Apprentissage non supervisé

Si nous supprimons l'information target nous tombons dans le cas d'algorithmes de clustering : nous pouvons toujours demander à ce que notre algorithme sépare nos données en 3 groupes sans connaissance de leurs véritables étiquettes.

Saura-t-il retrouver les groupes initiaux ?

Réduction de dimension

Dans le cas des algorithmes non supervisés il est très fréquent de disposer de très grandes quantités de paramètres. Ne sachant pas encore qui est responsable de quoi on a tendance à tout livrer à la machine.

Cela pose 2 problèmes:

- La visualisation des données, au delà de 3 paramètres notre cerveau est bien mal outillé pour se représenter les données
- La complexité des calculs, plus le nombre de paramètres est grand, plus nous aurons des calculs complexes et longs

Pour contourner ces problèmes il est courant de réduire la dimension du vecteur de données à quelque chose de plus simple. La difficulté est alors de réduire le nombre de paramètres tout en conservant l'essentiel de l'information, notamment les variations succeptibles de permettre le regroupement des données.

Plusieurs techniques de réduction sont disponibles avec Scikit-Learn.

Nous utiliserons pour cet exemple l'analyse en composante principale, dite PCA. Le module manifold propose aussi d'autres types d'algorithmes.

PCA est une technique linéaire de réduction de dimension qui a l'avantage d'être très rapide. Elle s'utilise simplement:

- Vous définissez le nombre de paramètres
- Vous alimentez l'algorithme avec les données à réduire
- Vous lancez la prédiction ici appelée réduction/transformation

In [49]:

```
from sklearn.decomposition import PCA
# Définition de l'hyperparamètre du nombre de composantes voulues
model = PCA(n_components=2)
# Alimentation du modèle
model.fit(iris.data)
# Transformation avec ses propres données
reduc = model.transform(iris.data )
```

Nous venons de réduire notre vecteur de 4 paramètres en 1 vecteur de 2 paramètres dont les variations sont censées être similaires.

Autrement dit nous devrions être capable de classer nos fleurs avec ces vecteurs réduits en ayant une qualité proche de celle utilisant les vecteurs originaux !

In [50]:

```
1 iris.data[:5]
```

Out[50]:

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

Consultation des données originales:

Et de leur version réduite:

In [51]:

```
1 reduc[:5]
```

Out[51]:

```
array([[-2.68412563,  0.31939725],
       [-2.71414169, -0.17700123],
       [-2.88899057, -0.14494943],
       [-2.74534286, -0.31829898],
       [-2.72871654,  0.32675451]])
```

In [52]:

```
1 df.head()
```

Out[52]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	label
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

Ajoutons les nouveaux paramètres dans le dataframe d'origine:

In [53]:

```
df['PCA1'] = reduc[:, 0]
df['PCA2'] = reduc[:, 1]
df.head()
```

Out[53]:

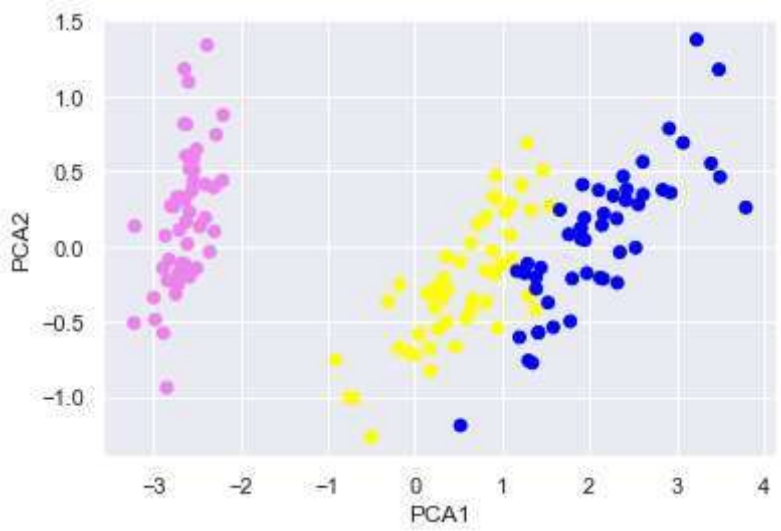
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	label	PCA1	PCA2
0	5.1	3.5	1.4	0.2	0	setosa	-2.684126	0.319397
1	4.9	3.0	1.4	0.2	0	setosa	-2.714142	-0.177001
2	4.7	3.2	1.3	0.2	0	setosa	-2.888991	-0.144949
3	4.6	3.1	1.5	0.2	0	setosa	-2.745343	-0.318299
4	5.0	3.6	1.4	0.2	0	setosa	-2.728717	0.326755

Puis affichons les nouveaux couples de points (PCA1, PCA2) avec la couleur de l'espèce associée:

In [54]:

```
colors = ['violet', 'yellow', 'blue']
plt.scatter(df['PCA1'], df['PCA2'], c=[ colors[c] for c in df['target'] ]);
plt.xlabel('PCA1')
plt.ylabel('PCA2');

#Nous obtenons 3 groupes plutôt bien dissociés !
```

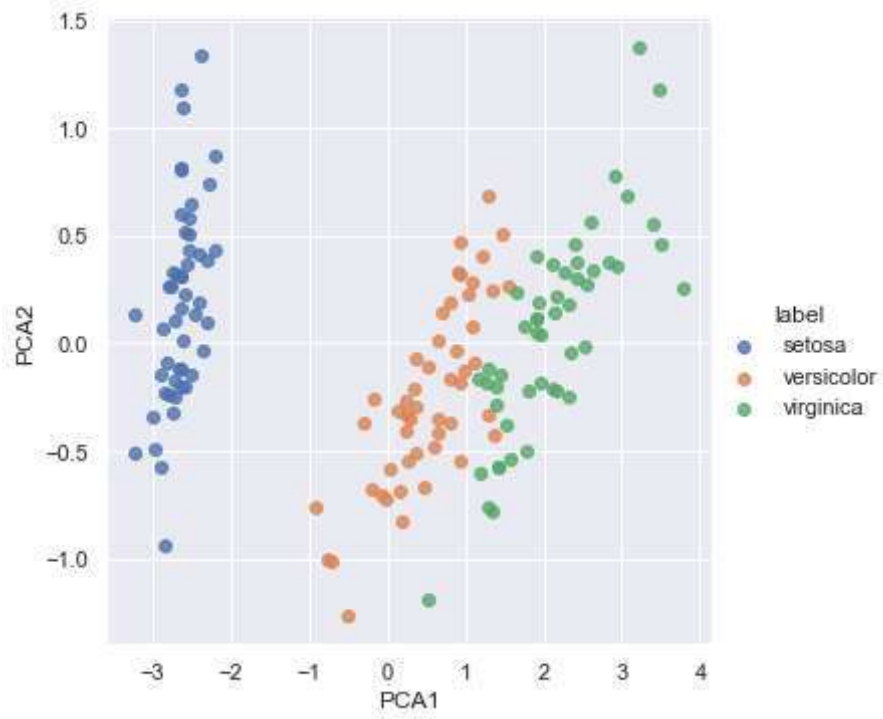


Ce qui peut aussi s'obtenir avec Seaborn

In [55]:

```
1 sns.lmplot("PCA1", "PCA2", hue='label', data=df, fit_reg=False);
```

```
C:\Users\MSI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other argu
ments without an explicit keyword will result in an error or misinterpretati
on.
  warnings.warn(
```



Le résultat est plutôt satisfaisant, et ce qui est assez bluffant c'est que l'algorithme ne connaîssait pas du tout les types de fleurs !

Maintenant, ce nouveau classement peut-il permettre un bon regroupement des 3 espèces ? le graphique semble le confirmer, vérifions cela avec le clustering

Clustering

Il ne reste plus qu'à exécuter le regroupement. Plusieurs algorithmes existent:

- k-Means, qui se rapproche des plus proches voisins dans sa logique, recherche les données proches des centres des clusters. Il est très simple mais n'est vraiment efficace que sur des données organisées en cercles. ici nous avons plutôt des droites ou des ellipses

- GMM, Gaussian Mixture Models est plus complexe mais s'adapate très bien à différentes formes de clusters (groupes).
- Scikit Learn en propose beaucoup d'autres comme Spectral clustering, Mean Shift, Hierarchical clustering, ...

Nous utiliserons GMM pour cet exemple.

In [56]:

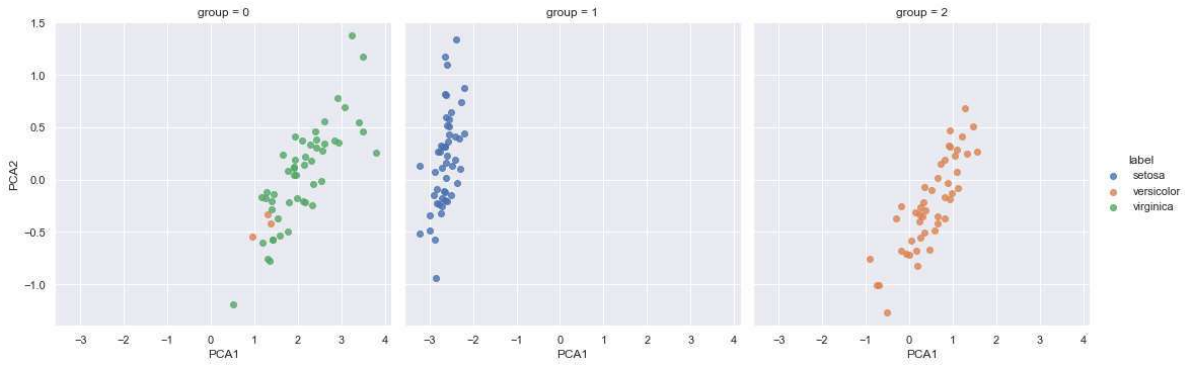
```
from sklearn.mixture import GaussianMixture
# Création du modèle avec 3 groupes de données
model = GaussianMixture (n_components=3, covariance_type='full')
# Apprentissage, il n'y en a pas vraiment
model.fit(df[['PCA1', 'PCA2']])
# Prédiction
groups = model.predict(df[['PCA1', 'PCA2']])
```

La prédiction étant faîte, pour chaque groupe généré nous affichons la couleur réelle des espèces, si le clustering a été efficace, il n'y aura qu'une seule couleur par groupe:

In [57]:

```
df['group'] = groups
sns.lmplot("PCA1", "PCA2", data=df, hue='label',
           col='group', fit_reg=False);
```

```
C:\Users\MSI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWa
rning: Pass the following variables as keyword args: x, y. From version 0.1
2, the only valid positional argument will be `data`, and passing other argu
ments without an explicit keyword will result in an error or misinterpretati
on.
  warnings.warn(
```



Le groupe setosa est très bien identifié, il y a toujours une imprécision entre les classes virginica et versicolor mais le résultat reste remarquable sur ce petit échantillon.

Vue d'ensemble - The big picture

Pour vous aider à choisir vos algorithmes d'apprentissage automatique, nous reprenons cette image de Scikit-Learn qui pose assez bien la problématique:

Pour aller plus loin

Il est important de disposer de ressources pour s'essayer au machine learning

Le site Kaggle propose de nombreuses ressources (jeux de données et exemples) sur le machine learning. Il propose aussi des compétions ou les membres peuvent comparer leurs algorithmes Beaucoup de sites proposent des jeux de données:

Open data du gouvernement français Enigma.io Une recherche Internet sur Open data vous donnera beaucoup de ressources

Conclusion

Nous voilà initiés au Machine Learning avec Scikit-Learn. La librairie propose de nombreux exemples et jeux de données. Elle est extrêmement riche et simple.

Les difficultés de cette discipline consistent à:

- comprendre les notions mathématiques derrière chaque algorithme pour avoir une idée de leurs limites
- choisir les hyperparamètres
- bien dimensionner ses jeux de données d'apprentissage

Ensuite, quelque soit votre algorithme c'est très simple:

- On instancie la classe et ses hyper-paramètres
- On fournit les données d'apprentissage à la méthode fit (si supervisé)
- On demande la détermination des données avec la méthode predict

In [ ]:

```
1
```