PTML Project

Guillaume CARRIERE, Romain GREGOIRE, Alex Poiron, Tom THIL

July 2022

1 Apprentissage supervisé

1.1 Description du jeu de données

Pour cette partie du projet, nous avons décidé de prendre un jeu de données basé sur la musique. Plus partiulièrement, sur les différents éléments que composent un signal sonore (tempo, beats, zero crossing rate, spectral centroid, short time Fourier transform ...).

Ces éléments sont souvent à but très utile dans le domaine de traitement de signal. En plus de ces éléments numériques, on retrouve pour chaque musique, son genre musical associé (blues, jazz, classique, ...). Notre objectif avec ce jeu de données est de savoir s'il est possible de prédire le genre musical en fonction de ces divers éléments que composent un signal sonore.

Avant de passer à l'établissement de plusieurs modèles pour arriver à notre objectif, nous allons dans un premier temps analyser celui-ci et observer ce que l'on peut en déduire. Vous trouverez toutes les étapes d'analyses ainsi que les visualisations associées au sein du Notebook **analysis_music_features**

1.2 Analyses Statistiques

1.2.1 Valeurs fondamentales et histogrammes

Dans un premier temps, nous avons voulu pour chaque valeur numérique calculer les valeurs suivantes afin d'avoir un résumé statistique :

- Moyenne
- Ecart-type
- Médiane
- Valeur Minimale
- Fractiles
- Valeur Maximale

	tempo	beats	chroma_stft	rmse	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	119.601702	57.138000	0.378656	0.130929	2201.834226	2242.559613	4571.702159	0.103637	-144.479173
std	28.297367	14.225728	0.081689	0.065685	715.961347	526.337663	1574.770035	0.041834	100.235661
min	54.978391	18.000000	0.171782	0.005276	569.930721	897.994319	749.062137	0.021701	-552.064063
25%	99.384014	47.000000	0.319641	0.086625	1627.793931	1907.136505	3380.956639	0.070281	-200.695133
50%	117.453835	56.000000	0.383075	0.122448	2209.468780	2221.408983	4658.671830	0.099539	-120.206072
75%	135.999178	65.250000	0.435974	0.175793	2691.969702	2578.474352	5534.197785	0.132007	-73.895019
max	234.907670	117.000000	0.663573	0.398012	4434.439444	3509.578677	8676.405868	0.274829	42.034587

FIGURE 1 – Valeurs statistiques de nos features

Par la suite, nous avons voulu visualiser nos différentes valeurs, notamment à l'aide d'histogrammes. Par exemple, si on s'intéresse aux valeurs que preuve prendre le **Roll off** (Le roll-off est la pente d'une

fonction de transfert avec la fréquence). On obtient donc cet histogramme avec ces statistiques :

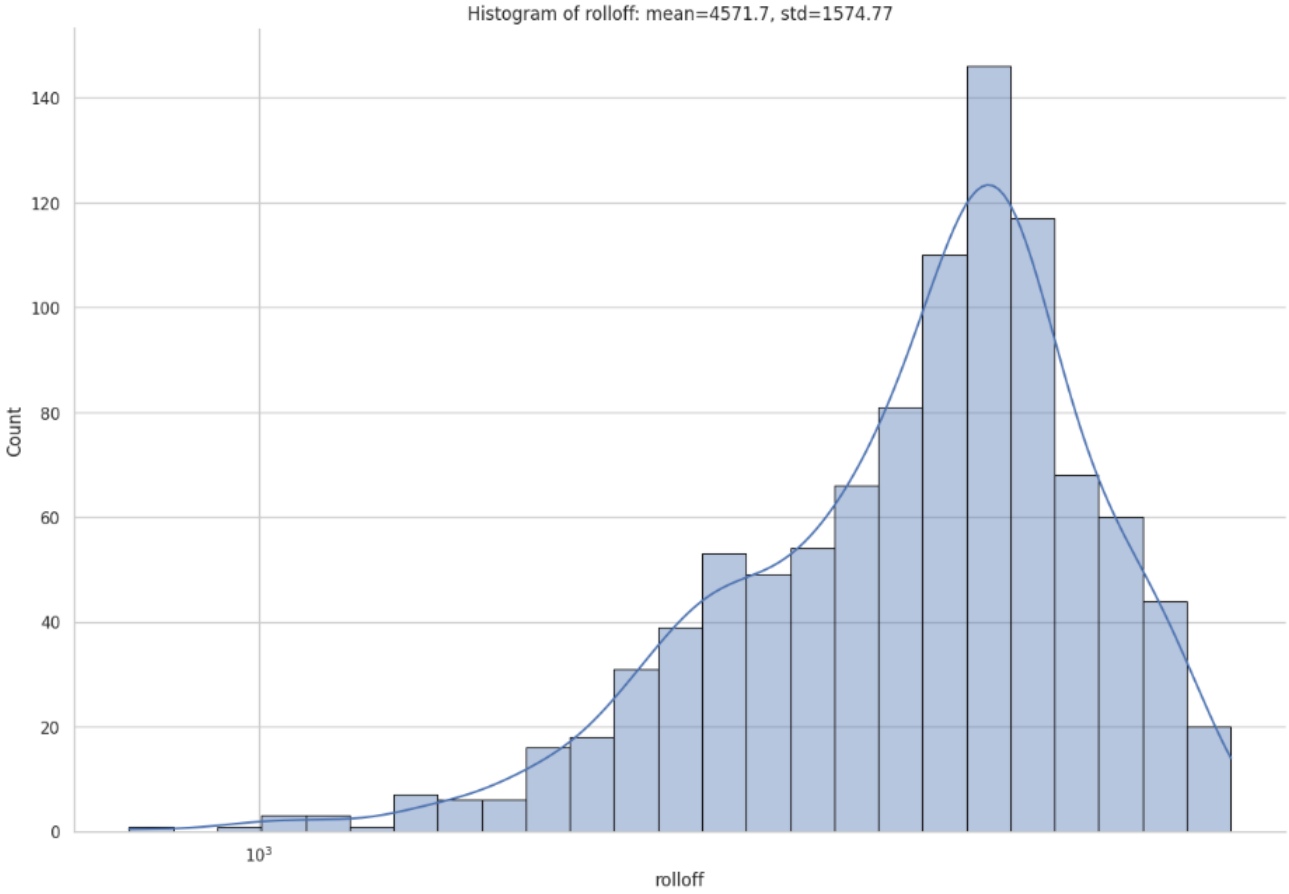

FIGURE 2 – Histogramme du Roll off

1.2.2 Eléments du signal et genre musical

Dans un second temps, on s'intéresse à vouloir représenter une variable quantitative avec une seconde pour voir leur comparaison et potentiellement voir s'il y a un premier lien que l'on peut observer visuellement. Ici, ce qui nous intéresse est le genre musical que l'on essaye de prédire en fonction de différents éléments composant un signal sonore.

Comme exemple, on peut s'intéresser de savoir comment le tempo est représenté en fonction du genre musical. Pour cet exemple, voici une légende pour comprendre un peu mieux :

- Blues = 0
- Classical = 1
- Disco = 2
- Jazz = 3
- Rock = 4
- Pop = 5
- Metal = 6
- Reggae = 7
- Country = 8
- Hiphop = 9

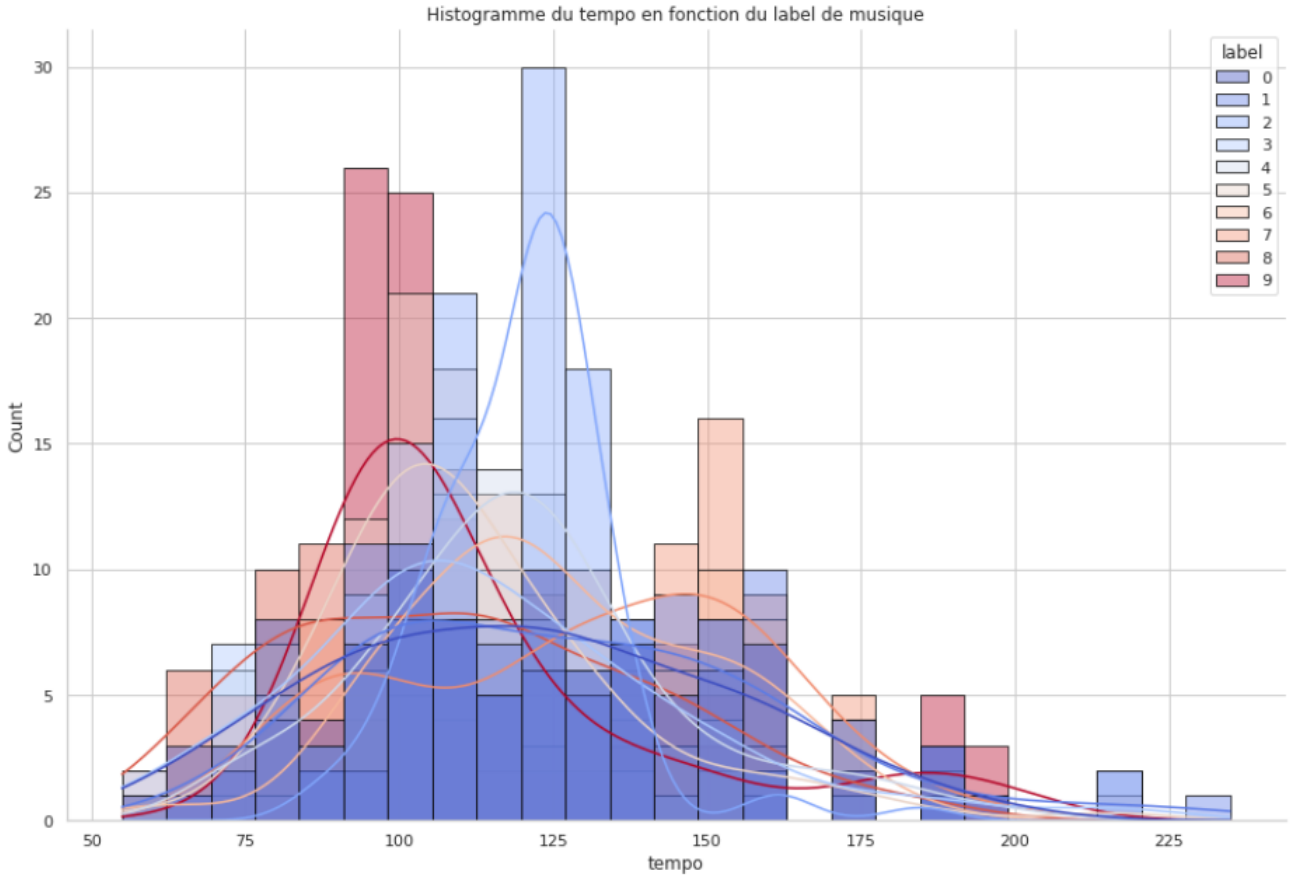

FIGURE 3 – Histogramme du tempo en fonction du label de musique

On peut observer qu'un tempo aux alentours de 100 correspond majoritairement à une musique HipHop, un tempo plutôt vers 125-130 s'associe majoritairement avec le Jazz.

1.2.3 Corrélation

Enfin, nous allons nous intéresser à la corrélation entre nos divers éléments composant nos signaux sonores. Pour ce faire, il est assez intuitif d'établir une matrice de corrélation. Pour ce jeu de données, nous obtenons donc cette matrice de corrélation :

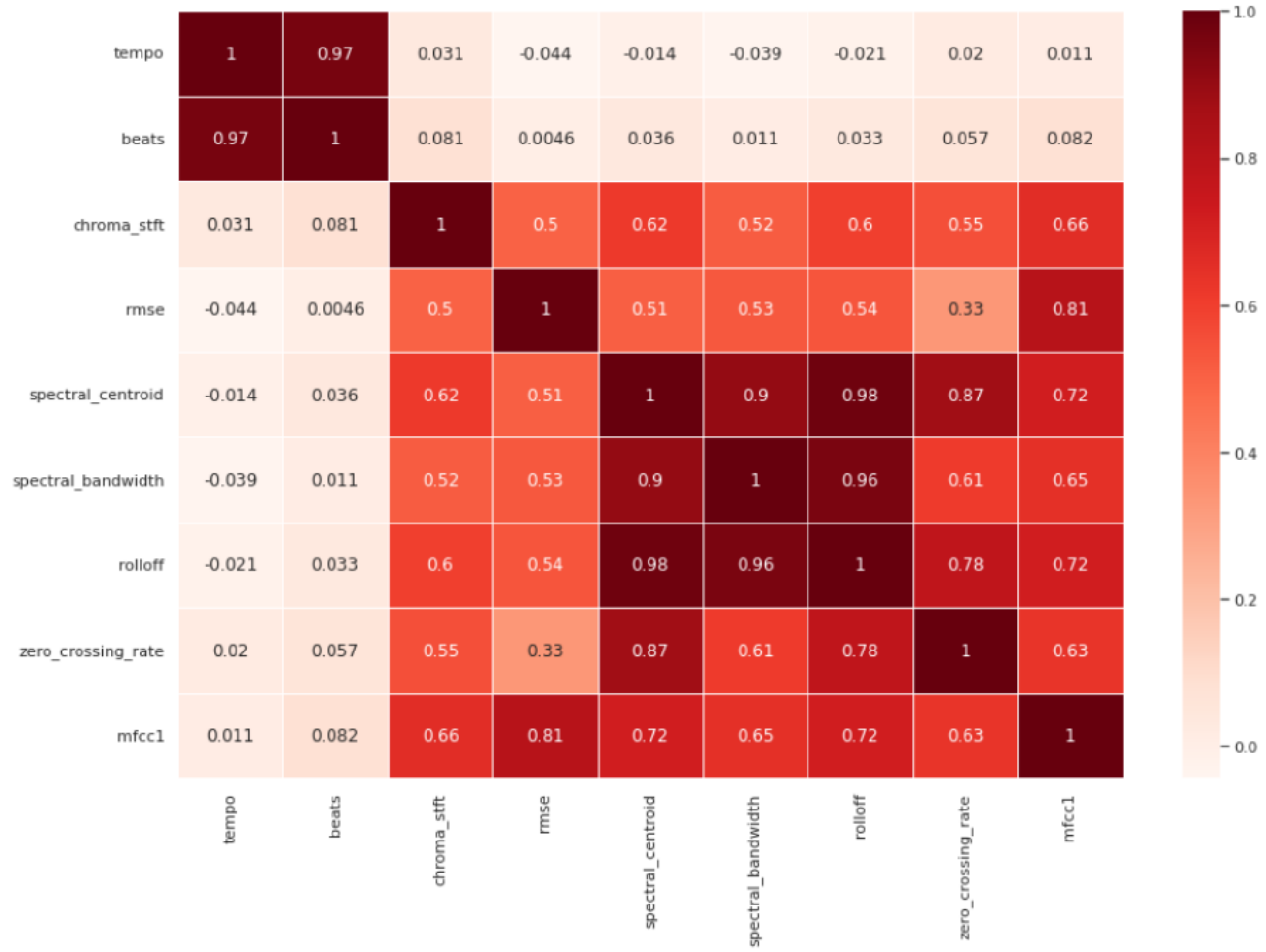

FIGURE 4 – Matrice de corrélation

Comme on peut le remarquer, on distingue deux groupes. Le premier étant composé du **Tempo** ainsi que du **Beats** qui sont fortement corrélés entre eux et absolument pas du reste. Le deuxième groupe est formé de toutes les autres valeurs qui sont plus proche du signal en lui même. Particulièrement, on observe que l'élément **Roll off** est très corrélé avec le **Spectral centroïd**, le **Spectral bandwidth** et également le **Zero crossing rate**.

1.3 Modèles

Pour la partie modèle, nous avons utilisé la librairie **sklearn** pour Python qui nous permet d'avoir accès à de très nombreuses méthodes et de modèles différents. Nous avons retenu pour cette partie 3 modèles différents que nous allons détailler dans cette partie.

Avant de commencer à les présenter, voici à quoi ressemble notre jeu de données et comment celui-ci est représenter :

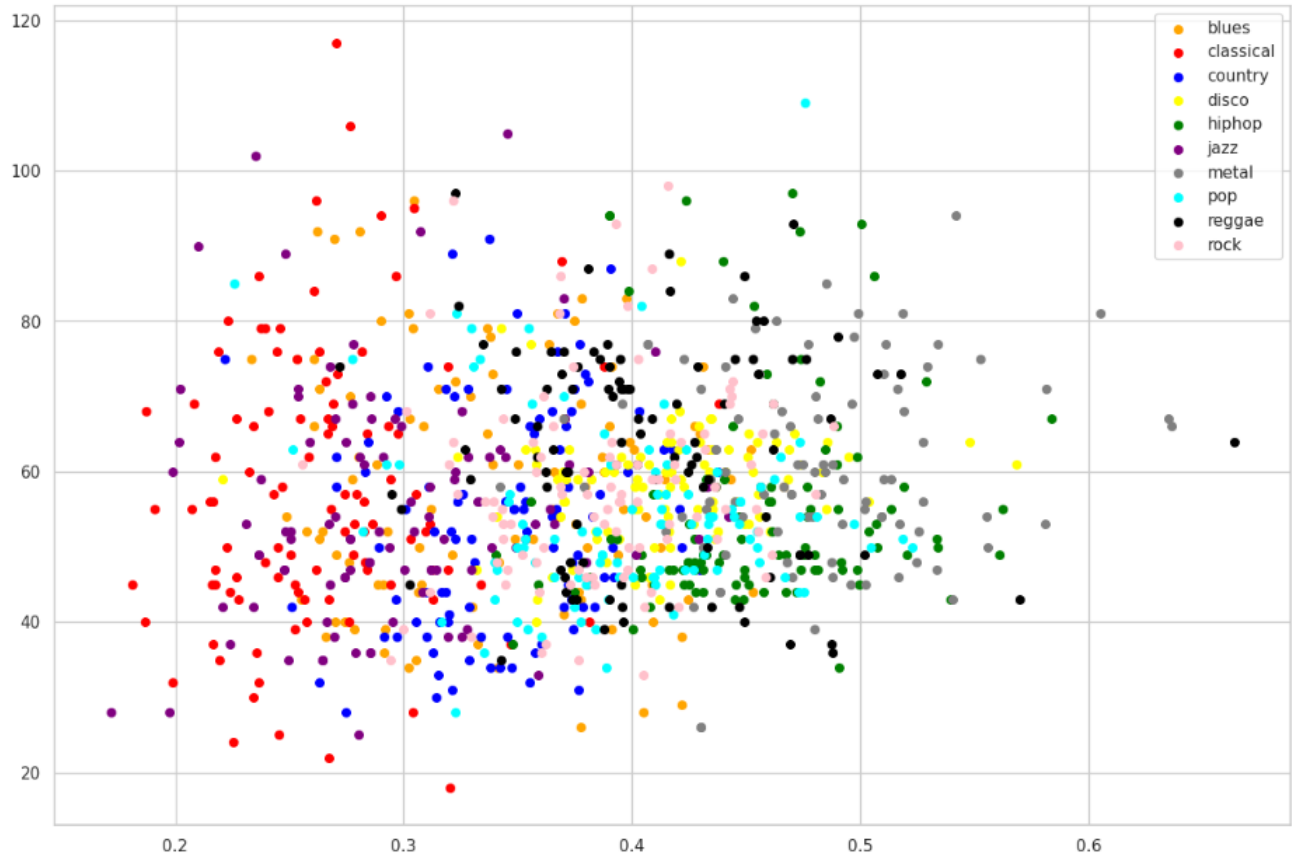

FIGURE 5 – Représentation des valeurs numériques de notre jeu de données

1.3.1 Stochastic Gradient Descent Classifier

Dans un premier temps, nous avons décidé d'utiliser le modèle **SGD Classifier**, qui permet d'utiliser différents algorithmes soit de classification ou même de régression. En effet, c'est par le choix de la **loss function** que l'algorithme est déterminé. Dans notre exemple, nous avons décidé d'utiliser comme loss function : l = **Hinge**, ce qui nous permet d'avoir accès à un algorithme de **SVM linéaire**.

On remarque en lisant la documentation de cette classe, qu'il existe encore beaucoup de **loss functions** différentes pour utiliser d'autres algorithmes comme une régression logisitque. En revanche, celle-ci n'est pas appropriée dans notre cas car nous avons 10 classes différentes et non 2.

Pour chaque modèle, nous avons évalué son accuracity, son F1 score ainsi que sa cross validation.

Le principe de SVM est simple, il consiste à ramener un problème de classification ou de discrimination à un hyperplan dans lequel les données sont séparées en plusieurs classes. Ce modèle a été rapidement adopté en raison de sa capacité à travailler avec des données de grandes dimensions, ses garanties théoriques et les bons résultats obtenus en pratique.

Passons à l'évaluation de l'algorithme. Pour celui-ci, nous obtenons comme résultats :

- Accuracy = 26,96
- F1 Score = 18,36
- Cross validation : [0.18181818, 0.27272727, 0.18181818, 0.18181818, 0.3030303, 0.3030303, 0.42424242, 0.18181818, 0.15151515, 0.24242424]

On le remarque assez facilement, mais SVM ne semble pas le bon algorithme par rapport à notre jeu de données. Une des raisons peut être le nombre de classes que nous avons qui peut être tout simplement trop élevé et qui donc n'est pas adapté, en tout cas, dans un cas d'un simple Classifier de descente de gradient stochastique.

Il faut donc voir si en améliorant cette descente de gradient, nous pourrions obtenir des résultats plus probants. C'est ce que nous verrons dans la partie suivante.

1.3.2 Gradient boosting Classifier

Dans un second temps nous avons choisi d'utiliser le gradient boosting classifier. Le principe de ce modèle est de combiner plusieurs modèles relativement faibles entre eux pour créer un modèle prédictif plus performant.L'algorithme se base sur l'hypothèse boosting reposant sur le concept de **l'apprentissage PAC** (de anglais probably approximately corect). Les gradient boosting classifier sont devenus particulièrement populaire en raison de leur efficacité à classifier des datasets complexes.

Pour ce qui est de l'évaluation de ce modèle nous obtenons les résultats suivants :

- Accuracy = 53,6
- F1 Score = 53,1
- Cross validation : [0.42424242, 0.42424242, 0.39393939, 0.48484848, 0.42424242, 0.42424242, 0.48484848, 0.42424242, 0.39393939, 0.33333333]

Comme vous pouvez le remarquer les résultats sont bien meilleurs que lorsque nous avons utilisé une SVM. On peut potentiellement expliquer cette variation en raison du fait que le modèle de la SVM se prête moins à la classification de notre dataset que le gradient boosting classifier.

1.3.3 MultiLayer Perceptron

Le multilayer perceptron, ou perceptron multicouche, est un type de réseau neuronal artificiel organisé en plusieurs couches au sein desquelles une information circule entre la couche d'entrée vers la couche de sortie **uniquement**. Chaque couche cachée est constituée d'un nombre variable de neurones, les neurones de la dernière couche étant les sorties du système global.

Pour le MultiLayer Perceptron nous obtenons les résultats suiants :

- Accuracy = 60.9
- F1 Score = 59.8
- Cross validation : [0.45454545, 0.45454545, 0.48484848, 0.48484848, 0.54545455, 0.42424242, 0.51515152, 0.33333333, 0.48484848, 0.36363636]

Il s'agit donc de notre modèle le plus performant d'environ 10%. Néanmoins les résultats obtenus auraient probablement pu être encore meilleurs si nous avions poussé l'utilisation d'un modèle de deep learning.

1.3.4 Résultats

Enfin, dans cette partie nous allons comparé nos différents résultats avec tous nos algorithmes en apprentissage supervisé.

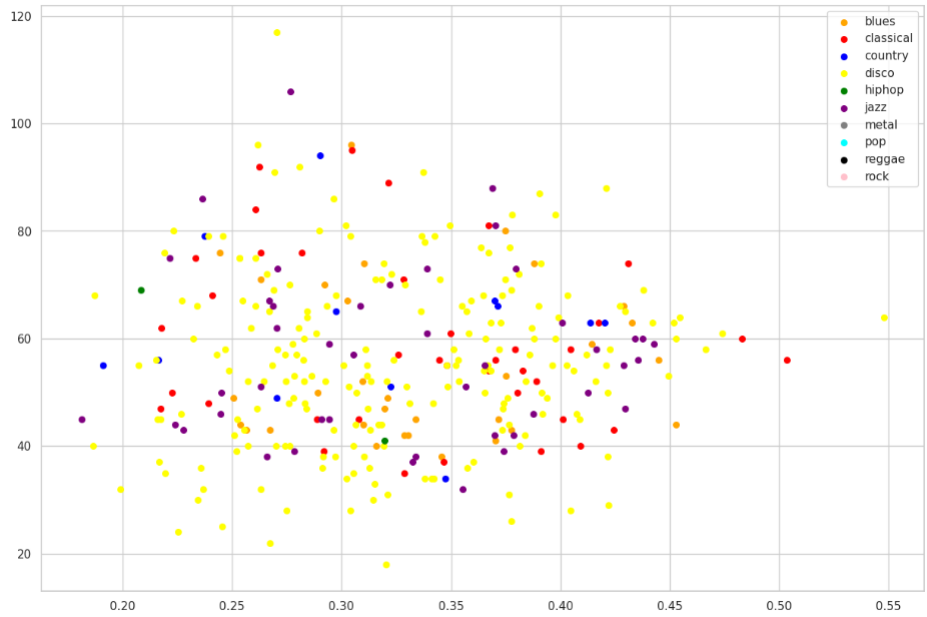

FIGURE 6 – Résultats en utilisant un SGD Classifier

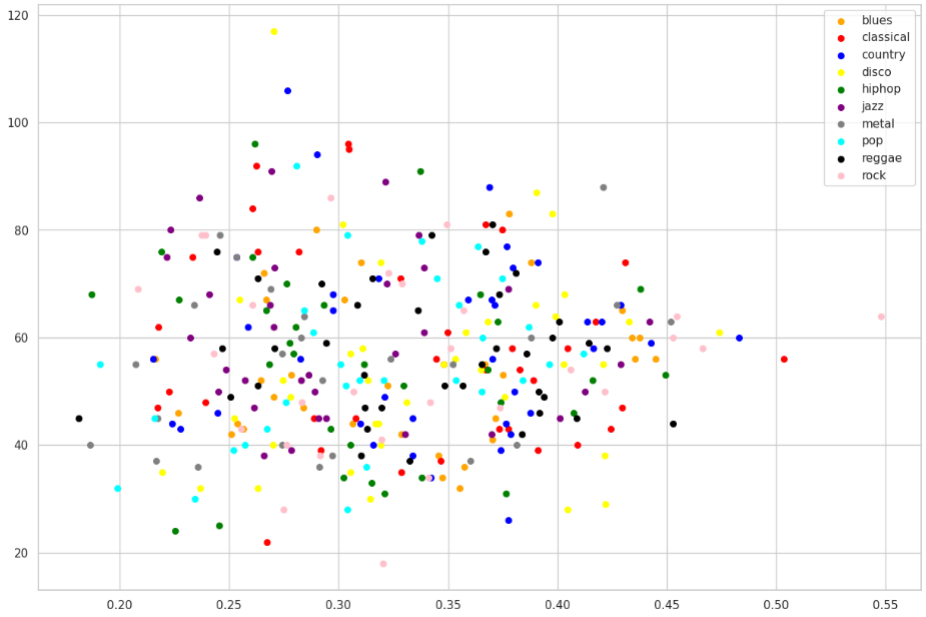

FIGURE 7 – Résultats en utilisant un Gradient Boosting Classifier

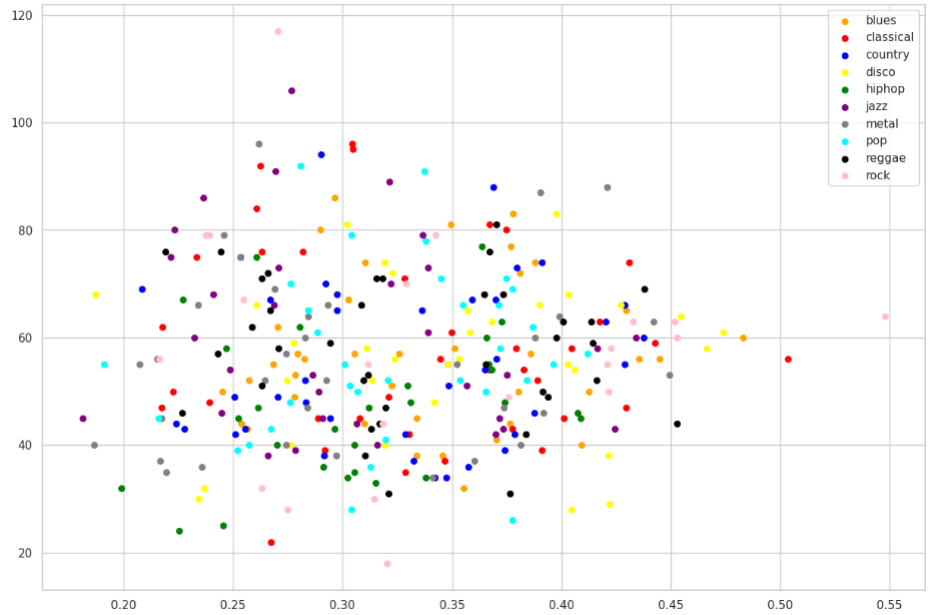

FIGURE 8 – Résultats en utilisant un MultiLayer Perceptron Classifier

2 Apprentissage non-supervisé

2.1 Description du jeu de données

Pour cette partie du projet, nous avons décidé de prendre un jeu de données répertoriant les charges d'assurance d'un échantillon de personnes, en fonction de leur âge et de leur état de santé.

Notre objectif pour cette partie sera de reconnaître des patterns dans nos données, afin de voir si nous sommes capables d'identifier des profils type de personnes. Nous allons donc tester différentes méthodes de clustering.

A l'instar de la partie supervisée, nous allons cependant d'abord commencer par une analyse des données.

2.2 Analyses Statistiques

Tout d'abord nous avons réalisé une analyse descriptive de chaque données avec :

- Moyenne
- Écart-type
- Médiane
- Valeur Minimale
- Fractiles
- Valeur Maximale

	age	sex	bmi	children	smoker	region	charges
count	1338.0	1338.0	1338.0	1338.0	1338.0	1338.0	1338.0
mean	39.20702541106129	0.4947683109118087	30.66339686098655	1.0949177877429	0.20478325859491778	1.515695067264574	13270.422265141257
std	14.049960379216154	0.500159569284377	6.098186911679014	1.205492739781914	0.4036940375456173	1.1048849185826897	12110.011236694001
min	18.0	0.0	15.96	0.0	0.0	0.0	1121.8739
25%	27.0	0.0	26.29625	0.0	0.0	1.0	4740.28715
50%	39.0	0.0	30.4	1.0	0.0	2.0	9382.033
75%	51.0	1.0	34.69375	2.0	0.0	2.0	16639.912515
max	64.0	1.0	53.13	5.0	1.0	3.0	63770.42801

FIGURE 9 – Analyse descriptive

Ensuite, nous avons réalisé différents histogrammes et graphes permettant de mieux visualiser le jeu de données :

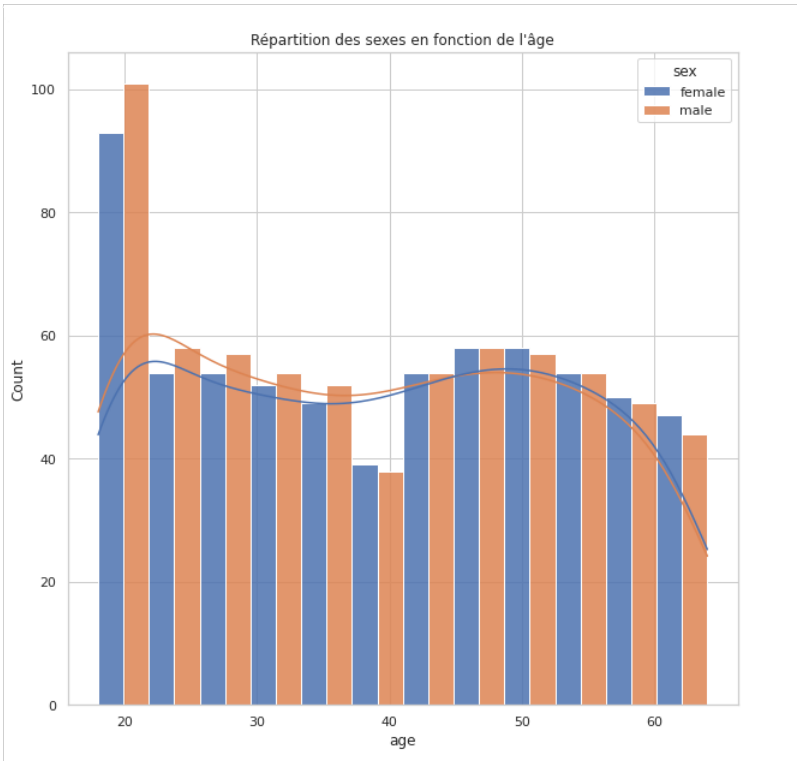

FIGURE 10 – Répartition des sexes en fonction de l'âge

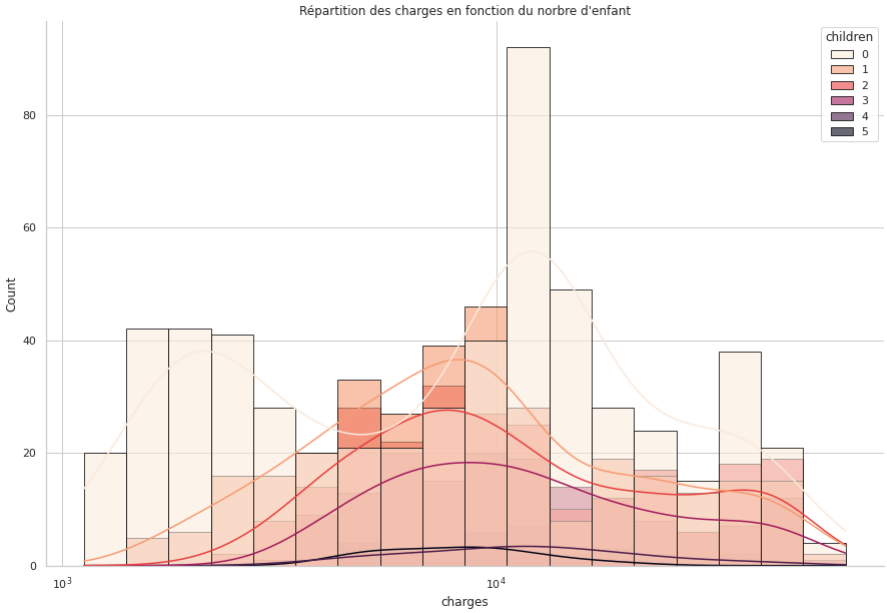

FIGURE 11 – Répartition des charges en fonction du nombre d'enfant

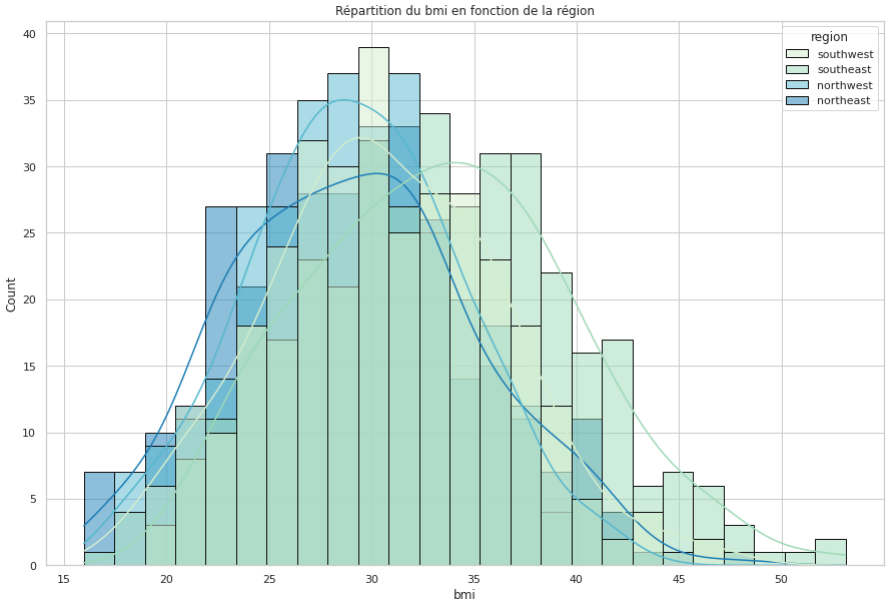

FIGURE 12 – Répartition du bmi en fonction de la région

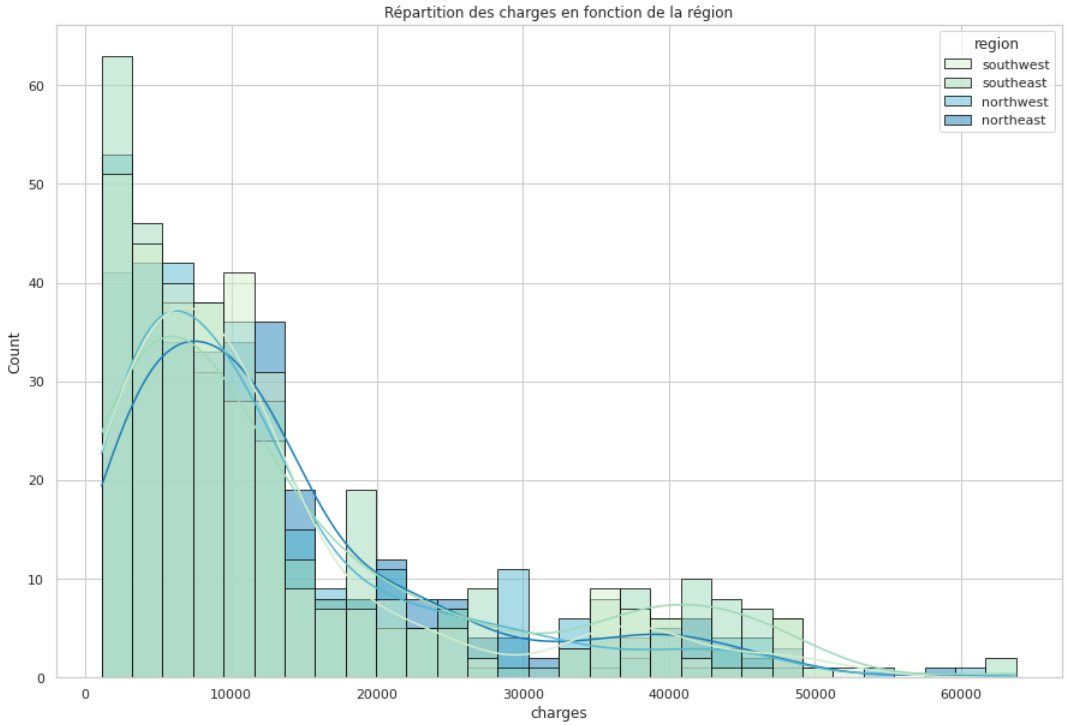

FIGURE 13 – Répartition des charges en fonction de la région

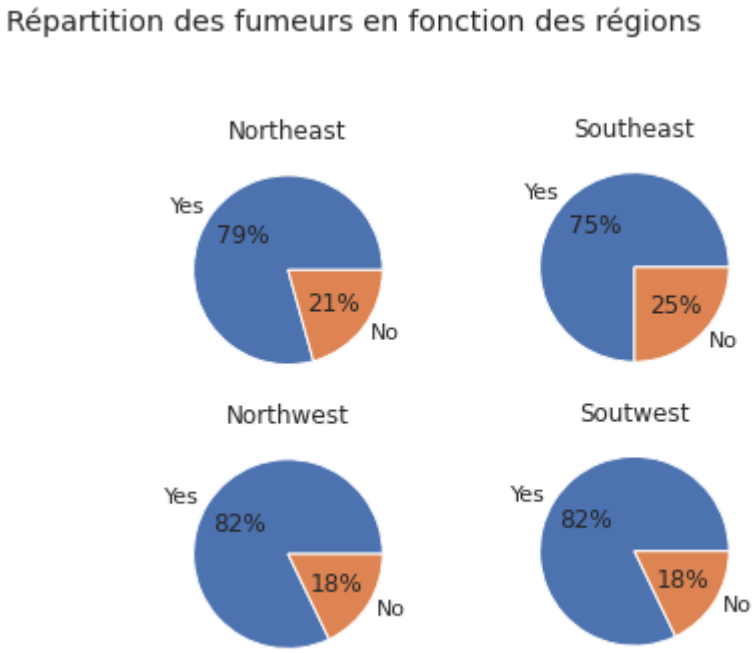

FIGURE 14 – Répartition des fumeurs en fonction des régions

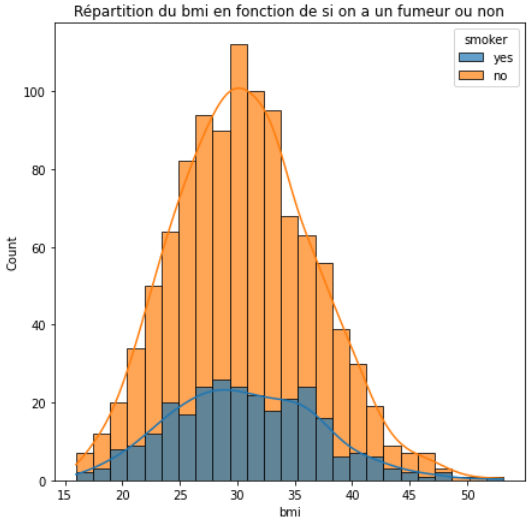

FIGURE 15 – Répartition du bmi en fonction de si on a un fumeur ou non

Ici, nous avons l'histogramme qui révèle l'information la plus importante du dataset, les charges en fonction de si on est fumeur ou non.

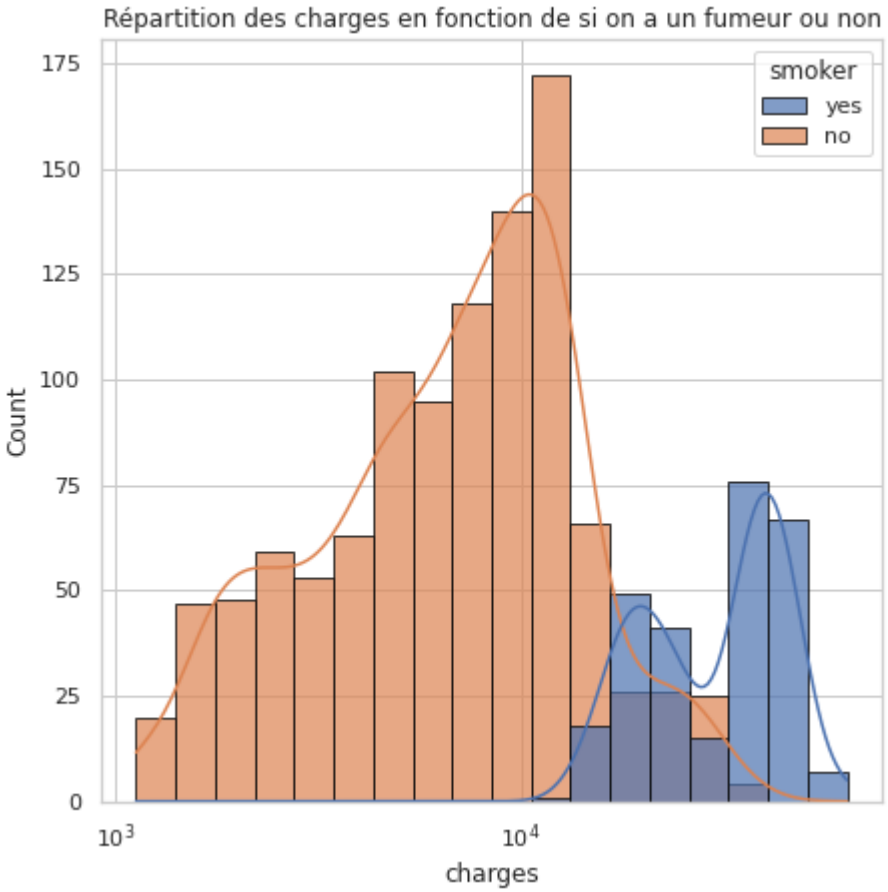

FIGURE 16 – Répartition des charges en fonction de si on a un fumeur ou non

Cela est confirmé par la matrice de corrélation du dataset :

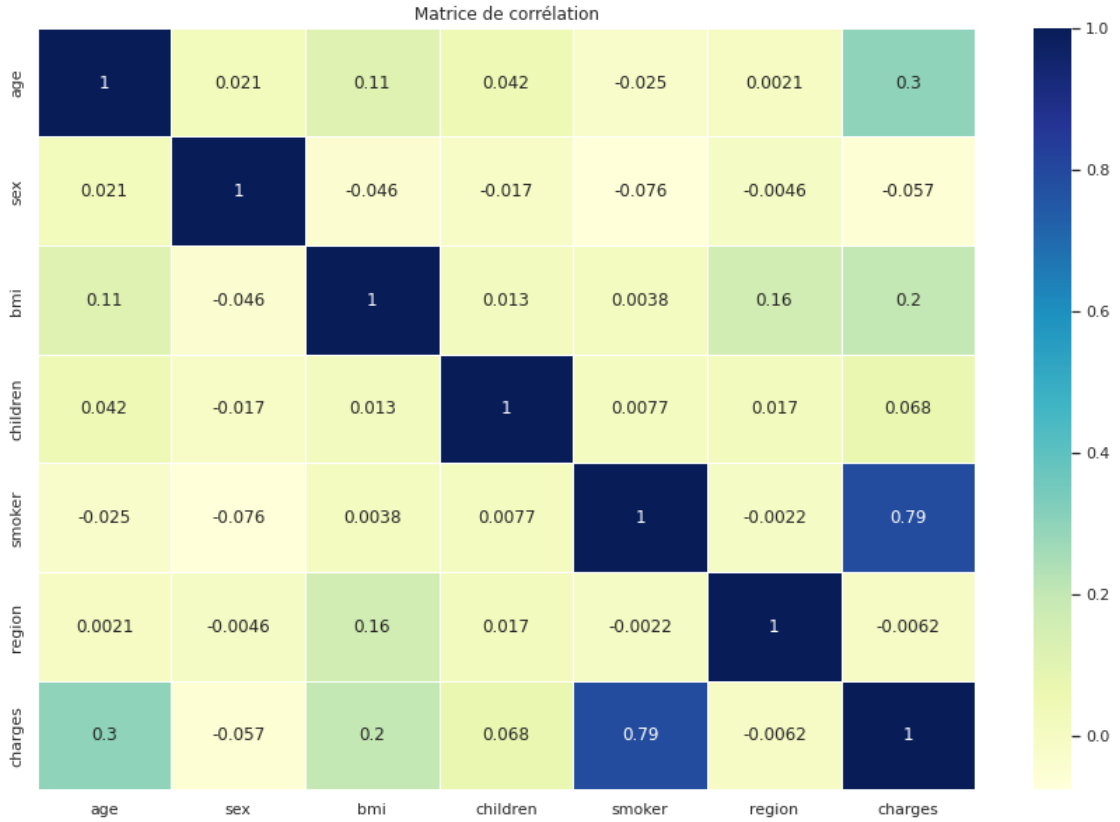

FIGURE 17 – Matrice de corrélation

2.3 Modèles

Au vu des résultats de l'analyse, nous avons observé 3 groupes principaux dans les points de données, que nous avons tenté de repérer automatiquement à l'aide de méthodes de clustering. Les groupes principaux que nous avons repéré à la main sont visible sur la fig.18.

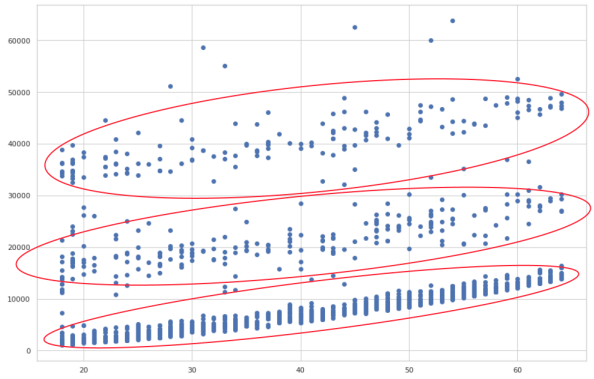

FIGURE 18 – Groupes repérés manuellement

2.3.1 K-means

Pour commencer, nous avons décidé de constituer une baseline basique en utilisant du K-means clustering sur nos données, réduites aux seules features "age" et "charges". Cela nous a permit d'obtenir les résultats disponibles en fig.19. Le premier problème apparent est que les données d'un des cluster que nous avons observé à la main semblent ne pas être détéctées en totalité. Cependant ces premiers résultats sont déjà prometteurs.

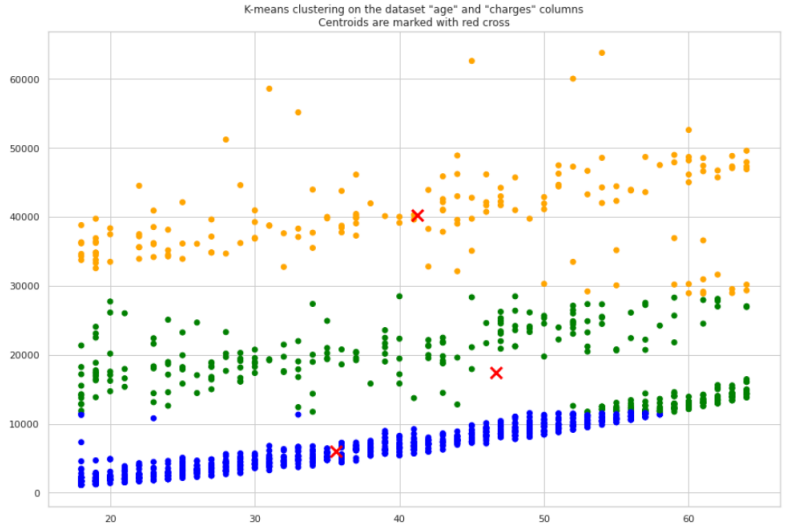

FIGURE 19 – K-means clustering

Nous avons ensuite tenté de rajouter de nouvelles variables : la "bmi" (body mass index), ainsi que les variables catégorielles correspondant au fait d'être un fumeur, le sexe ainsi que la région d'habitat de la personne. Pour ce faire, nous avons factorisé les variables catégorielles, puis réduit notre nouveau dataset à l'aide d'un PCA en conservant seulement les deux premiers Principal Components.

Avec ces nouvelles données et la même méthode de K-means, nous avons obtenu les résultats visibles en fig.20. Ces résultats sont assez similaires à ceux obtenus précédemment, et on peut visuellement constater que les deux PC on l'air de correspondre à peu près aux variables "age" et "charges. Cela est

FIGURE 20 – K-means clustering using PCA

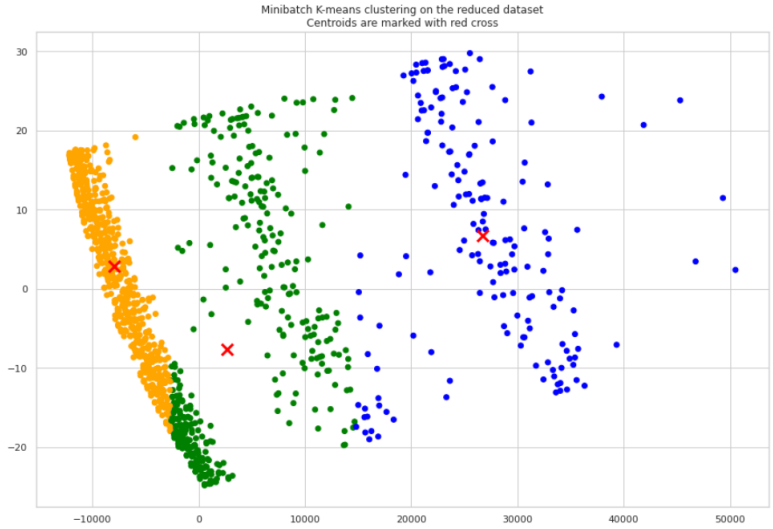

FIGURE 21 – Minibatch K-means clustering

probablement du a la colinéarité entre les nouvelles variables ajoutées par le PCA et les deux variables que nous utilisions déjà. Cependant, les résultats étant très légèrement différents nous avons tout de même fait le choix de conserver ces données pour les futures méthodes.

2.3.2 Minibatch K-means

Ensuite nous avons employé la méthode de Minibatch K-means. L'avantage du Minibatch K-means par rapport au K-means consiste en un gain en performance, contre une légère perte de précision. La performance du K-means n'étant pas problématique dans notre cas, la décision de tester le Minibatch K-means provient simplement de notre propre curiosité quant à la perte de précision de cette méthode par rapport au K-means. Les résultats disponibles en fig.21 correspondent à cette nouvelle méthode, et on peut effectivement constater visuellement que le problème précédemment mentionné pour le K-means a empiré.

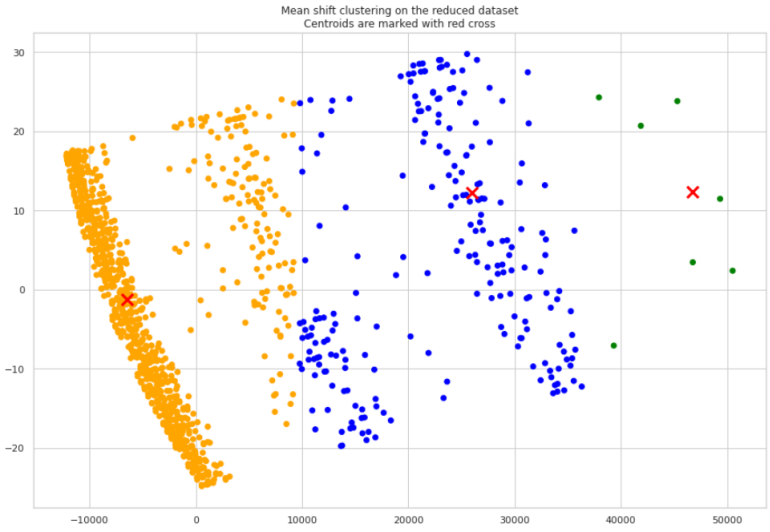

FIGURE 22 – Mean Shift clustering

2.3.3 Mean shift

La méthode suivante que nous avons employée est le Mean Shift clustering, une autre méthode de clustering assez classique, avec cette fois réellement pour objectif d'obtenir un clustering qui nous convenait mieux. Nous l'avons aussi appliqué à nos données réduites par PCA et ainsi obtenu les résultats en fig.22. Ces résultats ne correspondent pas à ce que nous attendions, les 3 groupes que nous avons observé manuellement sont interprétés par 2 cluster par la méthode de Mean Shift, qui a également constitué un 3e cluster avec les points de données aberrants. Dans notre cas cette méthode est donc trop sensibles aux données aberrantes, qui sont nombreuses dans notre dataset.

2.3.4 Spectral Clustering

Nous avons ensuite employé la méthode du spectral clustering. Notre raisonnement derrière l'utilisation de cette méthode est qu'elle est plus performante que le K-means lorsque les clusters sont "non-globulaires". Or, nous suspectons la méthode de K-means de ne pas réussir à identifier le cluster dense constaté visuellement car ce dernier n'a pas une forme "globulaire", d'où notre choix.

Pour que le spectral clustering fonctionne, il nous faut des données normalisées. Nous avons donc normalisé les données réduites par PCA et ainsi constitué un nouveau dataset avec lequel nous avons obtenu les résultats disponibles en fig.23 en utilisant le spectral clustering. Encore une fois, le clustering ne correspond pas à ce que nous recherchions et les clusters identifiés visuellement ne sont pas repérés, ce qui est peut-être à nouveau du à la quantité de valeurs aberrantes entre les clusters.

2.3.5 DBSCAN

Enfin la dernière méthode que nous avons employé est la méthode du DBSCAN clustering. Les avantages de cette méthode qui ont justifié son utilisation sont notamment sa capacité à identifier des clusters de formes variées ainsi que repérer et gérer les données aberrantes. Sur ce dernier point, la particularité du DBSCAN est que cette méthode va identifier les données aberrantes comme un groupe à part. Ainsi, nous avons pu obtenir les résultats visible en fig.24, avec en noir les données marquées comme aberrantes. Cette fois-ci les données obtenues correspondent à nos attentes, le seul problème étant que nous avons donc du abandonné l'idée de pouvoir classifier les données aberrantes parmi nos 3 clusters.

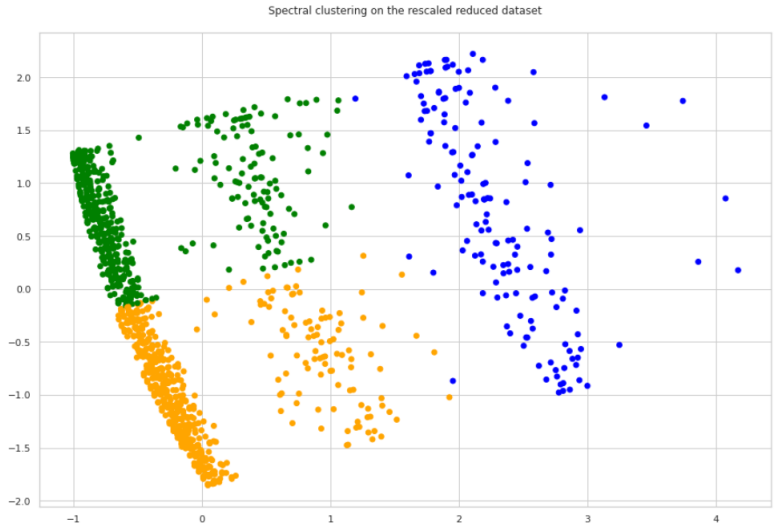

FIGURE 23 – Spectral clustering

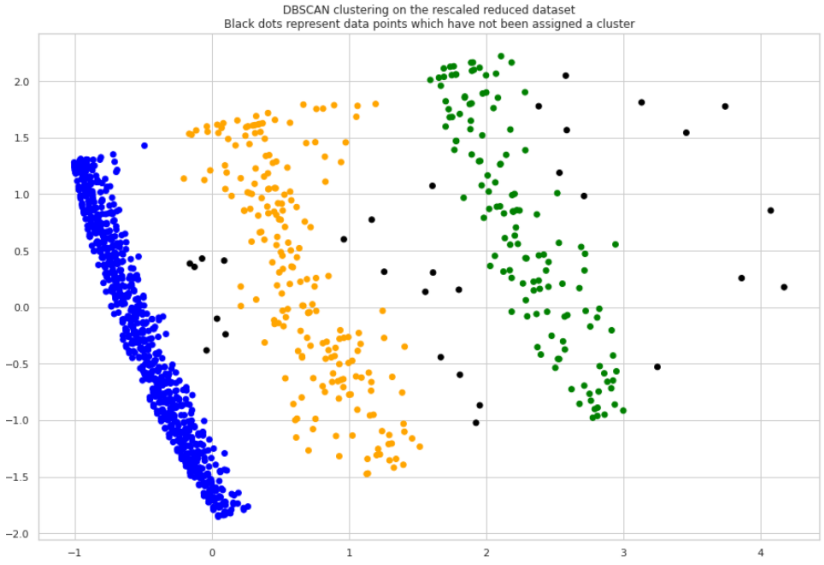

FIGURE 24 – DBSCAN clustering