Travaux pratiques de
Machine learning
Cycle pluridisciplinaire d'études supérieures
Université Paris sciences et lettres

Joon Kwon

vendredi 27 mars 2020

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Nous allons travailler avec un jeu de données portant sur des vins. Nous allons le télécharger dans le dossier de travail depuis l'adresse suivante.

https://joon-kwon.github.io/cpes-ml/wine_dataset.csv

On veillera à ne pas ouvrir le fichier `csv` avec Excel pour l'enregistrer ensuite, car Excel peut altérer le fichier. Nous le chargeons sous la forme d'une `DataFrame` pandas et affichons ses 5 premières lignes comme suit.

```
data = pd.read_csv('wine_dataset.csv')
data.head()
```

Question 1. — Examiner la `DataFrame` `data`. Combien d'exemples et combien de variables le jeu de données contient-il ?

On souhaite apprendre à prédire le type de vin (rouge ou blanc) à partir des propriétés chimiques. La variable `quality` n'est pas une propriété chimique, mais une note subjective donnée par des personnes ayant goûté le vin ; elle n'est pas pertinente pour notre problème ; nous allons donc la supprimer. Pandas permet de manipuler (en l'occurrence, supprimer) les colonnes en les désignant par leurs noms.

```
data = data.drop(columns=['quality'])
```

D'anciennes versions de Pandas peuvent renvoyer une erreur pour la commande ci-dessus. On pourra alors essayer `data.drop(['quality'],axis=1)`. Avant de construire des prédicteurs, nous souhaitons sélectionner un petit nombre de variables explicatives (disons quatre) : celles qui semblent les plus prometteuses pour la prédiction du type de vin. Pour la variable `fixed_acidity` par exemple, nous pouvons visualiser sa répartition pour chaque type de vin de la façon suivante.

```
g = sns.FacetGrid(data=data, hue='style')
g.map(sns.distplot, 'fixed_acidity').add_legend()
plt.show()
```

On peut également tracer plusieurs graphes dans une même figure. Il faut d'abord donner un nom à la figure (`fig` ci-dessous), et ensuite déclarer chaque nouvelle sous-figure à l'aide de `fig.add_subplot()`.

```
fig = plt.figure()

fig.add_subplot()
g = sns.FacetGrid(data=data, hue='style')
g.map(sns.distplot, 'fixed_acidity').add_legend()

fig.add_subplot()
g = sns.FacetGrid(data=data, hue='style')
g.map(sns.distplot, 'volatile_acidity').add_legend()
```

Question 2. — Produire une figure qui contient les visualisations pour chaque variable explicative. On pourra faire intervenir une boule `for`, ainsi que `data.columns` qui donne la liste des noms des variables.

Question 3. — En s'appuyant sur les figures obtenues, choisir quatre variables qui semblent les plus prometteuses pour la prédiction du type de vin. Créer une `DataFrame` `data_reduced` ne contenant que ces variables sélectionnées ainsi que la variable à prédire. On pourra utiliser la syntaxe suivante.

```
data_reduced = data[['column_name1','column_name2']]
```

Nous allons à présent séparer le jeu de données en des échantillons de test et d'apprentissage. Nous pouvons pour cela faire appel à une fonction dédiée fournie par `scikit-learn` qui s'occupe également de mélanger l'ordre des exemples.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
 ↪  train_test_split(data_reduced.drop(columns='style'),data_reduced['style'])
```

Question 4. — Observer les échantillons obtenus et déterminer le ratio de leurs tailles.

Nous allons à présent entraîner une régression logistique.

```
from sklearn.linear_model import LogisticRegression

f = LogisticRegression()
f.fit(X_train, y_train)
```

Question 5. — Afficher les erreurs d'apprentissage et de test du prédicteur construit. On pensera à utiliser la fonction `f.score()`.

Question 6. — Entraîner également un prédicteur Perceptron qu'on pourra charger à l'aide de la commande

```
from sklearn.linear_model import Perceptron
```

ainsi que des prédicteurs kNN (pour différentes valeurs de k). Comparer les résultats.

Nous allons à présent consulter, par exemple pour le prédicteur `f`, la *matrice de confusion* calculée sur l'échantillon de test.

```
pd.crosstab(y_test,f.predict(X_test))
```

Question 7. — Expliquer ce que représente la matrice de confusion.

On remarque en consultant la matrice de confusion que l'échantillon de test contient nettement plus de vins blancs que de vins rouges. On en déduit en particulier que lorsqu'on évalue la qualité des prédicteurs avec l'échantillon de test, la justesse des prédicteurs sur les vins blancs influence davantage le score que la justesse sur les vins rouges.

Question 8. — À partir de `X_test` et `y_test`, construire un sous-échantillon de test (`X_test_` et `y_test_`) qui contienne des vins rouges et blancs en quantités égales. On pourra faire appel à la fonction `resample`, dont on pourra consulter la documentation, et qu'on chargera au préalable avec la commande suivante.

```
from sklearn.utils import resample
```

Question 9. — Évaluer sur ce sous-échantillon la qualité des prédicteurs construits. Commenter les résultats.

Question 10. — De la même manière, créer un sous-échantillons d'apprentissage `X_train_` et `y_train_`. Entraîner les prédicteurs sur ce sous-échantillon et observer leurs scores. Commenter.

Question 11. — Entraîner les estimateurs en utilisant l'ensemble des variables explicatives initialement disponibles. Commenter les résultats.