DataMining

TP 1 : prétraitement des données numériques

**Introduction**

Le prétraitement des données pour l’apprentissage automatique consiste à suivre les étapes suivantes :

* Étape 1 : Importation des bibliothèques ;
* Étape 2 : Importation de la dataset en question ;
* Étape 3 : Vérification des valeurs manquantes ;
* Étape 4 : Vérification des valeurs catégorielles ;
* Étape 5 : Fractionnement de l’ensemble de données ;
* Étape 6 : Mise à l’échelle des caractéristiques.

**1. Importation des bibliothèques**

Les bibliothèques fondamentales dont on aura besoin sont les suivantes :

* **Code :**

import numpy as np  
import pandas as pd

**2. Importation de la dataset**

df = pd.read\_csv ('consommation-nourriture.csv')  
df.head()

**3. Vérification des valeurs manquantes**

Nous affichons ci-dessous le nombre de valeurs manquantes pour chaque colonne de l’ensemble de données.

* **Code :**

**df.isnull().sum()**

Nous nous intéresserons dans ce Tp à deux méthodes jugées les plus importantes pour résoudre le problème des valeurs manquantes :

* La première méthode consiste à : soit supprimer une ligne particulière contenant une valeur manquante pour une caractéristique donnée, soit supprimer toute une colonne si elle comporte plus de 75% de valeurs manquantes. Cette méthode est recommandée lorsqu’il y a suffisamment d’échantillons dans la dataset. Il faut s’assurer que la suppression de données n’impactera pas les résultats finaux.
* La deuxième méthode s’applique aux éléments contenant les données numériques où les valeurs manquantes peuvent être remplacées par la moyenne, la médiane ou le mode de l’élément en question. Cette méthode donne un meilleur résultat comparant à la première méthode parce qu’il s’agit d’une approximation statistique qui peut ajouter de la variance à l’ensemble de données. On peut aussi approximer avec l’écart des valeurs voisines si les données sont linéaires.

**Méthode 1 :**

On retourne les dimensions du dataframe ***df*** à l’aide de l’attribut de pandas ***shape***.

* **Code :**

**df.shape**

À l’aide de la fonction ***dropna(),***on supprime toutes les valeurs manquantes de notre ensemble de données, l’option inplace = True permet de modifier directement la dataframe et non pas de créer une copie. Ensuite, en faisant appel aux deux fonctions ***isnull()***et ***sum(),*** on retourne la somme des valeurs manquantes pour chaque colonne et qui est 0 partout !

Finalement, on affiche la dimension de la dataframe qui est différente de la première, car on vient de supprimer les valeurs manquantes de chaque colonne.

* **Code :**

#méthode 1  
df.dropna( inplace = True)  
df.isnull().sum()

**Méthode 2 :**

Dans ce qui suit, on calcule la moyenne de la colonne « Biscuits », on affiche cette colonne pour comparer ses valeurs après avoir modifié la valeur manquante par la moyenne des valeurs de la colonne « Biscuits ».

* **Code :**

#méthode 2  
df['Biscuits'].mean()  
print( df['Biscuits'])  
df['Biscuits'].replace( np.NaN, df ['Biscuits'].mean())

Jusqu’à présent, nous avons vu comment traiter des données numériques, l’étape suivante se focalisera sur les données catégorielles.

**4. Vérification des valeurs catégorielles**

Les modèles d’apprentissage automatique se basent sur des équations mathématiques, alors intuitivement la présence de données catégorielles entrainera un problème car on ne peut garder que des nombres dans les équations. Alors ces données catégorielles doivent être codées en données numériques.

**Exemple 4 :**

Prenons la dataset « iris », le type de l’iris est une donnée catégorielle :

* **Code :**

import numpy as np  
import pandas as pd  
df = pd.read\_csv('iris.csv')  
df.head()

Pour convertir les données de la colonne « flower » en données numériques, on peut faire appel à la classe ***LabelEncoder*()**de la bibliothèque ***preprocessing.***

Après avoir créé une instance de cette classe, on l’ajuste à l’aide de ***fit\_transform()***à la colonne « flower » pour qu’elle revoie celle-ci sous format encodée.

* **Code :**

y = df.iloc[ : , 4: ].values  
print(y)  
from sklearn.preprocessing import LabelEncoder  
Label\_encoder = LabelEncoder()  
y = Label\_encoder.fit\_transform(y .ravel())  
print(y)

Maintenant que les données catégorielles sont encodées, un problème se présente : puisque 0<1 et 1<2, les équations des modèles d’apprentissage penseront que « Iris-setosa » a une valeur moins élevée que « Iris-versicolor » et que « Iris-versicolor » a une valeur moins élevée que « Iris-verginica », alors que ce n’est pas le cas, ce sont trois différentes catégories qu’elles n’ont pas d’ordre relationnel entre eux.

Alors, pour éviter cela, nous utiliserons les ***Dummies Variables***qui sont des variables qui prennent la valeur 0 pour indiquer l’absence d’un effet catégorique pouvant modifier le résultat et 1 pour sa présence.

**Exemple 5 :**

Nous continuons avec l’exemple de la dataset « iris ». Alors dans ce cas au lieu d’avoir une colonne, nous allons avoir trois.

Cette méthode de création des ***dummies variables*** consiste à faire appel à la fonction ***get\_dummies()*** de la bibliothèque Pandas.

* **Code :**

dummy = pd.get\_dummies (df ['flower'])  
dummy

Maintenant, il ne reste plus qu’à concaténer les deux tableaux et de supprimer la colonne « flower »:

* **Code :**

df = pd.concat( [df, dummy ], axis = 1)  
df.drop( ['flower'], axis = 1)

**5. Mise à l’échelle des caractéristiques**

La mise à l’échelle ou la standardisation des caractéristiques est la méthode qui permet de limiter la différence de fourchette des variables pour qu’elles puissent être comparées sur des bases communes.

**Exemple 7 :**

Voici l’exemple d’un dataset, les variables « Salaire » et « Age » n’ont pas la même échelle ce qui engendrera des problèmes dans les modèles d’apprentissage automatique.

* **Code :**

import numpy as np  
import pandas as pd  
df = pd. df = pd.read\_csv('nba.csv')  
df.head()

Si on prend deux valeurs de ces deux colonnes et on utilise la distance euclidienne, la colonne Salaire sera dominée par celle-ci.

**Remarque :**

Dans certains cas, les modèles d’apprentissage automatique ne sont pas basés sur la distance euclidienne, mais la mise à l’échelle des caractéristiques permettra à l’algorithme de converger plus rapidement. C’est le cas du modèle d’arbre de décision.

Dans le code suivant, on fait appel à la classe ***StandardScaler***qui permet de standardiser les variables. On crée une instance de celle-ci puis on l’entraine sur notre ensemble de données grâce à la méthode ***fit\_transform().***

Résultat : toutes les valeurs sont à la même échelle.

* **Code :**

x = df[['Age', 'Salary']]  
from sklearn.preprocessing import StandardScaler  
mise\_a\_echelle = StandardScaler()  
x = mise\_a\_echelle.fit\_transform(x)  
print(x)

Le code suivant permet de transformer le tableau contenant les valeurs standardisées des deux variables Salaire et Age en un dataframe, pour ensuite le concaténer avec les données de la dataset. Puis on supprime les deux anciennes colonnes en question.

* **Code :**

dfx = pd.DataFrame(data = x, columns = ['Age2', 'Salaire2'])  
df = pd.concat( [df, dfx ], axis = 1)  
df = df.drop( ['Age', 'Salaire'], axis = 1)  
print(df)

**7. Exercices**

Appliquez les différentes étapes du prétraitement à la dataset suivante Pokemon:

* **Code :**

import numpy as np  
import pandas as pd  
#import des données  
df = pd.read\_csv ('pokemon.csv')  
#affichage des 5 premiers enregistrements  
df.head ()

On affiche les informations relatives à la dataset en utilisant la fonction suivante :

**df.info ()**

Pour savoir la somme des valeurs manquantes on procède ainsi :

**df.isnull ().sum ()**

Encodage de la colonne Type\_1 en dummies variables.

* **Code :**

df ['type1'].unique ()

dummy = pd.get\_dummies ( df ['type1'])

dummy

Finalement, on standardise les valeurs des colonnes suivantes : Total, HP, Attack, Defense, Height\_m, Weight\_kg.

* **Code :**

x = df[['height\_m', 'hp', 'weight\_kg']]  
from sklearn.preprocessing import StandardScaler  
mise\_a\_echelle = StandardScaler ()  
x = mise\_a\_echelle.fit\_transform(x)  
print(x)