TP1- Techniques de base pour le nettoyage des données Semi-structurées (avec Python). [Révision]

1. Introduction

Le nettoyage des données est une étape cruciale dans tout projet de science de données (particulièrement d'apprentissage automatique). Dans **les données tabulaires**, il existe de nombreuses **techniques d'analyse statistique et de visualisation des données** que vous pouvez utiliser pour explorer les données afin d'identifier les opérations de nettoyage des données. il existe des opérations de nettoyage de données très basiques que vous devriez probablement effectuer sur chaque projet de science de données. Ceux-ci sont si basiques quelles sont souvent négligés par les praticiens chevronnés de l'apprentissage automatique, mais ils sont **si critiques que s'ils sont ignorés**, les modèles peuvent casser ou signaler des résultats de performance trop optimistes. Dans ce TP, vous découvrirez quelques opérations sur le nettoyage de base des données que vous devez toujours effectuer sur votre ensemble de données **semi structurées.** Après avoir complété ce TP, vous saurez :

- ✓ Comment identifier et supprimer les variables de colonne qui n'ont qu'une seule valeur.
- ✓ Comment identifier et considérer les variables de colonne avec très peu de valeurs uniques.
- ✓ Comment identifier et supprimer les lignes contenant des observations en double.

2. Ensembles de données désordonnés

Un ensemble de données peut avoir de nombreux types d'erreurs, bien que certaines des erreurs les plus simples incluent des colonnes qui **ne contiennent pas beaucoup** d'informations ou **des lignes dupliquées.**

2.1 Oil Spill Dataset (Fichier .CSV) à telecharger sur classrom:

C'est un ensemble de données sur les **déversements d'hydrocarbures** est un ensemble de données d'apprentissage automatique standard. Utilisé pour prédire si le patch contient ou non un déversement d'hydrocarbures, par ex. du déversement illégal ou accidentel de pétrole dans l'océan, étant donné un vecteur qui décrit le contenu d'un patch d'un satellite image. Il y a **937 cas**. Chaque cas est composé de **48 caractéristiques dérivées** de la vision numérique par ordinateur, d'un numéro de patch et d'une étiquette de classe. Le cas normal est l'absence de déversement d'hydrocarbures avec l'étiquette de **classe 0,** alors qu'un déversement d'hydrocarbures est indiqué par une étiquette de classe 1. Il y a **896 cas** d'absence de déversement d'hydrocarbures et **41 cas** de déversement d'hydrocarbures.

2.2 Analyser visuellement le Dataset :

a- ouvrir le fichier **oil-spill.csv ,** discutez son contenue.

B. Identifier les colonnes qui contiennent une seule valeur : Les colonnes qui ont une **seule valeur** pour toutes les lignes ne contiennent aucune information pour la modélisation.

Selon le choix des algorithmes de préparation et de modélisation des données, les variables à valeur unique peuvent également entraîner des erreurs ou des résultats inattendus.
Vous pouvez détecter les lignes qui ont cette propriété à l'aide de **la fonction unique() NumPy** qui indiquera le nombre de valeurs uniques dans chaque colonne.

Écrire le programme Python qui charge le jeu de données (oil-spill.csv) et résume le nombre de valeurs uniques pour chaque colonne.

B- Supprimer les colonnes contenant une seule valeur

Il faut supprimer les colonnes qui ont une seule valeur unique. C'est facile de supprimer d'un tableau NumPy ou Pandas DataFrame une colone.

Une approche consiste à enregistrer toutes les colonnes qui ont une seule valeur puis à les supprimer du Pandas DataFrame en appelant la fonction drop().

Ecrire cette fonction .

C- Considérez les colonnes qui ont très peu de valeurs : Les variables avec très peu de valeurs numériques peuvent également entraîner des erreurs ou des résultats inattendus. Pour remédier à ce type de problème aussi, Pour faciliter la mise en évidence des colonnes de ce type, vous pouvez calculer le nombre de valeurs uniques pour chaque variable sous forme de pourcentage du nombre total de lignes dans l'ensemble de données.

Écrire le code python suivant montre comment calculer le pourcentage des valeurs uniques.

Nous pouvons mettre à jour l'exemple pour ne résumer que les variables qui ont des valeurs uniques inférieures à 1 % du nombre de lignes.

```
#résumer le pourcentage de valeurs
# uniques pour chaque colonne en utilisant numpy

from numpy import loadtxt
from numpy import unique

# charger le jeu de données

data = loadtxt('oil-spill.csv', delimiter=',')

#résumer le nombre de valeurs uniques dans chaque colonn

for i in range(data.shape[1]):
	num = len(unique(data[:, i]))
	percentage = float(num) / data.shape[0] * 100
	# pour afficher les colonnes inf à 1%
	if percentage < 1:
		print('%d, %d, %.1f%%' % (i, num, percentage))
```

En exécutant l'exemple, nous pouvons voir que 11 des 50 variables ont des variables numériques qui ont des valeurs uniques inférieures à 1 % du nombre de lignes. Cela ne signifie pas que ces lignes et colonnes doivent être supprimées, mais elles nécessitent une attention particulière. Par exemple:

- ✓ Peut-être que les valeurs uniques peuvent être codées en tant que valeurs ordinales ?
- ✓ Peut-être que les valeurs uniques peuvent être encodées en tant que valeurs catégorielles ?
- ✓ Peut-être comparer la compétence du modèle avec chaque variable supprimée de l'ensemble de données ?

```

# supprimer les colonnes où le nombre de
# valeurs uniques représentent moins de 1 % des lignes

from pandas import read_csv

# charger le jeu de données
df = read_csv('oil-spill.csv', header=None)
print(df.shape)

# obtenir le nombre de valeurs uniques pour
#chaque colonne
counts = df.nunique()

# enregistrer les colonnes à supprimer
to_del = [i for i,v in enumerate(counts) if (float(v)/df.shape[0]*100) < 1]
print(to_del)

# laisser tomber la colonne inutiles
df.drop(to_del, axis=1, inplace=True)
print(df.shape)
```