TD : `pandas`, `numpy`, `matplotlib` & introduction à `sklearn`

Romain Tavenard

Dans cette séance, nous nous focaliserons sur la préparation de vos données en vue d'une utilisation avec `sklearn`.

Pour disposer de tous les objets / fonctions dont vous aurez besoin, commencez votre code avec l'en-tête suivante :

```
import numpy
import pandas
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Imputer, StandardScaler
```

1 Cas des variables catégorielles

Dans la suite de cette section, on suppose que vous travaillez avec un *dataframe* `pandas`. Si vos données sont stockées dans un *array* `numpy`, il est toujours possible (au pire) de le transformer en *dataframe* `pandas` (cf la doc `pandas`).

Lorsque vous êtes confrontés à des variables catégorielles, deux cas de figures doivent être distingués :

— si la variable catégorielle est la variable à prédire (dans un problème de classification supervisée), vous n'avez pas à vous en soucier ;
— si la variable catégorielle est une variable explicative, vous devrez la transformer avant de l'utiliser. Pour cela, la fonction `get_dummies()` de `pandas` vous sera d'une aide précieuse.

1. Importez le contenu du fichier `us_people_with_nans.csv` dans un *dataframe* (attention, il contient des valeurs manquantes qui peuvent soit correspondre à une cellule vide `""` ou à une valeur `"NA"`).
2. Encodez la variable `"Sex"` en formant trois nouvelles colonnes numériques, une pour chaque modalité (`"f"` pour féminin, `"m"` pour masculin et `"n"` pour neutre).

2 Gestion des valeurs manquantes

Dans toutes les questions qui suivent, il vous sera demandé de modifier le *dataframe* initial pour gérer le problème des valeurs manquantes. Pour ne pas écraser ce *dataframe*, il sera judicieux à chaque fois d'en faire une copie et de travailler sur cette copie :

```
df_new = df.copy()
```

3. Comme précisé plus haut, le jeu de données contient des valeurs manquantes. Commencez par vous simplifier la vie en supprimant (à l'aide de la fonction `dropna()` de `pandas`) toutes les lignes contenant des valeurs manquantes.
4. En repartant de votre *dataframe* initial, supprimez plutôt les **colonnes** contenant des valeurs manquantes.
5. **Question subsidiaire : à faire lorsque vous aurez fini le reste du TD.** Aucune de ces deux solutions n'est satisfaisante. Une autre solution pour gérer ces valeurs manquantes est de les imputer. Pour cela, on utilisera la classe `Imputer` de `sklearn`.

3 Mise à l'échelle des données

Pour un grand nombre de modèles de *machine learning*, il est préférable de travailler sur des données mises à l'échelle (par exemple centrées-réduites, ou à valeurs entre `0` et `1`).

Pour cela, on va utiliser le module `preprocessing` de `sklearn` qui contient tout un tas de méthodes pour préparer vos données avant de les utiliser pour apprendre un modèle (c'est d'ailleurs là que la classe `Imputer` citée plus haut est définie). Pour tous ces pré-traitements, le fonctionnement est le même :

a. on construit un objet de la classe visée en lui précisant certains hyper-paramètres ;
b. on appelle sa méthode `fit` pour ajuster les paramètres (par exemple moyenne et écart-type dans le cas où l'oin souhaite centrer-réduire nos données) ;
c. on appelle sa méthode `transform` pour appliquer la transformation à nos données.

Attention, les objets `sklearn` attendent des données quantitatives pour être estimés. Vous devrez donc ne leur passer qu'un sous-ensemble de votre *dataframe* composé de ses colonnes quantitatives.

6. Utilisez la classe `StandardScaler` pour centrer-réduire les données des colonnes `"Age"` et `"Height"` de votre *dataframe*.

7. **Question subsidiaire : à faire lorsque vous aurez fini le reste du TD.** Appliquez la transformation inverse pour retrouver vos données initiales.

4 Génération de données synthétiques & régression linéaire

Dans la suite, vous allez tirer des données aléatoirement. Pour que vos expériences soient répétables, vous **devez**, avant toute chose, initialiser la graine de votre générateur aléatoire :

```
numpy.random.seed(0)
```

Notez que, sauf indication contraire, `sklearn` utilise l'état courant du générateur aléatoire de `numpy`, donc en fixant cette graine, vous rendez répétable le comportement de `numpy` ainsi que celui de `sklearn` pour la suite de votre programme.

8. À l'aide du module `numpy.random`, générez une matrice `X` de 100 observations en dimension 1 tirées d'une loi gaussienne centrée réduite. Générez également un vecteur `y` tel que :

$$\forall i, y_i = \sin(X_i) + \varepsilon_i, \text{ où } \varepsilon_i \sim N(0, 0.1)$$

9. Affichez ces données dans une fenêtre `matplotlib`.
10. Vous allez maintenant chercher à estimer les paramètres d'un modèle de régression linéaire (classe `LinearRegression`) à ces données. Pour ce faire, les deux premières étapes (création d'une instance de la classe et appel de la méthode `fit()`) seront identiques à celles évoquées plus haut pour la classe `StandardScaler`. La troisième étape consistera à obtenir les prédictions $(\hat{y}_i)$ du modèle à l'aide de la méthode `predict()`.
11. Quels sont les attributs des instances de la classe `LinearRegression` ? Quels sont leurs valeurs dans votre cas ?
12. Affichez, dans une fenêtre `matplotlib`, les données en bleu et les valeurs prédites correspondantes en rouge.

5 Régression Lasso

13. Générez une matrice `X` de 100 observations en dimension 10 tirées d'une loi gaussienne de moyenne nulle et dont la matrice de variance-covariance est égale à l'identité. Générez également un vecteur `y` tel que :

$$\forall i, y_i = \sin(X_{i,0}) + \varepsilon_i, \text{ où } \varepsilon_i \sim N(0, 0.1)$$

Jetez un oeil aux dimensions de `y`. Vous devriez avoir un vecteur colonne (*ie.* une matrice de dimension `(100, 1)`). Si ce n'est pas le cas, c'est qu'il faut redimensionner la sortie de la fonction `numpy.sin` à l'aide de la méthode `reshape`.

14. À l'aide de la fonction `train_test_split` du module `model_selection`, divisez votre jeu de données en un jeu d'apprentissage et un jeu de test, de tailles égales.
15. En utilisant uniquement les données d'apprentissage, estimez les paramètres d'un modèle `Lasso` (pour `alpha=0.2`). Affichez les paramètres estimés. Qu'en pensez-vous ?