Projet d’apprentissage non supervisé sur les vins

07.01.2017

Elaboré par:
BENNISS Salma
TOUZI Marwen

Table des matières

I. Introduction et objectives:

Nous avons sous nos mains un jeu de données qui contient les composantes chimiques de quelques vins et leurs qualités. Durant ce projet nous avons essayé de répondre à ces questions:

- Est-ce que la centralisation et la normalisation des données facilitent et améliorent nos prédictions?
- Quelles sont les types de segmentations que nous pouvons entraîner pour ce jeu de données et comment pouvons-nous les comparer?
- En se basant sur les préférences d'un buveur, comment pouvons-nous construire un algorithme qui recommande des vins susceptibles à lui plaire.

II. Statistique Descriptive:

1. Statistique univariée:

Notre jeu de données est constitué par 4898 lignes qui présentent nos observations et 12 colonnes qui présentent nos variables.

Nom variable:	Type:	Mean & Variance:	Commentaires:
fixed acidity	Float	mean is:6.85478766844 var is:0.7121135857	
fixed acidity	Float	mean is:0.278241118824 var is:0.0101595409922	
citric acid	Float	mean is:0.334191506737 var is:0.0146457930093	
residual sugar	Float	mean is:6.39141486321 var is:25.7257701644	Besoin d'une centralisation
chlorides	Float	mean is:0.0457723560637 var is:0.000477333709825	
free sulfur dioxide	Float	mean is:35.3080849326 var is:289.242719999	Besoin d'une centralisation
total sulfur dioxide	Float	mean is:138.360657411 var is:1806.08549085	Besoin d'une centralisation
density	Float	mean is:0.99402737648 var is:8.94552418578e-06	Besoin d'une centralisation
pH	Float	mean is:3.18826663944 var is:0.0228011810841	
sulphates	Float	mean is:0.489846876276 var is:0.0130247059745	
alcohol	Float	mean is:10.5142670478 var is:1.51442698179	
quality	Int	mean is:5.87790935076 var is:0.784355685471	7 modalités, nous pouvons la considérer comme variable qualitative.

Comme nous avons différentes unités pour les variables, une des alternatives qui peut améliorer notre phase de modélisation est la normalisation des variables, ce qui revient à diviser toutes les données de chaque variable par l'écart type de cette variable. De plus, comme nous avons constaté que plusieurs variables possèdent une grande variance donc elles sont bien dispersées dans l'espace, une phase de centralisation sera bien utile. Voilà le code utilisé pour **la centralisation** et **la normalisation** de nos variables:

```
#data.quality=data.quality.astype("category")
import math
data["fixed acidity"]=((data["fixed acidity"])/math.sqrt(data["fixed acidity"].var()))
data["volatile acidity"]=((data["volatile acidity"])/math.sqrt(data["volatile acidity"].var()))
data["citric acid"]=((data["citric acid"])/math.sqrt(data["citric acid"].var()))
data["residual sugar"]=((data["residual sugar"]-data["residual sugar"].mean())/math.sqrt(data["residual sugar"].var()))
data["chlorides"]=((data["chlorides"])/math.sqrt(data["chlorides"].var()))
data["free sulfur dioxide"]=((data["free sulfur dioxide"]-data["free sulfur dioxide"].mean())/math.sqrt(data["free sulfur dioxic
data["total sulfur dioxide"]=((data["total sulfur dioxide"]-data["total sulfur dioxide"].mean())/math.sqrt(data["total sulfur di
data["density"]=((data["density"]-data["density"].mean())/math.sqrt(data["density"].var()))
data["pH"]=((data["pH"])/math.sqrt(data["pH"].var()))
data["sulphates"]=((data["sulphates"])/math.sqrt(data["sulphates"].var()))
data["alcohol"]=((data["alcohol"])/math.sqrt(data["alcohol"].var()))
```

Nous avons constaté aussi que notre jeu de données ne contient pas de **données manquantes**:

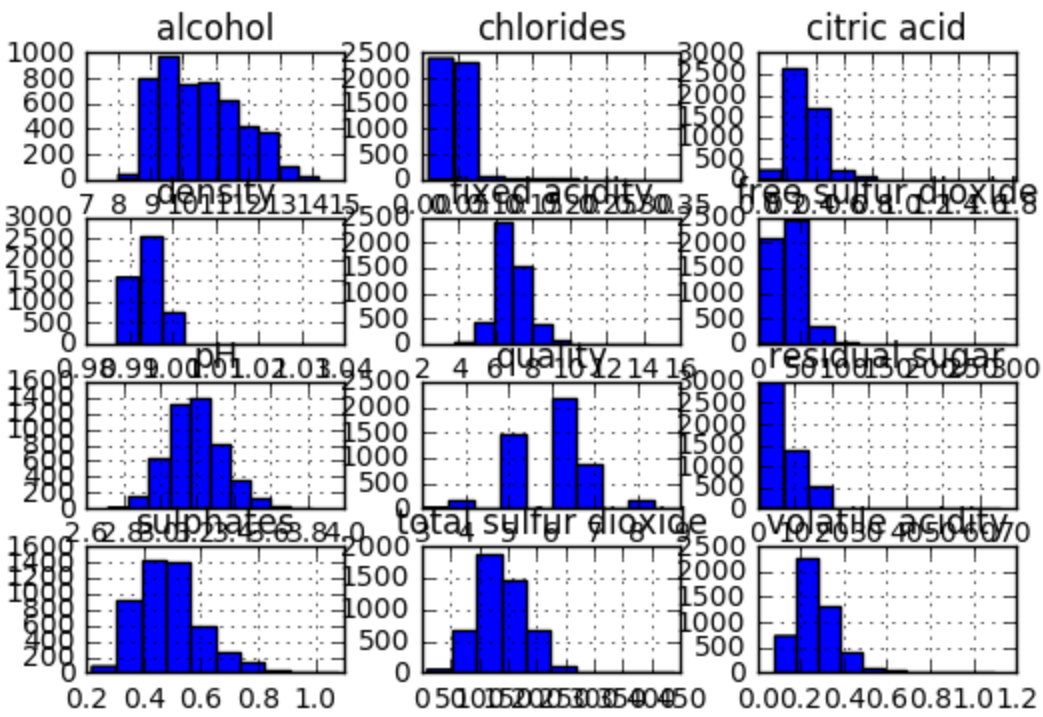
```
In [7]: data.isnull().sum()

Out[7]: fixed acidity           0
        volatile acidity        0
        citric acid             0
        residual sugar          0
        chlorides               0
        free sulfur dioxide     0
        total sulfur dioxide    0
        density                 0
        pH                      0
        sulphates               0
        alcohol                 0
        quality                 0
        dtype: int64
```

D'autre part, d'après nos notions en chimie, toutes les données de notre jeu de données sont raisonnables. Par exemple le ph est un réelle entre 0 et 7 ce qui est validé sur toutes les données de notre échantillon.

Donc nous avons conclu que notre base ne contient pas des **données aberrantes**.

L'étude des **données extrêmes** sera étudiée dans la partie de modélisation vu qu'il est plus délicat et nous allons le faire seulement sur les données issues des variables les plus significatives dans les modèles que nous allons construire.

Regardons maintenant la distribution de nos variables:



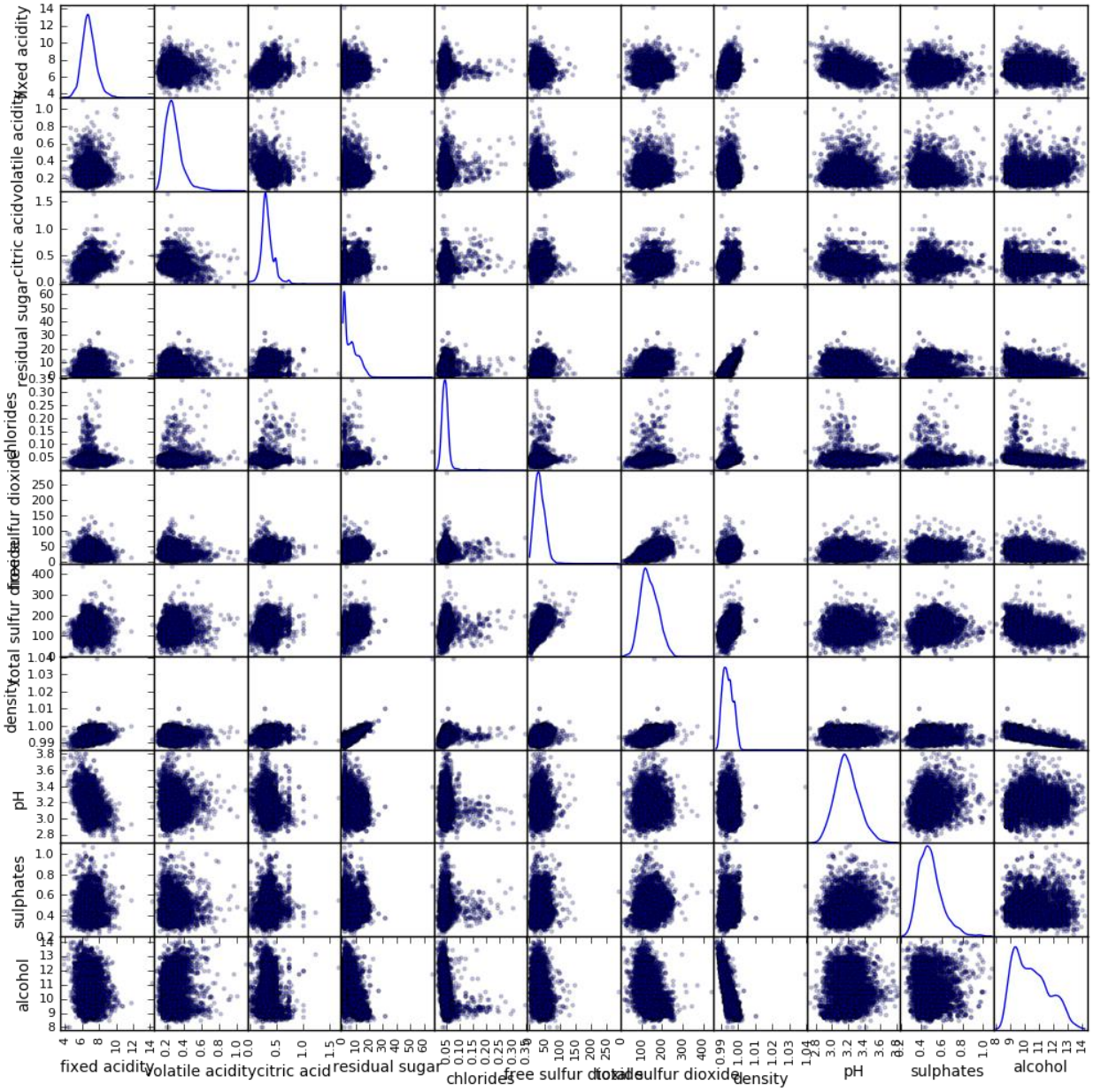
Nous remarquons que les variables ph, volatile acidity, total sulfure dioxide et sulfates suivent la loi normale. Nous pouvons pas dire grand-chose sur les autres. En tous cas, nous aurons pu utiliser ces constatations pour la prédiction des données manquantes, ce qui n'est pas notre cas.

2. Statistique multivariée:

Voilà la matrice qui présente les corrélations entre les variables de notre jeu de données:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
fixed acidity	1.000000	-0.022697	0.289181	0.089021	0.023086	-0.049396	0.091070	0.265331	-0.425858	-0.017143	-0.120881	-0.113663
volatile acidity	-0.022697	1.000000	-0.149472	0.064286	0.070512	-0.097012	0.089261	0.027114	-0.031915	-0.035728	0.067718	-0.194723
citric acid	0.289181	-0.149472	1.000000	0.094212	0.114364	0.094077	0.121131	0.149503	-0.163748	0.062331	-0.075729	-0.009209
residual sugar	0.089021	0.064286	0.094212	1.000000	0.088685	0.299098	0.401439	0.838966	-0.194133	-0.026664	-0.450631	-0.097577
chlorides	0.023086	0.070512	0.114364	0.088685	1.000000	0.101392	0.198910	0.257211	-0.090439	0.016763	-0.360189	-0.209934
free sulfur dioxide	-0.049396	-0.097012	0.094077	0.299098	0.101392	1.000000	0.615501	0.294210	-0.000618	0.059217	-0.250104	0.008158
total sulfur dioxide	0.091070	0.089261	0.121131	0.401439	0.198910	0.615501	1.000000	0.529881	0.002321	0.134562	-0.448892	-0.174737
density	0.265331	0.027114	0.149503	0.838966	0.257211	0.294210	0.529881	1.000000	-0.093591	0.074493	-0.780138	-0.307123
pH	-0.425858	-0.031915	-0.163748	-0.194133	-0.090439	-0.000618	0.002321	-0.093591	1.000000	0.155951	0.121432	0.099427
sulphates	-0.017143	-0.035728	0.062331	-0.026664	0.016763	0.059217	0.134562	0.074493	0.155951	1.000000	-0.017433	0.053678
alcohol	-0.120881	0.067718	-0.075729	-0.450631	-0.360189	-0.250104	-0.448892	-0.780138	0.121432	-0.017433	1.000000	0.435575
quality	-0.113663	-0.194723	-0.009209	-0.097577	-0.209934	0.008158	-0.174737	-0.307123	0.099427	0.053678	0.435575	1.000000

Nous remarquons que nous possédons également une bonne corrélation entre ces variables (density, residual sugar) et (density, alcohol). Voilà une autre présentation de cette matrice qui présente une forme quasi linéaire entre ces deux variables couples de variables:



Nous remarquons aussi, que notre variable quality n'est pas corrélé avec n'importe quelles autres variables.

Pour vérifier la règle qui dit qu'une centralisation et une normalisation ne change pas la corrélation entre les variables, nous avons centralisé et normalisé nos données puis, nous avons construit la nouvelle matrice et nous l'avons trouvé identique à la première.

Regardons maintenant si nous obtiendrons les mêmes résultats si nous passons par une réduction de dimensions.

III. Analyse factorielle:

Cette analyse a pour but de réduire le nombre des variables en calculant des variables latentes comme combinaison linéaires des variables observées. Pour notre cas, vu que toutes nos variables sont des variables numériques, nous avons opté pour l'ACP ou l'analyse en composantes principales.

Le premier résultat obtenu est le pourcentage d'inertie de chaque axe qui nous sera utile pour la détermination de nombre d'axes nécessaires pour présenter notre jeu de données. Si nous passons par la centralisation et la normalisation, voilà ce que nous obtenons:

```
print(pca.explained_variance_ratio_)
```

```
[ 0.28106746  0.13416479  0.11124089  0.09048936  0.08272947]
```

Ce qui implique que les cinq premiers axes ensemble présentent seulement 69% de l'information de notre base. Si nous utilisons nos vrais données (non normalisées), nous obtenons:

```
print(pca.explained_variance_ratio_)
```

```
[  9.09331225e-01   7.93141114e-02   1.01514160e-02   6.19631057e-04
   3.23291244e-04]
```

Avec seulement deux axes, nous pouvons représenter plus que 90% de l'information de notre base.

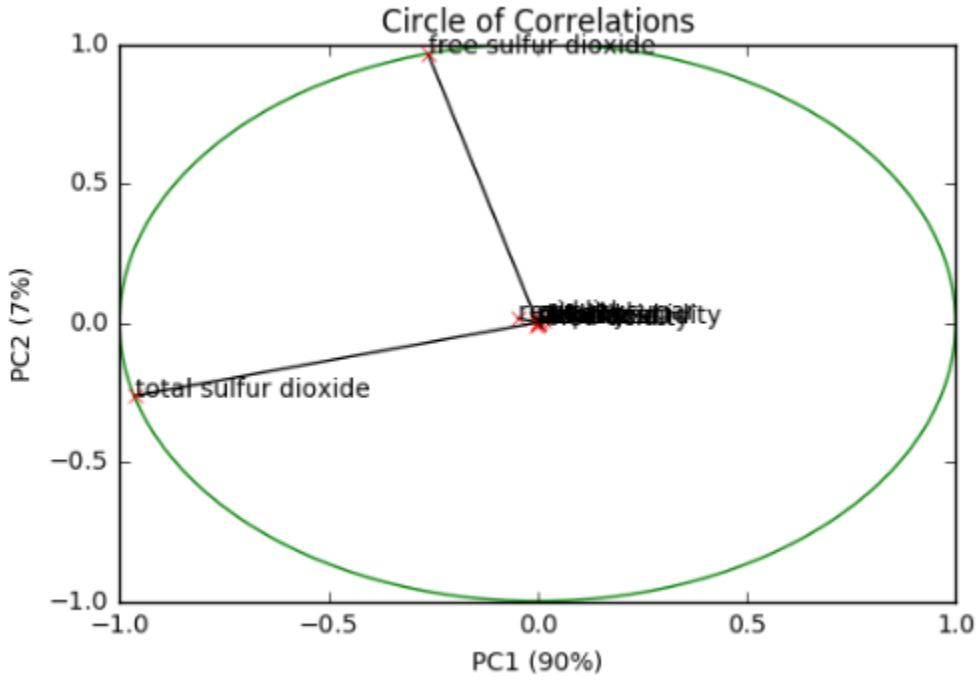
Conclusion, nous n'allons pas utiliser les données centrées réduites pour simplifier la tâche (travailler sur deux axes au lieu de cinq) et pour améliorer nos résultats (obtenir des interprétations sur plus 90% au lieu de 70%).

Comme nous allons nous intéresser par les 2 premiers axes, voilà les coordonnées de la projection de notre base de données dans le nouveau plan:

```
pd.DataFrame(np.transpose(pca.components_), columns=['PC1','PC2'], index=data.columns)
```

	PC1	PC2
fixed acidity	-0.001545	-0.009167
volatile acidity	-0.000169	-0.001546
citric acid	-0.000339	0.000140
residual sugar	-0.047328	0.014931
chlorides	-0.000098	-0.000072
free sulfur dioxide	-0.261872	0.964638
total sulfur dioxide	-0.963853	-0.262682
density	-0.000036	-0.000018
pH	-0.000003	-0.000041
sulphates	-0.000341	-0.000361
alcohol	0.012504	0.006480
quality	0.003280	0.010993

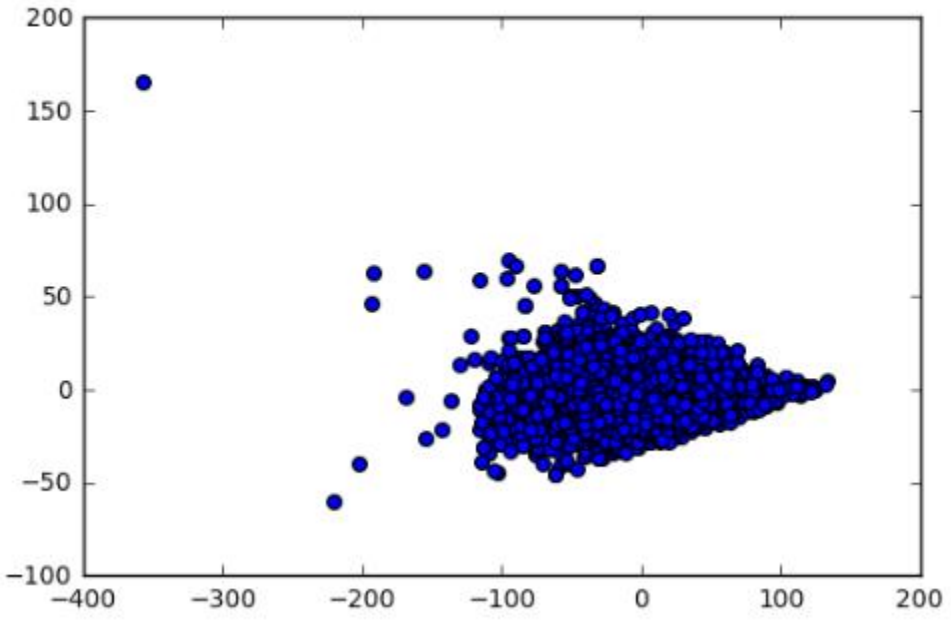
Et voilà le cercle de corrélation de notre premier plan:



La partie qui nous intéresse le plus dans ce projet est la suivante. Nous allons commencer à présenter nos observations sur ce plan factorielle et, à chaque fois, nous allons colorier nos segments ou clusters à l'aide d'une des techniques (de modélisation non supervisé) que nous allons l'expliquer dans la partie suivante. Le

but est de construire le meilleur modèle qui réussit à séparer les observations. Voilà la représentation de nos individus sans clustering:

```
plt.scatter(existing_2d[:,0], existing_2d[:,1], 30,)
plt.show()
```



IV. Modélisation:

1. La classification ascendante hiérarchique CAH

a) La classification hiérarchique avec Scipy

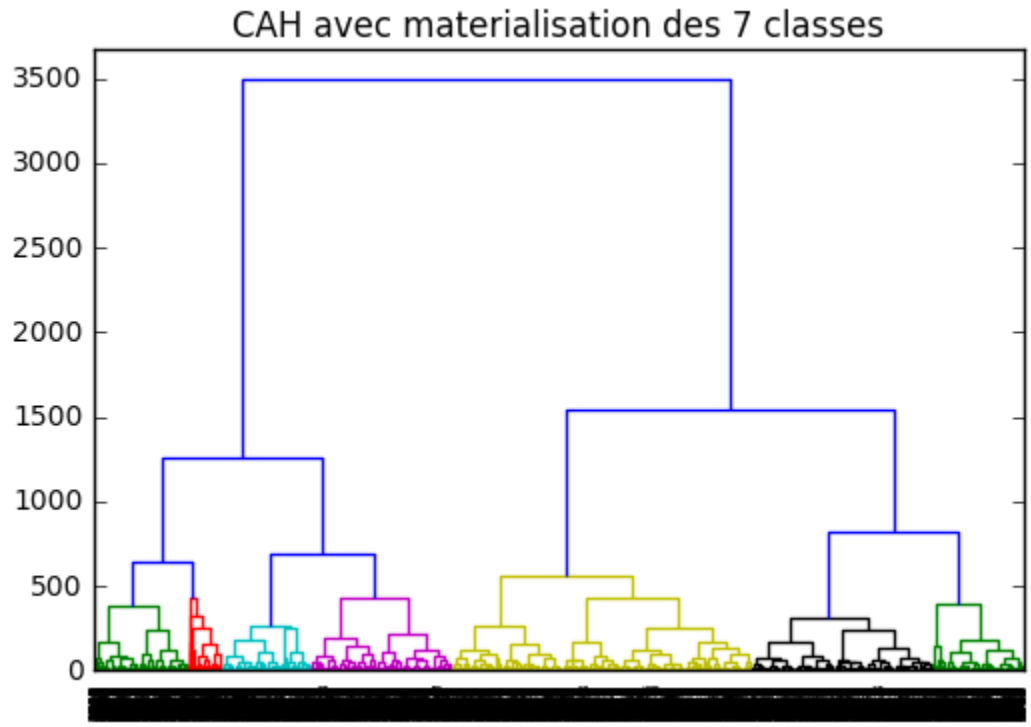
La classification ascendante hiérarchique (CAH) est une méthode de classification itérative dont le principe est simple.

1. On commence par calculer la dissimilarité entre les N objets.
2. Puis on regroupe les deux objets dont le regroupement minimise un critère d'agrégation donné, créant ainsi une classe comprenant ces deux objets.
3. On calcule ensuite la dissimilarité entre cette classe et les N-2 autres objets en utilisant le critère d'agrégation. Puis on regroupe les deux objets ou classes d'objets dont le regroupement minimise le critère d'agrégation.

On continue ainsi jusqu'à ce que tous les objets soient regroupés.

Ces regroupements successifs produisent un arbre binaire de classification (dendrogramme), dont la racine correspond à la classe regroupant l'ensemble des individus. Ce dendrogramme représente une hiérarchie de partitions. On peut alors choisir une partition en tronquant l'arbre à un niveau donné, le niveau dépendant

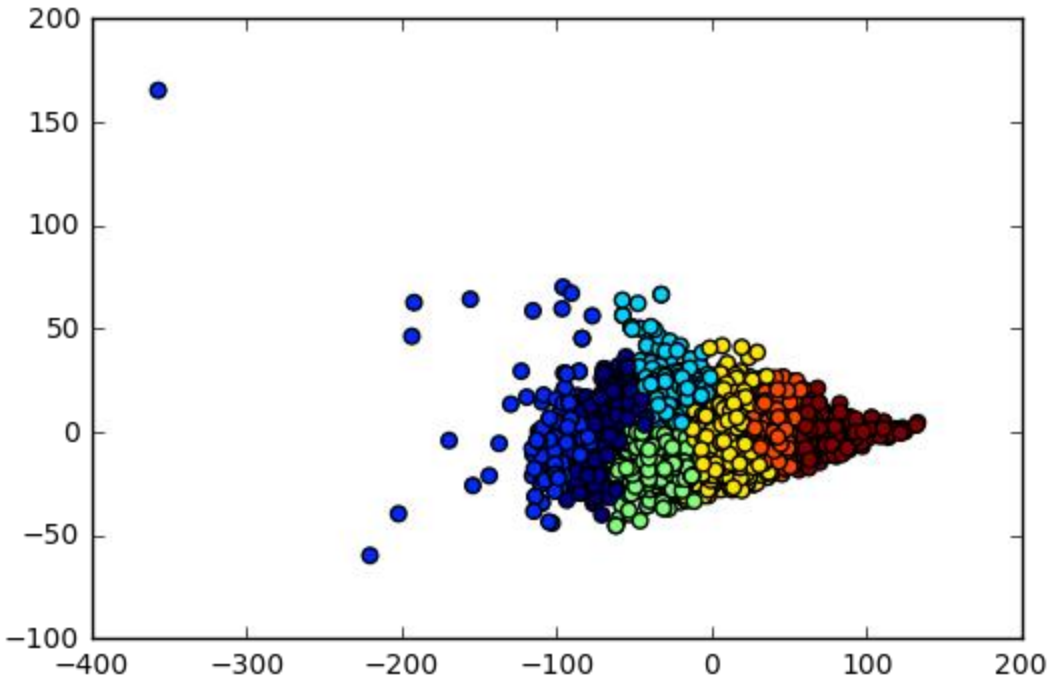
soit des contraintes de l'utilisateur (l'utilisateur sait combien de classes il veut obtenir), soit de critères plus objectifs.




```
groupes_cah = fcluster(Z,t=600,criterion='distance')
md=pd.Series(groupes_cah)
data_norm['Clust_CAH']=md
data_norm.groupby('Clust_CAH').mean()
```

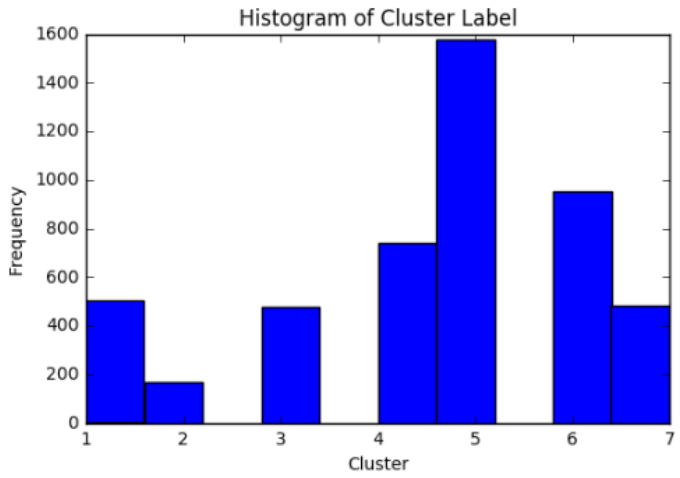
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
Clust_CAH											
1	6.959681	0.288842	0.360339	9.923752	0.053705	54.156687	197.535928	0.996575	3.175170	0.511357	9.575449
2	7.008284	0.313343	0.378107	10.739053	0.048402	62.286982	237.000000	0.997114	3.155740	0.520533	9.485799
3	6.738105	0.251716	0.354105	8.370000	0.049699	57.308421	156.952632	0.995131	3.185137	0.492695	10.085158
4	7.058593	0.298099	0.342963	7.779229	0.050432	33.386333	173.004060	0.995442	3.193342	0.503911	10.036085
5	6.813982	0.279004	0.325536	5.926982	0.044536	31.501585	130.124604	0.993682	3.194052	0.485574	10.644386
6	6.766317	0.260346	0.321480	4.088615	0.041151	27.193075	101.436516	0.992234	3.196474	0.478793	11.186236
7	6.803099	0.283471	0.312087	3.203616	0.038804	16.101240	71.061983	0.991726	3.173492	0.468285	11.249346

Pour visualiser concrètement nos clusters obtenus grâce à CAH, on crée une nouvelle variable qu'on appelle "Clust_CAH", et on la projette sur les deux axes obtenus grâce à l'ACP



On peut voir clairement grâce à la coloration des clusters, que la méthode de CAH a donné de bon résultat, tel que les points les plus proches appartiennent au même cluster.
Intéressant nous à présent à la distribution de notre échantillon de vin sur nos clusters crées:

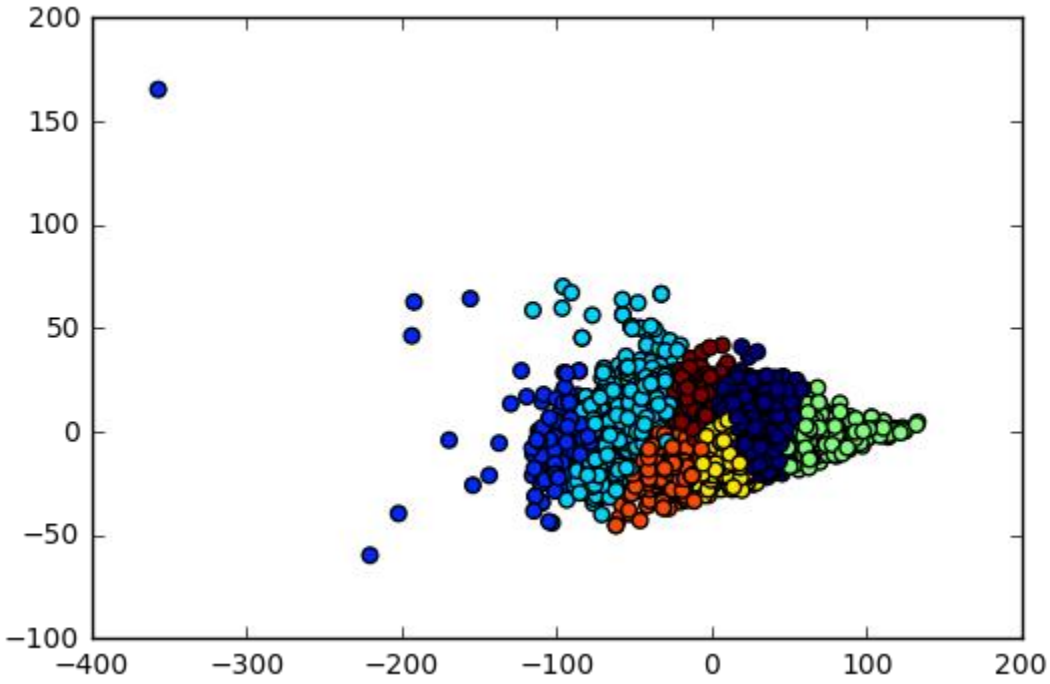
```
In [111]: plt.hist(data_wine['Clust_CAH'])
          plt.title('Histogram of Cluster Label')
          plt.xlabel('Cluster')
          plt.ylabel('Frequency')
Out[111]: <matplotlib.text.Text at 0x240cfeb8>
```



On peut dire que notre échantillon de vin est distribué d'une façon plus uniforme comparé à sa distribution basé sur la qualité du vin.

b) La classification hiérarchique avec scikit-learn

```
model_AggloClust = AgglomerativeClustering(n_clusters=7, linkage='ward')
ward=model_AggloClust.fit(data_norm)
md=pd.Series(ward.labels_)
data_norm['Clust_Hiearchical']=md
```



Pareil que la section précédente, grâce à cette méthode on a pu créer des clusters qui sont très bien représentés sur les deux axes de l'ACP.

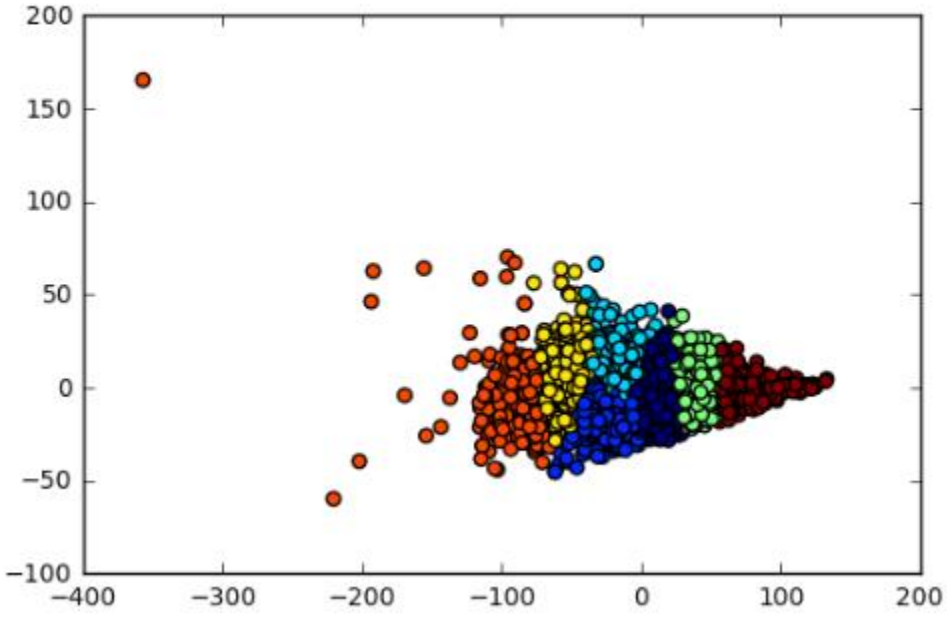
2. K-Means Clustering

K-Means est une méthode de partitionnement de données et un problème d'optimisation combinatoire. Étant donnés des points et un entier k, le problème est de diviser les points en k cluster, de façon à minimiser une certaine fonction. On considère la distance d'un point à la moyenne des points de son cluster ; la fonction à minimiser est la somme des carrés de ces distances.
Comme dans la partie on a décidé de diviser notre jeu de données en 7 cluster, ce qui correspond au nombre de groupe de qualité des vins.

```
data_norm.groupby('Clust_Kmeans').mean()
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
Clust_Kmeans												
0	6.826661	0.279432	0.325913	5.631791	0.043810	29.739697	126.764087	0.993457	3.189941	0.484121	10.736765	6.015139
1	7.098680	0.304183	0.344175	7.477723	0.050746	29.344884	166.419142	0.995301	3.185990	0.499620	10.097827	5.686469
2	6.745852	0.253941	0.346696	7.558296	0.047653	50.222963	149.541481	0.994634	3.203007	0.486519	10.280049	5.977778
3	6.746584	0.265493	0.319478	4.242362	0.041357	26.516603	100.667932	0.992306	3.193776	0.478055	11.150427	6.097723
4	6.896904	0.281029	0.352570	9.332817	0.051464	53.558824	187.318885	0.996153	3.182678	0.510155	9.706914	5.592879
5	7.036304	0.309010	0.361419	10.294224	0.051832	55.136964	227.123762	0.996981	3.172145	0.526733	9.526403	5.524752
6	6.833647	0.281953	0.312400	3.190588	0.039162	15.625882	68.816471	0.991776	3.169741	0.469294	11.211373	5.748235

```
plt.scatter(X_tmp[:,0], X_tmp[:,1], 30,data_norm['Clust_Kmeans'])
plt.show()
```



On remarque la présence d'une valeur extrême dans notre représentation, qu'on peut supprimer.

Dans un premier temps il a fallu récupérer l'indice de la valeur extrême.

Ensuite on a supprimé cette valeur de notre jeu de donnée.
Par la suite on a refait l'ACP pour projeter les labels des Clusters obtenus grâce à la méthode des K-Means sur les nouveaux axes.

Intéressons-nous maintenant à la distribution de notre échantillon de vins par rapports à nos Clusters :

Comme on peut observer, la distribution de notre échantillon de vin est plus uniforme sur nos clusters (obtenu par les K-Means) par rapport à la distribution basée sur la qualité du vin, et même par rapport aux clusters obtenus grâce à la méthode du clustering hiérarchique.

Il s'agit donc bien d'une amélioration de la classification basé uniquement sur la qualité de vin, car ça nous a permis de mieux identifier les clusters distincts.

On note également que le temps de calcul de la méthode des K-Means est nettement plus rapide que celui avec la méthode du clustering hiérarchique.

V. Systèmes de recommandation:

1. Non personnalisé:

On sait de manière général, que le goût et la qualité du vin dépend fortement de la quantité d'acide, alcool et sucre.

Toutefois nos résultats du clustering peuvent être très utiles en termes de recommandation par exemple.

Prenons les résultats obtenus grâce à la dernière méthode appliquée : K-Means:

- Les gens habitants dans les régions froides auront plutôt tendance à préférer les vins qui contiennent beaucoup de volatile acid, et dans ce cas il sera plus judicieux de faire une stratégie de marketing pour mettre plus en valeur les vins du cluster 1 et 5
- Beaucoup de gens préfèrent plutôt les vins qui sont forts en alcool, et donc pour cette gamme de clientèle, il sera plus judicieux de leur recommander les vins du cluster 3 et 6
- D'autres préfèrent plutôt choisir leur vins en tenant compte surtout des notations des experts et donc il faudra mieux leur recommander des vins appartenant au cluster 3

2. Plus personnalisé:

Nous allons expliquer ce système à l'aide d'un exemple. Soit un utilisateur qui a bu un ensemble de vins dont 70% appartiennent au premier cluster, 20% au deuxième et 10% au troisième. Nous allons calculer pour cet utilisateur 3 vecteurs que nous allons les appeler "**profile cluster i**" qui présentent pour un utilisateur la moyenne des variables des vins qu'il a essayé et qui appartiennent à ce cluster.

Supposons que nous voulons recommander 10 vins pour cet utilisateur, ce que nous allons faire est de lui recommander 7 vins du premier cluster, 2 vins du deuxième et un du troisième (selon les proportions qu'il a bu). Pour trouver par exemple les 7 vins du premier cluster, nous allons calculer la distance (en d'autre termes la corrélation) entre le vecteur "profile cluster 1" et tous les autre vins que cette utilisateur n'a jamais goûté et qui se trouvent dans ce même cluster. Une fois, nous avons trouvé ces 10 vins, nous allons les triés par les scores que nous avons obtenu par la corrélation. Ce système de recommandation présente une version modérée d'un système de recommandation hybride qui est composé par un algorithme de collaboratif filtering (l'événement principale est le fait qu'un utilisateur a bu un vin) et d'un algorithme content based (vu que nous nous basons sur les caractéristiques des vins et des utilisateurs).

Conclusion:

Le clustering peut être très utilisé pour faire de la recommandation et de mieux cibler les clients qui préfèrent des ingrédients particuliers, mais pas que.
Le clustering peut également être très utile pour tout ce qui concerne la partie pricing ou segmentation.