manquantes sur la colonne "Bare_Nuclei" ou on a 16 valeurs vides manquantes non renseignés, qu'on va par la suite traiter dans une prochaine partie.

## 2-2-2-1 Mesures de tendance centrale :

Dans cette étape, on va voir les différentes mesures comme : mean, mean trimmed, median, min, max, mode et distribution de chaque attribut, et on va voir la distribution de la maladie sur label Class.

**Remarque :** Pour le code de chaque figure suivante, je vous invite à aller consulter le NoteBook Breast_Cancer pour plus d'informations.

**Mean**

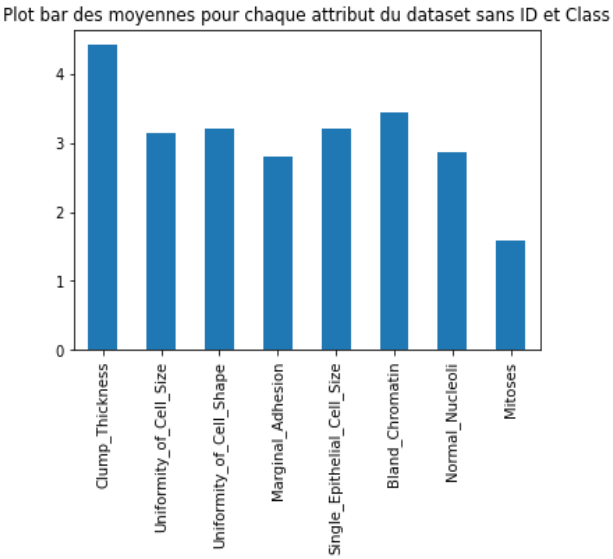



Figure 3 – Plot bar des moyennes pour chaque attribut

**Mean Trimmed**

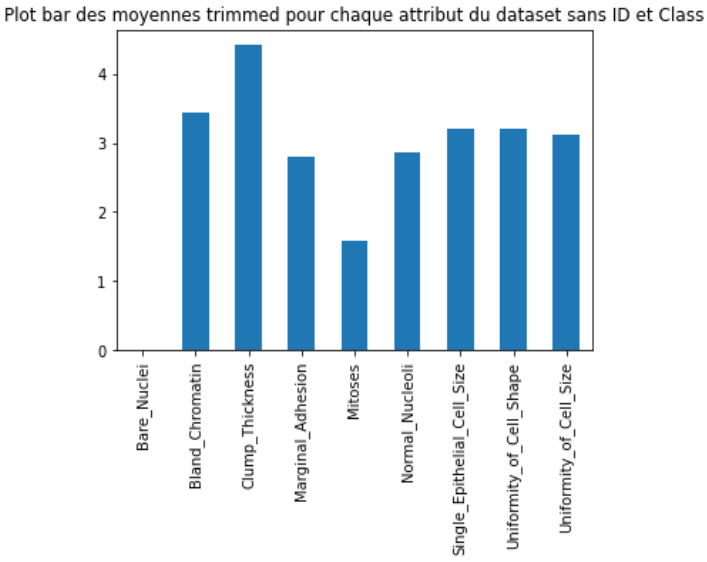
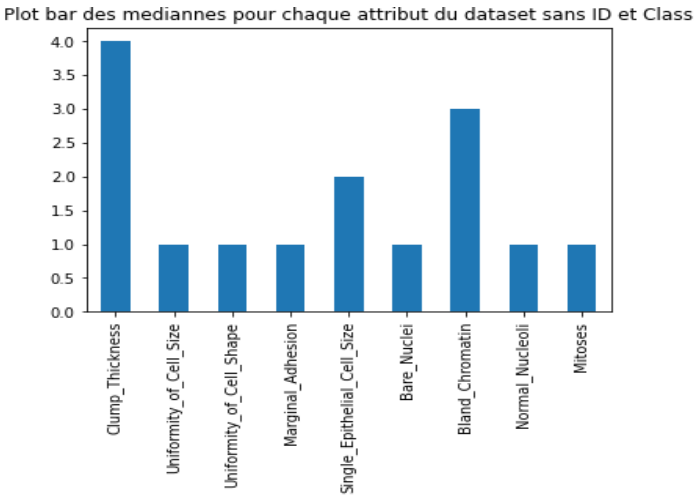



Figure 4 –Plot bar des moyennes coupées d'extrémités pour chaque attribut

A partir des deux figures on ne voit pas vraiment de différence entre les deux moyennes calculés pour chaque attribut de dataset.

**Median**




```
Clump_Thickness                 4.0
Uniformity_of_Cell_Size         1.0
Uniformity_of_Cell_Shape        1.0
Marginal_Adhesion               1.0
Single_Epithelial_Cell_Size     2.0
Bare_Nuclei                     1.0
Bland_Chromatin                 3.0
Normal_Nucleoli                 1.0
Mitoses                         1.0
```

Figure 5 –Plot bar des médianes pour chaque pour chaque attribut

On peut voir que la plus grande médiane est celle de Clump_Thickness, et la plupart des médianes des autres attributs sont à 1.

**Mode** : Le mode d'un ensemble de valeurs est la valeur qui apparaît le plus souvent. Il peut s'agir de plusieurs valeurs.

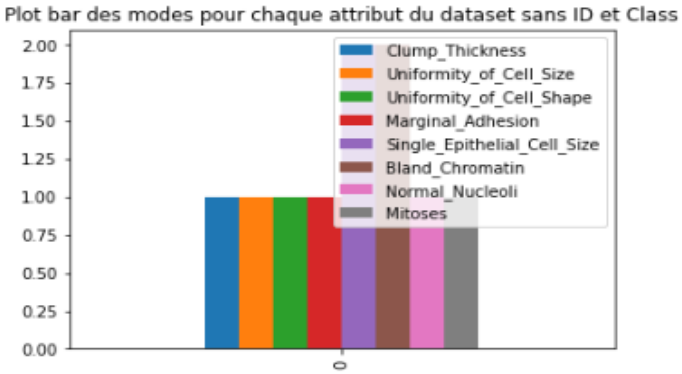



Figure 6 –Plot bar des modes pour chaque pour chaque attribut

On remarque à partir de cette figure que la valeur 1 est une valeur qui apparait souvent dans la plupart des valeurs des attributs, et il y'a que Single_Epithelial_Cell_Size et Bland_Chromatin qui ont des valeurs récurrentes égalent à 2.

**Min :** **Max :**

| Min de chaque attribut : | | Max de chaque attribut : | |
|---|---|---|---|
| Clump_Thickness | 1 | Clump_Thickness | 10 |
| Uniformity_of_Cell_Size | 1 | Uniformity_of_Cell_Size | 10 |
| Uniformity_of_Cell_Shape | 1 | Uniformity_of_Cell_Shape | 10 |
| Marginal_Adhesion | 1 | Marginal_Adhesion | 10 |
| Single_Epithelial_Cell_Size | 1 | Single_Epithelial_Cell_Size | 10 |
| Bland_Chromatin | 1 | Bland_Chromatin | 10 |
| Normal_Nucleoli | 1 | Normal_Nucleoli | 10 |
| Mitoses | 1 | Mitoses | 10 |

On peut voir que le Min de tous les attributs est à 1 et que tous leur Max est à 10.

**Distribution :**

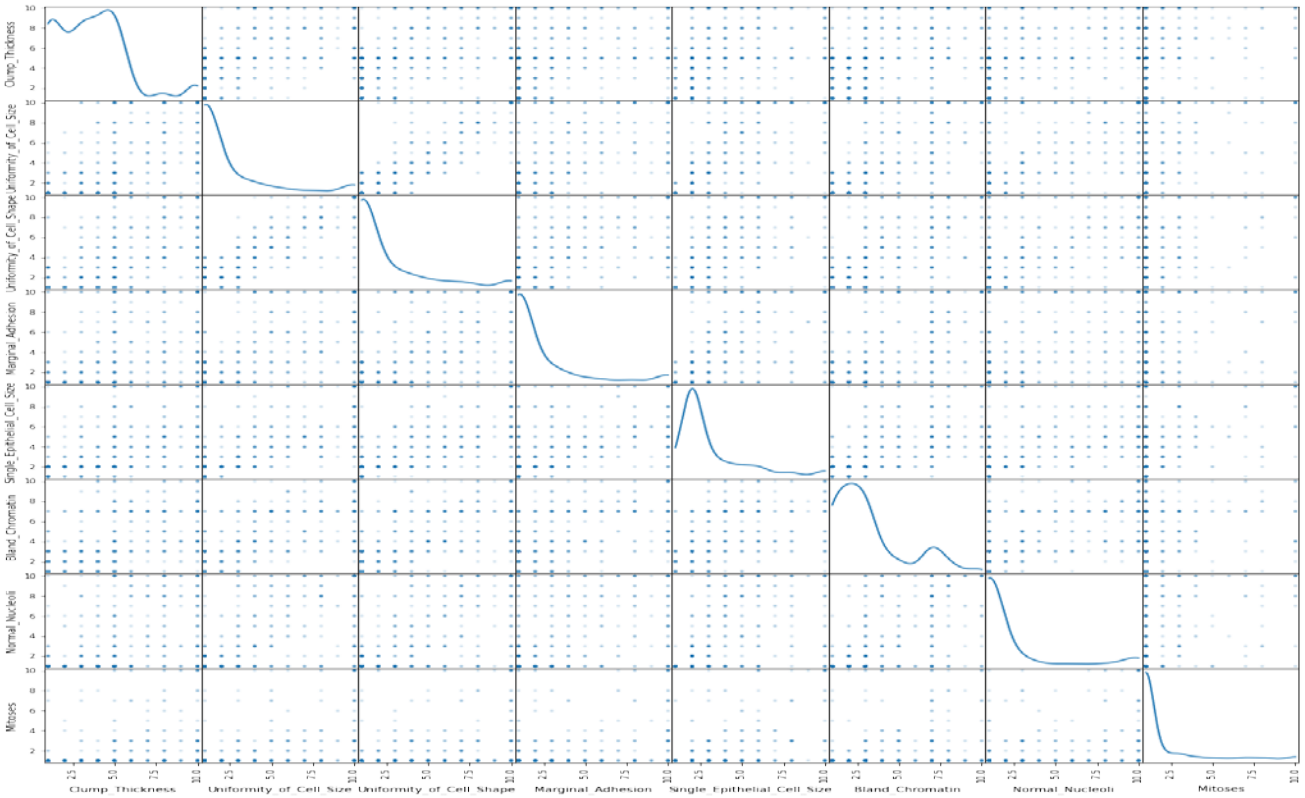



Figure 6 –Matrice de graphes de distribution des valeurs des attributs

A partir de cette figure, on aperçoit que les attributs suivants ne sont pas en distribution gaussienne, par la suite il faut les transformer, on verra çà dans une prochaine étape dans la data transformation.

**Distribution de la maladie sur label Class :**

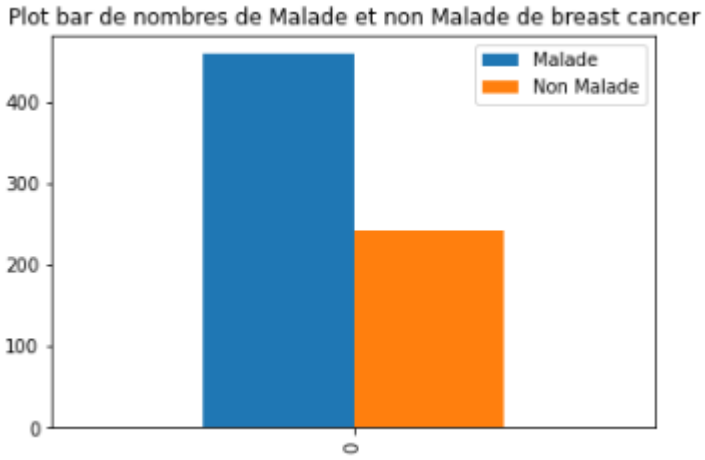



Figure 7 –Plot bar de nombres de malade et non malades de breast cancer

```
Nombre de gens atteintes de cancer brea
st Disease: 458
Nombre de gens non atteintes de cancer
breast Disease: 241
```

On remarque à partir de cette figure, que les classes sont imbalanced ; ça implique que le dataset est imbalanced, et qu'on a besoin de traiter ce type de problème qu'on va voir dans les prochaines étapes.

## 2-2-2-2 Mesures de la dispersion des données

Dans cette étape, on va voir les différentes mesures comme : quartiles, Interquartiles, Variance et standard déviation.

**Quartiles :**

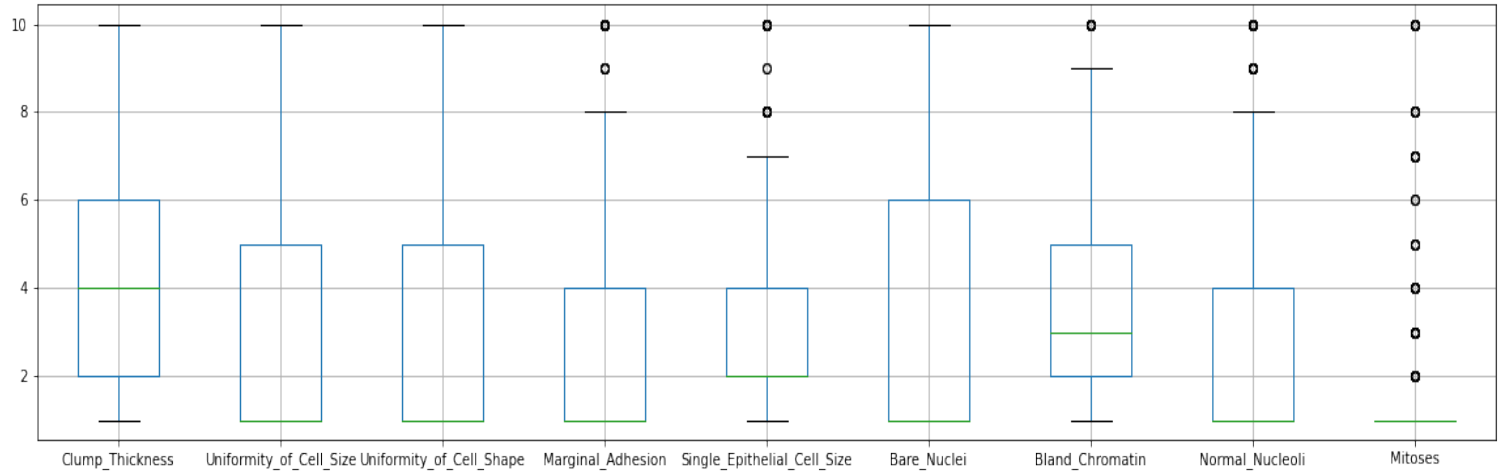



Figure 8 –Box Plot des différents attributs

A partir de cette figure on remarque qu'ils ne y'a pas de vrai outliers même ci qu'on a quelques données qui appartiennent pas aux différents Box plots, et cela revient au faite que les instances prennent des valeurs de 1 à 10 pour les valeurs attributs de chaque instance, donc cela nous évite de travailler sur les problèmes outliers.

**Interquartiles, Variance et standard déviation :**

| Interquartiles | Variance | Standard Déviation |
|---|---|---|
| Les interquartiles du dataset sont : | La variance des attributs du data set | La deviation standard des attributs du data set |
| Clump_Thickness 4.0 | Clump_Thickness 7.928395 | Clump_Thickness 2.815741 |
| Uniformity_of_Cell_Size 4.0 | Uniformity_of_Cell_Size 9.311403 | Uniformity_of_Cell_Size 3.051459 |
| Uniformity_of_Cell_Shape 4.0 | Uniformity_of_Cell_Shape 8.832265 | Uniformity_of_Cell_Shape 2.971913 |
| Marginal_Adhesion 3.0 | Marginal_Adhesion 8.153191 | Marginal_Adhesion 2.855379 |
| Single_Epithelial_Cell_Size 2.0 | Single_Epithelial_Cell_Size 4.903124 | Single_Epithelial_Cell_Size 2.214300 |
| Bare_Nuclei NaN | Bare_Nuclei 13.277695 | Bare_Nuclei 3.643857 |
| Bland_Chromatin 3.0 | Bland_Chromatin 5.945620 | Bland_Chromatin 2.438364 |
| Normal_Nucleoli 3.0 | Normal_Nucleoli 9.324680 | Normal_Nucleoli 3.053634 |
| Mitoses 0.0 | Mitoses 2.941492 | Mitoses 1.715078 |

## 2-2-3 Data Preprocessig :

### 2-2-3-1 Data cleaning:

Comme c'était le cas à partir de la partie de data understanding, on a été confronté à certains problèmes, notamment le problème de des valeurs manquantes sur la colonne "Bare_Nuclei".

Le but ici, dans cette étape est de remplir les valeurs manquantes par d'autres valeurs non nulle et qui ont un contexte avec les données initiales qu'on a, dans ce cas on va faire une data imputation, et cela à partir de l'algorithme de l'imputation des données qui est : **KNNImputer**.

Dans notre exemple, on a au depart :

```
Nombre de valeurs manquantes pour chaque attribut de dataset:
        Clump_Thickness: 0
        Uniformity_of_Cell_Size: 0
        Uniformity_of_Cell_Shape: 0
        Marginal_Adhesion: 0
        Single_Epithelial_Cell_Size: 0
        Bare_Nuclei: 16
        Bland_Chromatin: 0
        Normal_Nucleoli: 0
        Mitoses: 0
        Class: 0
```

En appliquant ce code, avec KNNImputer on aura ce qui suit :

```
import numpy as np
from numpy import isnan
from sklearn.impute import KNNImputer

# define imputer
imputer = KNNImputer()
# fit on the dataset
imputer.fit(missing_DF)
# transform the dataset
Xtrans = imputer.transform(missing_DF)
# print total missing
print('Missing instances of the dataset: %d' % sum(isnan(Xtrans).flatten()))

no_missing_breast_DF = pd.DataFrame(Xtrans, columns = [
                    'Clump_Thickness', 'Uniformity_of_Cell_Size', 'Uniformity_of_Cell_Shape','Marginal_Adhesion',
                    'Single_Epithelial_Cell_Size','Bare_Nuclei','Bland_Chromatin','Normal_Nucleoli','Mitoses','Class'])

#after imputation
no_missing_breast_DF
```

```
Missing instances of the dataset: 0
```

A partir de ceci, on remarque qu'on a plus de valeurs nulles ou manquantes.

**2-2-3-2 Data Intégration :**

Etant donné qu'on travaille sur un seul dataset, ce n'est pas nécessaire de faire cette étape.

**2-2-3-3 Data Transformation :**

Comme c'était le cas à partir de la partie de data understanding, on a été confronté à beaucoup de problèmes, notamment sur la non distribution des données en loi gaussienne, aussi le problème de imbalanced data car les labels classification ne sont pas equally représenté.

**Imbalanced data :** Comme on peut le voir sur la figure 7, on a :

```
Nombre de gens atteintes de cancer breast Disease : 458
Nombre de gens non atteintes de cancer breast Disease : 241
```

Cela indique que les données sont imbalanced, et pour résoudre çà on a appliqué la technique de ove r sampeling de la classe malignant, comme suit dans ce code :

```
#Imbalanced Data
#Pour ce probleme on vas utiliser Oversampling
from sklearn.utils import resample

#create two different dataframe of majority and minority class
df_majority = no_missing_breast_DF[(no_missing_breast_DF['Class'] == 2)]
df_minority = no_missing_breast_DF[(no_missing_breast_DF['Class'] == 4)]

# upsample minority class
df_minority_upsampled = resample(df_minority,
                                 replace=True,     # sample with replacement
                                 n_samples= 458, # to match majority class
                                 random_state=42)  # reproducible results
# Combine majority class with upsampled minority class
breast_DF_balanced = pd.concat([df_minority_upsampled, df_majority])
print(breast_DF_balanced['Class'].value_counts())

print("Bar plot du dataset balanced(label=Class)")
sns.countplot(breast_DF_balanced['Class'])
```

```
2.0    458
4.0    458
```

Par la suite on a eu le résultat, ou chaque classe a 458 instances.

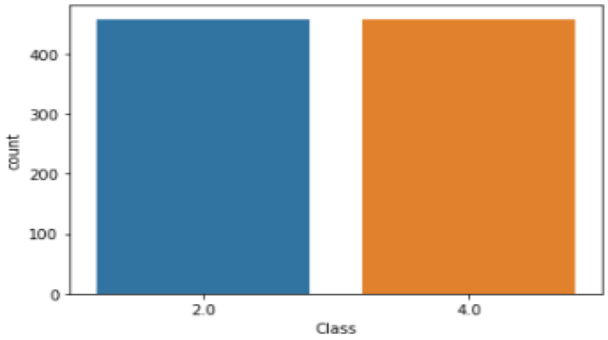


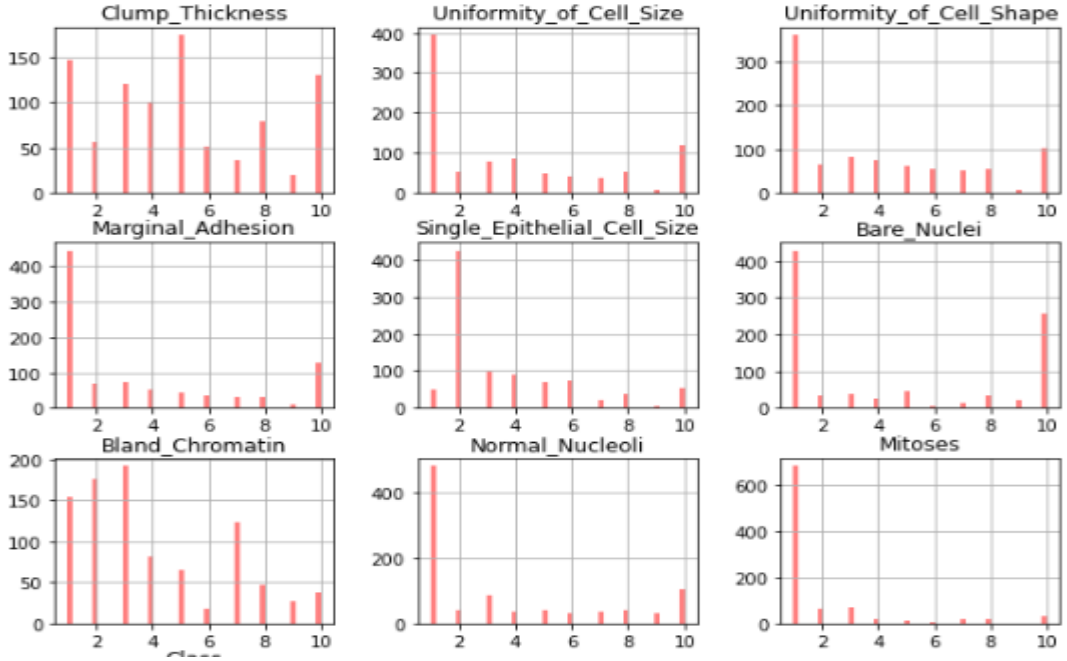

Figure 9 –Bar plot de oversampling du dataset

**Gaussian Distribution :** Comme on peut le voir sur la figure 6, on n'a pas de distributions normales des valeurs des attributs du Dataset.

Pour résoudre ce problème, on peut transformer les données en utilisant des transformations LOG, SQRT, COXBOX ou bien Normalizer. Dans notre cas, on a utilisé la transformation LOG.

Avant transformation :

```
Text(0.5, 1.0, 'Distribution des données de chaque colonne avant transformation')

<Figure size 432x288 with 0 Axes>
```

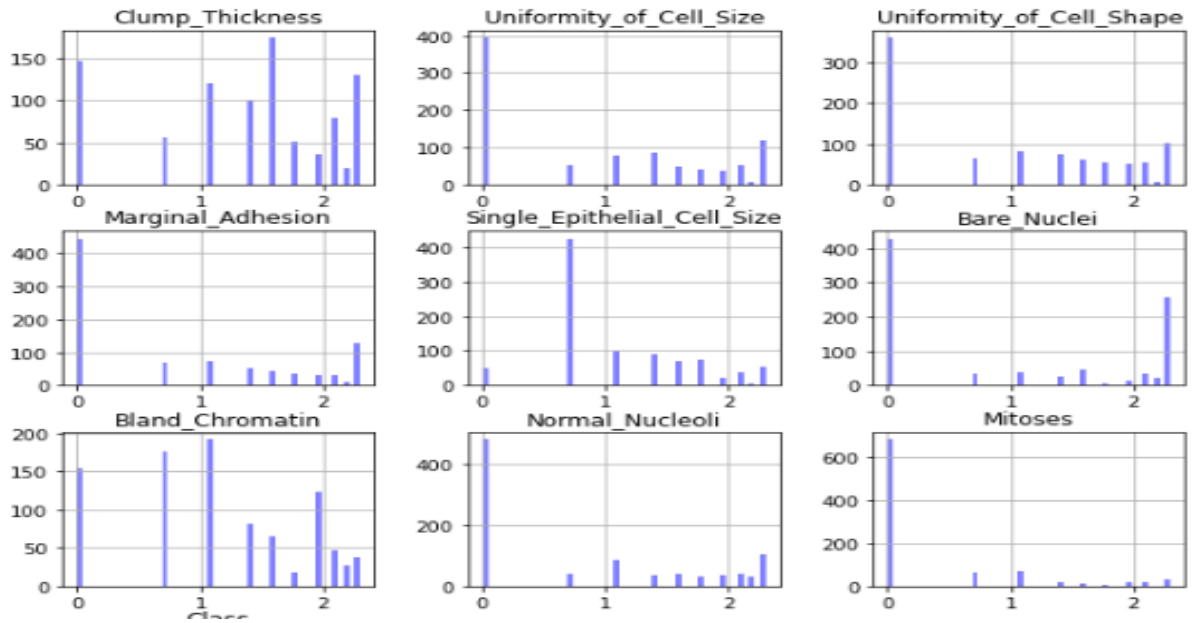



Le code utilisé, pour cette transformation est le suivant :

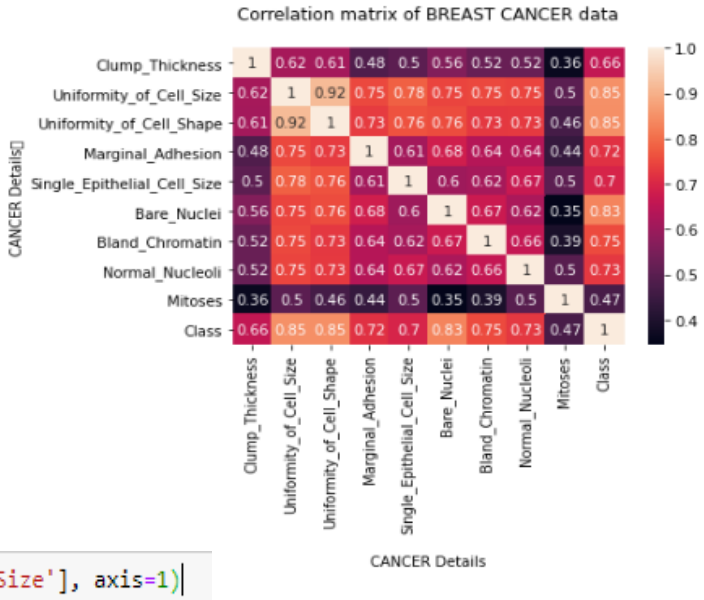
```
#LOG transformations
breast_DF_balanced["Clump_Thickness"] = breast_DF_balanced["Clump_Thickness"].map(lambda x: log(x))
breast_DF_balanced["Uniformity_of_Cell_Size"] = breast_DF_balanced["Uniformity_of_Cell_Size"].map(lambda x: log(x))
breast_DF_balanced["Uniformity_of_Cell_Shape"] = breast_DF_balanced["Uniformity_of_Cell_Shape"].map(lambda x: log(x))
breast_DF_balanced["Marginal_Adhesion"] = breast_DF_balanced["Marginal_Adhesion"].map(lambda x: log(x))
breast_DF_balanced["Single_Epithelial_Cell_Size"] = breast_DF_balanced["Single_Epithelial_Cell_Size"].map(lambda x: log(x))
breast_DF_balanced["Bare_Nuclei"] = breast_DF_balanced["Bare_Nuclei"].map(lambda x: log(x))
breast_DF_balanced["Bland_Chromatin"] = breast_DF_balanced["Bland_Chromatin"].map(lambda x: log(x))
breast_DF_balanced["Normal_Nucleoli"] = breast_DF_balanced["Normal_Nucleoli"].map(lambda x: log(x))
breast_DF_balanced["Mitoses"] = breast_DF_balanced["Mitoses"].map(lambda x: log(x))

print("Le Skew de chaque colonne apres la transformation:")
breast_DF_balanced.skew()
```

Apres transformation :

```
ext(0.5, 1.0, 'Distribution des données de chaque colonne apres transformatior
<Figure size 432x288 with 0 Axes>
```




On voit que certains attributs sont normalisés et que d'autres restent un peu skewed cela revient aux données déjà qui sont à l'échelle.




## 2-2-3-4 Data Réduction :

A partir de cette matrice de corrélation, on remarque que les deux colonnes Uniformity_of_Cell_Size et Uniformity_of_Cell_Shape sont fortement corrélés 0.92, et qu'on peut retirer une des deux colonnes. Mais, puisqu'il y'en a pas trop de données et de colonnes on peut aussi la garder.

Mon choix est de supprimer une colonne :

```
breast_DF_balanced = breast_DF_balanced.drop(['Uniformity_of_Cell_Size'], axis=1)
```

On va retirer la colonne Uniformity_of_Cell_Size, car après avoir supprimer cette colonne, le modèle répond mieux avec une meilleur V-Measure de 0.8138934242085933(avec Kmeans) à 0.8261869877795835(avec Kmeans). Pour ces raisons de performance du modèle on va la supprimer.

## 2-2-3-5 Data Discretization :

On n'a pas besoin de faire une discrétisation, car une Discrétisation c'est pour transformer des variables continues en variable catégoriques.

# 2-3 Measure the quality of the clusters methods:

Il y'a deux types de cas qu'on peut ce retrouvé à travailler avec afin de choisir les bonnes métriques pour vérifier la qualité des clusters.

Pour cela, on a :
Les datasets qui viennent avec leurs étiquettes (initially predicted labels).
Les datasets qui viennent sans leurs étiquettes (no initially predicted labels), qu'on prédit par la suite avec algorithmes de clustering

Dans notre cas sur le dataset BREAST CANCER, on se retrouve avec un dataset contenant déjà des labels initiaux, ce qui implique par la suite qu'on doit utiliser des métriques de la qualité qui prennent en compte cette avantage d'avoir des labels initiaux, et de pouvoir les comparer avec les prédictions faites par la suite. Parmi ces métriques on a :

**Rand Index** : Compte tenu de la connaissance des affectations de classe de vérité terrain labels_true et de nos affectations d'algorithme de clustering des mêmes échantillons labels_pred, l'indice de Rand (ajusté ou non) est une fonction qui mesure la similitude des deux affectations, en ignorant les permutations.

**Mutual information based scores :** Compte tenu de la connaissance des affectations de classe de vérité terrain labels_true et de nos affectations d'algorithme de clustering des mêmes échantillons labels_pred, l'information mutuelle est une fonction qui mesure l'accord des deux affectations, en ignorant les permutations. Deux versions normalisées différentes de cette mesure sont disponibles, les informations mutuelles normalisées (NMI) et les informations mutuelles ajustées (AMI). Le NMI est souvent utilisé dans la littérature, tandis que l'AMI a été proposé plus récemment et est normalisé contre le hasard.

**Homogeinity, completeness and V-Measure** : En connaissant les affectations de classe de la vérité du sol des échantillons, il est possible de définir une métrique intuitive en utilisant l'analyse de l'entropie conditionnelle.

En particulier, Rosenberg et Hirschberg (2007) définissent les deux objectifs souhaitables suivants pour toute affectation de cluster :

Homogénéité : chaque cluster ne contient que les membres d'une même classe.

Complétude : tous les membres d'une classe donnée sont affectés au même cluster.

Nous pouvons transformer ces concepts en scores homogeneity_score et completeness_score. Les deux sont limités en-dessous par 0,0 et au-dessus par 1,0 (plus c'est mieux).

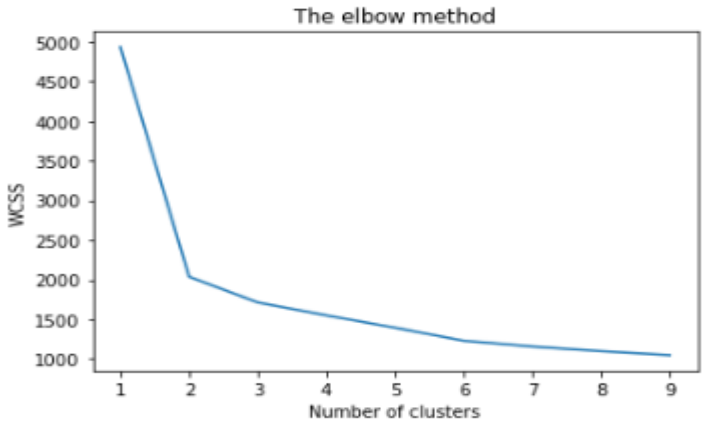
## 2-4 Les deux algorithmes de Clustering :

Après avoir préparer les données dans la partie de preprocessing de données, on va maintenant implémenter deux algorithmes de clustering pour faire de forecasting

### 2-4-1 KMeans:

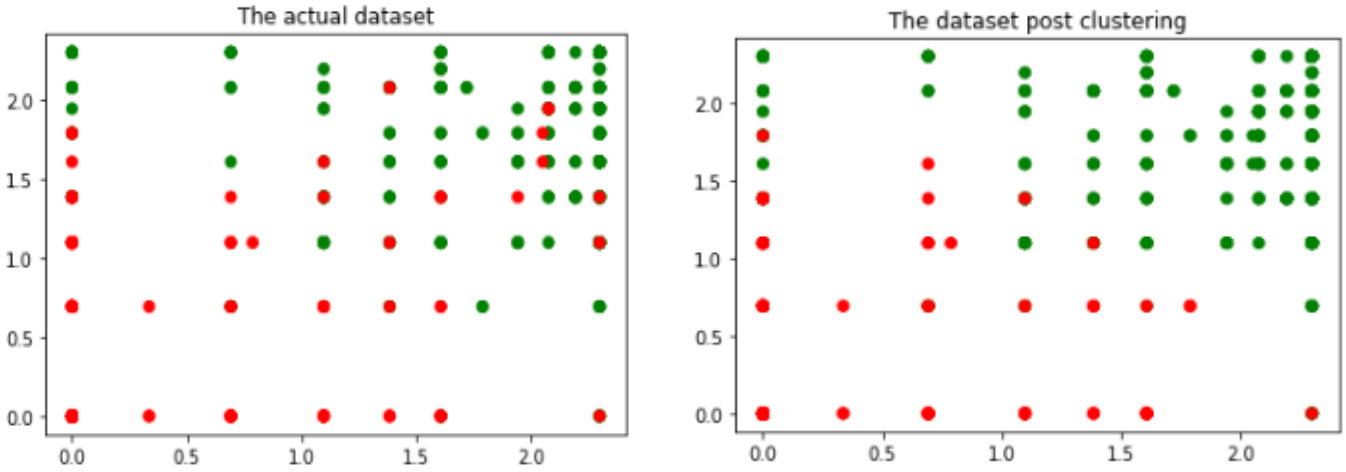
Avant d'utiliser KMeans on doit trouver le nombre optimum de clusters pour le classification KMeans avec la méthode du coude qui nous permet de choisir la quantité optimale de clusters pour la classification. Bien que nous sachions déjà que la réponse est 2, il est toujours intéressant de le voir.

**Remarque :** Pour le code de chaque figure suivante, je vous invite à aller consulter le NoteBook Breast_Cancer pour plus d'informations.




Comme on a pu voir, la méthode de elbow (coude) prouve qu'on doit prendre K=2, car sur le graphe on voit il devient comme un coude quand x=2.

A partir de la on peut lancer l'algorithme des KMeans pour K=2.



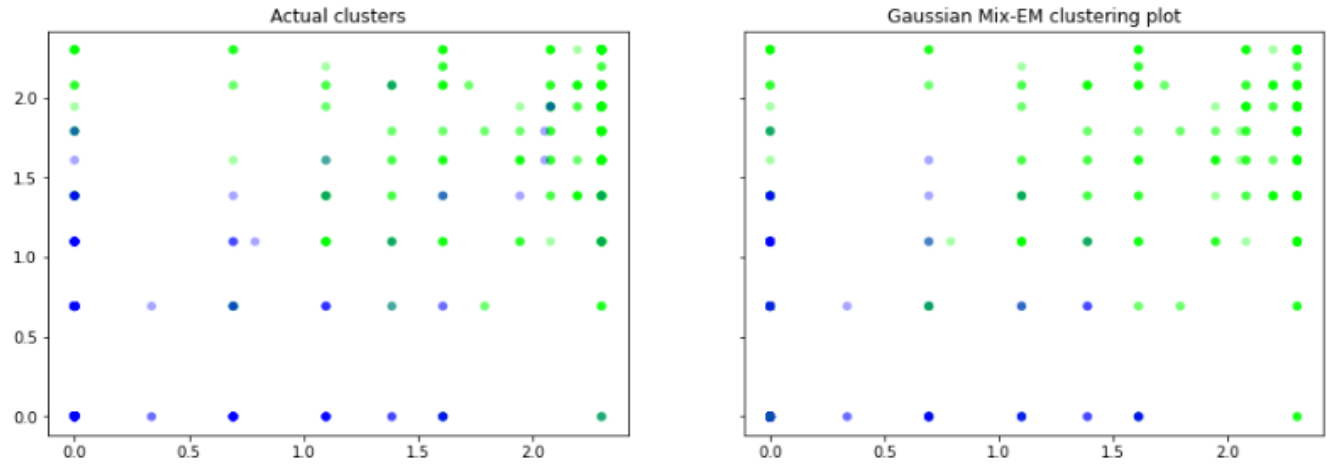

Sur ces deux graphes, on peut voir les différentes dispersions de dataset, en premier on a celui d'à gauche qui represente le dataset initial tenant compte de ces labels initiaux, puis on a droit la nouvelle dispersion des données après le clustering du l'algorithme de KMeans, ou on arrive à voir que l'algorithme à presque identiquement eu les mêmes labels que le premier graphe à gauche.

```
Homogeneity du clustering 0.8261358458787106
Completeness du clustering 0.8262381360127226
Leur moyenne harmonique appelée V-mesure est calculée par v_measure_score :
0.8261869877795835
```

## 2-4-2 Gaussian Mixture with Expectation Maximization (EM) Clustering

L'algorithme d'espérance-maximisation, ou algorithme EM en abrégé, est une approche pour l'estimation du maximum de vraisemblance en présence de variables latentes. L'algorithme EM a été expliqué et donné son nom dans un article classique de 1977 par A. Dempster et D. Rubin dans le Journal of the Royal Statistical Society.

```
Text(0.5, 1.0, 'Gaussian Mix-EM clustering plot')
```




Sur ces deux graphes, on peut voir les différentes dispersions de dataset, en premier on a celui d'à gauche qui représente le dataset initial tenant compte de ces labels initiaux, puis on a droit la nouvelle dispersion des données âpres le clustering du l'algorithme de GM(EM), ou on arrive à voir que l'algorithme à presque identiquement eu les mêmes labels que le premier graphe à gauche.

```
Homogeneity du clustering 0.757030615922175
Completeness du clustering 0.7604231326039544
Leur moyenne harmonique appelée V-mesure est calculée par v_measure_score :
0.758723081999397
```

# 2-5 KMeans VS Gaussian Mixture with Expectation Maximization (EM) Clustering

On va comparer dans cette section la qualité des deux algorithmes en utilisant les trois différentes métriques (même ci une est largement suffisante), mais c'est pour voir c'est y'a vraiment des grands changements avec ces métriques.

Aussi, on va utiliser silhouette qui est une métrique qu'on a pas décrit là-dessus, mais c'est une métrique pour le cas où on ne connaît pas les labels initiaux de dataset. (Le code veuillez regarder le notebook)

| | ARI | AMI | Homogenity | Completeness | V-measure | Silhouette |
|---|---|---|---|---|---|---|
| **K-means** | 0.897831 | 0.826050 | 0.826136 | 0.826238 | 0.826187 | 0.506151 |
| **Gaussian Mixture with EM** | 0.824813 | 0.758532 | 0.757031 | 0.760423 | 0.758723 | 0.473224 |

A partir des résultats obtenus, on peut conclure que le meilleure model pour cette expérience sur le Breast Cancer Dataset est KMeans. Ou on peut voir, il a légèrement le dessus sur GM sur toutes les mesures de qualité vu et effectué sur ces deux modèles.

Cela conclut, qu'on doit prendre KMeans que GM(EM), car il donne des meilleurs résultats que çà soit avec les labels initiaux de dataset ou sans, il est toujours devant plus précis que l'autre algorithme de clustering.