Gestion des données manquantes

Source :

- Massaron Luca, Mueller John Paul, Data Science avec Python pour les nuls, First, 2019, p.141-142.
- doc Pandas(fillna) : https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.fillna.html et dropna() : https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.dropna.html
- Pandas DataFrame fillna() Method : https://www.w3schools.com/python/pandas/ref_df_fillna.asp
- fillna() pour remplacer les valeurs Null dans la trame de données : https://fr.acervolima.com/python-pandas-dataframe-fillna-pour-remplacer-les-valeurs-null-dans-dataframe/
- Nettoyage de données avec Python : détection des valeurs manquantes : https://moncoachdata.com/blog/nettoyage-de-donnees-python/
- Comment remplir les données manquantes à l'aide de Python Pandas : https://www.journaldufreenaute.fr/comment-remplir-les-donnees-manquantes-a-laide-de-python-pandas/

Généralement, on remplace les valeurs manquantes par :

- `mean` : la moyenne
- `median` : la médiane
- `most_frequent` : la valeur qui revient le plus souvent
- 0
- Une valeur choisie.

Trouver les données manquantes

Affichage des valeurs nulles (renvoie True/False)

```
print(df['colonne'].isnull())
```

Nombre total de valeurs manquantes pour chaque caractéristique

```
print(df.isnull().sum())
```

Faire une liste de types de valeurs manquantes

`missing_values = ["n/a", "na", "--"]` # ajouter à la liste chaque nouvelle forme de valeur manquante `df = pd.read_csv("property data.csv", na_values = missing_values)`

Savoir s'il reste des valeurs manquantes (renvoie True/False)

`print(df.isnull().values.any())`

Nombre total de valeurs manquantes

`print(df.isnull().sum().sum())`

`fillna()`

In [ ]:
```python
import pandas as pd

# remplace par 0 dans le dataframe (df)
df.fillna(0)

# avec des valeurs précises
values = {"A": 0, "B": 1, "C": 2, "D": 3}
df.fillna(value=values)

# remplace seulement le premier NaN
df.fillna(value=values, limit=1)
```

Syntaxe

`df.fillna(value, method, axis, inplace, limit, downcast)`

- `values` : Required, Specifies the value to replace the NULL values with. This can also be values for the entire row or column.
  `(Value : int, str, dictionary, series, dataframe)`
- `method` : Optional, default None'. Specifies the method to use when replacing.
  `(Value : 'backfill', bfill', 'pad', 'ffill', None)`
- `axis` : line or column (1 or 0, 'index' or 'columns') - `inplace` : True/False. Optional, default False. If True: the replacing is done on the current DataFrame. If False: returns a copy where the replacing is done. - `limit` : Optional, default None. Specifies the maximum number of NULL values to fill (if method is specified) - `downcast` : Optional, a dictionary of values to fill for specific data types

---

`dropna()`

Supprime les lignes (rows) si au moins 1 élément est manquant.

`df.dropna()`

Supprime les rows où tous les éléments sont manquants. `df.dropna(how='all')`

Conserve uniquement les lignes contenant au moins 2 valeurs autres que NA.
`df.dropna(thresh=2)`

En fonction des colonnes :

`df.dropna(subset=['colonne1', 'colonne3'])`

Supprime une colonne si au moins 1 élément est manquant :

`df.dropna(axis='columns')`

Syntaxe

`dataframe.dropna(axis, how, thresh, subset, inplace)`

Avec Imputer de sklearn

In [ ]:
```
from sklearn.preprocessing import Imputer

s = [[1, 2, 3, np.NaN, 5, 6, None]]

imp = Imputer(missing_values='NaN', strategy='mean', axis=0)

imp.fit([[1, 2, 3, 4, 5, 6, 7]])

x = pd.Series(imp.transform(s).tolist()[0])

print(x)
```

'imp' est un objet de type Imputer permettant d'insérer d'autres valeurs à leur place.

- `missing_values` : indique quel symbole est recherché
- `axis` = à 0 pour les colonnes, à 1 pour les lignes
- `strategy` : décide par quoi remplacer les valeurs manquantes

  On fournit des statistiques à l'Imputer lors du `fit()`, on applique avec `transform()` sur 's' puis on affiche le résultat sous forme d'une série. Pour la créer, on obtient d'abord une liste (`tolist()`).

Méthode `replace()`

`df['colonne'].replace([numpy.nan], df['colonne'].mean(), inplace=True)`

Remplacer sur une position

`df.loc[num_ligne,'nom_colonne'] = nouvelle_valeur`