U.E. 5.2 Contrôle continu 5

le 29/11/2017

Nom, prénom : /10

*Durée : **15 minutes**.*

Accès à python, cours et google.

Dans ce qui suit, vous aurez besoin d'un terminal ipython ou un jupyter notebook, ainsi que d'un navigateur web.

Entrer les commandes suivantes :

```
import numpy as np
from sklearn import datasets
data = datasets.load_boston()
```

Une description du dataset est accessible avec la commande `print(data.DESCR)`. Les variables explicatives (ou features) sont accessible depuis `data.data`, et la variable à prédire est dans `data.target`.

Question 1 :

La commande `np.savetxt()` permet d'écrire un `numpy.array` dans un fichier `csv`. Elle s'utilise de la façon suivante :

`numpy.savetxt(`*fname, X, fmt='%.18e', delimiter=' ', newline='\n', header='', footer='', comments='# '*`)`

- A quoi servent les les arguments suivants :

fname : nom du fichier dans lequel est sauvegardé la donnée

X : donnée à écrire dans le fichier

Delimiter : caractère servant à séparer les colonnes

newline : caractère servant à séparer les lignes (retour à la ligne par défaut '\n')

2/ Ecrire la commande qui permet de sauvegarder le tableau `data.data` dans le fichier `"data.csv"` en utilisant la commande `np.savetxt()`

```
np.savetxt("data.csv", data.data)
```

3/ Ecrire la même commande, mais en utilisant un ";" comme séparateur entre les colonnes

```
np.savetxt("data.csv", data.data, delimiter=";")
```

4/ Ecrire la commande pour lire le fichier "data.csv" dans la variable `newdata` avec la commande `np.loadtxt()`

```
newdata = np.loadtxt("data.csv", delimiter=";")
```

TSVP

Question 2 :
1/ Que fait la commande suivante : `X = data.data[:,-1]`

Elle récupère et assigne à la variable `X` toutes les lignes de la dernière colonne de `data.data`

2/ Après avoir exécuté les commandes suivantes :

```
from sklearn.model_selection import train_test_split
y = data.target
X = X.reshape(-1,1)
y = y.reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8)
```

- Expliquer ce que fait la dernière commande qui utilise `train_test_split`

Elle sépare les données en 2 groupes, *train* et *test*, en allouant aléatoirement 80% à *train*, afin de préparer la cross-validation, où on fait le *fit* sur *train* et on valide sur *test*.

Question 3 :
1 / Entrer les commandes pour importer et instancier la classe `regr = LinearRegression()` du module `linear_model` de scikit-learn

```
from sklearn.linear_model import LinearRegression
regr = LinearRegression()
```

2/ Faire un fit en utilisant `X_train` et `y_train`

```
regr.fit(X_train, y_train)
```

3/ Donner les commandes et les valeurs de la pente et l'ordonnée à l'origine de la droite

```
print(regr.coef_, regr.intercept_)
```

4/ Représenter graphiquement les points `X_train` et `y_train` ainsi que la droite à l'aide de la librairie `matplotlib`

```
import matplotlib.pyplot as plt
plt.scatter(X_train, y_train)
plt.plot(X_train, regr.predict(X_train))
plt.show()
```

5/ Représenter l'histogramme des résidus

```
plt.hist(y_train - regr.predict(X_train))
plt.show()
```