Python pour les data scientistes

Chapitre 5 : Le package pandas

1. Introduction

Pandas est une bibliothèque Python utilisée pour travailler avec des ensembles de données. Elle possède des fonctions d'analyse, de nettoyage, d'exploration et de manipulation des données.

Le nom "Pandas", qui fait référence à la fois à "Panel Data" et à "Python Data Analysis", a été créé par Wes McKinney en 2008.

Pandas vous donne des réponses sur les données. Par exemple :

- Y a-t-il une corrélation entre deux ou plusieurs colonnes ?
- Quelle est la valeur moyenne ?
- La valeur maximale ?
- La valeur minimale ?

Pandas est également capable de supprimer les lignes qui ne sont pas pertinentes ou qui contiennent des valeurs erronées, comme des valeurs vides ou NULL. C'est ce qu'on appelle le nettoyage des données.

2. Installation de Pandas

Si Python et PIP sont déjà installés sur votre système, l'installation de Pandas est très simple.

Installez-le en utilisant cette commande : **pip install pandas**

Si cette commande échoue, utilisez une distribution python dans laquelle Pandas est déjà installé, comme Anaconda, Spyder, etc.

3. Importer Pandas

Une fois Pandas installé, importez-le dans vos applications en ajoutant le mot-clé `import` :

```
import pandas
```

Maintenant, Pandas est importé et prêt à être utilisé.

Exemple

```
import pandas
mydataset = {'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}
myvar = pandas.DataFrame(mydataset)
print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

Pandas est généralement importé sous l'alias pd.

Alias : En Python, les alias sont un nom alternatif pour se référer à la même chose.

Créez un alias avec le mot-clé as lors de l'importation :

```
import pandas as pd
```

Maintenant, le paquet Pandas peut être désigné par pd au lieu de pandas.

Exemple

```
import pandas as pd
mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}
myvar = pd.DataFrame(mydataset)
print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

4. Les séries de Pandas

Une série Pandas est comme une colonne dans un tableau. Il s'agit d'un tableau unidimensionnel contenant des données de n'importe quel type.

Exemple

Créez une série Pandas simple à partir d'une liste :

```
import pandas as pd
liste = [1, 0, -2, 3]
myvar = pd.Series(liste)
print(myvar)
```

```
0    1
1    0
2   -2
3    3
dtype: int64
```

Si rien d'autre n'est spécifié, les valeurs sont étiquetées avec leur numéro d'index. La première valeur avec l'indice 0, la deuxième valeur à l'indice 1, etc.

Cette étiquette peut être utilisée pour accéder à une valeur spécifiée. Par exemple, print(myvar[0]) Retourne la première valeur de la série.

Avec l'argument index, vous pouvez nommer vos propres étiquettes.

```
import pandas as pd
liste = [1, 0, -2, 3]
```

```
myvar = pd.Series(liste, index = ["x", "y", "z","h"])
print(myvar)
```

```
x    1
y    0
z   -2
h    3
dtype: int64
```

Lorsque vous créez des étiquettes, vous pouvez accéder à un élément en vous référant à l'étiquette. Par exemple, `print(myvar["y"])` revoie la valeur de y qui est 0.

Vous pouvez également utiliser un objet clé/valeur, comme un dictionnaire, lors de la création d'une série.

Exemple

Créer une série simple à partir d'un dictionnaire, cet exemple illustre les ventes d'une enreprise par semestre.

```
import pandas as pd
ventes = {"Trimestre1": 500, "Trimestre2": 280, "Trimestre3": 400, "Trimestre4": 390}
mesventes = pd.Series(ventes)
print(mesventes)
```

```
Trimestre1    500
Trimestre2    280
Trimestre3    400
Trimestre4    390
dtype: int64
```

Les clés du dictionnaire deviennent les étiquettes.

5. Les DataFrames

Un DataFrame Pandas est une structure de données à deux dimensions, comme un tableau à deux dimensions ou un tableau avec des lignes et des colonnes.

Exemple

L'exemple suivant crée un dataframe à partir de deux séries.

```
import pandas as pd
donnees_ventes = {
  "ventes": [420, 380, 390],
  "mois": ["janvier", "février", "mars"]
}
mesventes = pd.DataFrame(donnees_ventes)
print(mesventes)
```

```
   ventes     mois
0     420  janvier
1     380  février
2     390     mars
```

La librairie pandas utilise l'attribut "**loc**" pour retourner une ou plusieurs lignes spécifiées, par exemple : `print(mesventes.loc[0])` réfère à l'index de la ligne 0. En voilà le résultat :

```
ventes        420
mois      janvier
Name: 0, dtype: object
```

On peut utiliser aussi une liste d'index : `print(mesventes.loc[[0, 1]])`

```
   ventes     mois
0     420  janvier
1     380  février
```

Remarque : lorsque vous utilisez [], le résultat est un DataFrame Pandas.

Avec l'argument index, vous pouvez nommer vos propres index.

Exemple

Ajouter une liste de noms pour donner un nom à chaque ligne :

```
mesventes = pd.DataFrame(donnees_ventes, index =
["tranche1", "tranche2", "tranche3"])
```

```
          ventes     mois
tranche1     420  janvier
tranche2     380  février
tranche2     390     mars
```

On peut aussi utiliser l'index nommé dans l'attribut "**loc**" pour retourner la ou les lignes spécifiées. Par exemple : `print(mesventes.loc["tranche1"])`

Voilà le résultat :

```
ventes        420
mois      janvier
Name: tranche1, dtype: object
```

Si les données sont stockées dans un fichier, Pandas peut aussi les charger dans un DataFrame.

Exemple

L'exemple suivant permet de charger un fichier CSV dans un DataFrame. Les fichiers CSV contiennent du texte brut et sont un format bien connu qui peut être lu par tout le monde, y compris par Pandas. Le fichier peut exister dans le dossier local, dans un autre dossier, pour cela, il faut ajouter son chemin d'accès, sinon, sur internet, il suffit d'ajouter sinon URI.

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df)
```

```
   Province  State  Country  Region
0        12     25       62       0
1         1      0       16      12
2        14     15       12       0
```

Si le DataFrame contient un grand nombre de lignes, Pandas par défaut ne renvoie que les 5 premières lignes et les 5 dernières. Utilisez to_string() pour afficher le DataFrame entier: `print(df.to_string())`.

Pour lire les fichiers json, utilisez : `df = pd.read_json('data.json').`

L'une des méthodes les plus utilisées pour obtenir un aperçu rapide du DataFrame est la méthode head().

La méthode head() renvoie les en-têtes et un nombre spécifié de lignes, en commençant par le haut. `head(i)` affiche les premières IIème lignes. Si le nombre de lignes n'est pas spécifié, la méthode head() renvoie les 5 premières lignes.

Exemple

Pour afficher les 10 premières lignes d'un DataFrame :

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.head(10))
```

```
   Province  State  Country  Region
0        20     14       15    13.0
1         0      1        2    45.0
2       100    200       36    98.0
3       120     45       87    99.0
4        45     65       30     0.0
5         1      2        3     5.0
6         5      8        9     7.0
7         7      8        9    10.0
8        10     23       45    48.0
9       100    200       36    98.0
```

La méthode tail() renvoie les en-têtes et un nombre spécifié de lignes, en commençant par le bas. Pour afficher les 5 dernières lignes du DataFrame : `print(df.tail())`

```
    Province  State  Country  Region
18         7      8        9    10.0
19        45     65       30     0.0
20         1      2        3     5.0
21         5      8        9     7.0
22         7      8        9    10.0
```

Comme pour head, la méthode `tail(i)` affiche les dernières IIème lignes.

L'objet DataFrame possède une méthode appelée `info()`, qui vous donne plus d'informations sur l'ensemble de données: `print(df.info())`.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23 entries, 0 to 22
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Province  23 non-null     int64
 1   State     23 non-null     int64
 2   Country   23 non-null     int64
 3   Region    22 non-null     float64
dtypes: float64(1), int64(3)
memory usage: 864.0 bytes
None
```

6. Nettoyage de données

Le nettoyage des données consiste à corriger les données erronées dans votre ensemble de données.

Les mauvaises données peuvent être :

- Des cellules vides
- Des données au mauvais format

- Données erronées
- Des doublons

Analysons l'exemple suivant :

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	2020/12/26	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3
31	60	'2020/12/31'	92	115	243.0

- Le jeu de données contient des cellules vides ("Date" à la ligne 22, et "Calories" aux lignes 18 et 28).
- Le jeu de données contient un format incorrect ("Date" à la ligne 26).
- Le jeu de données contient des données erronées ("Durée" à la ligne 7).
- L'ensemble de données contient des doublons (lignes 11 et 12).

Les cellules vides peuvent potentiellement vous donner un résultat erroné lorsque vous analysez des données. Une façon de traiter les cellules vides est de supprimer les lignes qui contiennent des cellules vides.

Cela ne pose généralement pas de problème, car les ensembles de données peuvent être très volumineux et la suppression de quelques lignes n'aura pas un impact important sur le résultat.

Exemple

Ce code renvoie un nouveau cadre de données sans cellules vides :

```
import pandas as pd
df = pd.read_csv('data.csv')
```

```
new_df = df.dropna()
print(new_df.to_string())
```

Remarque : par défaut, la méthode dropna() renvoie un nouveau DataFrame, et ne modifie pas l'original.

Si vous souhaitez modifier le DataFrame d'origine, utilisez l'argument inplace = True :

```
df.dropna(inplace = True)
```

Une autre façon de traiter les cellules vides consiste à insérer une nouvelle valeur à la place.

De cette façon, vous ne devez pas supprimer des lignes entières à cause de quelques cellules vides.

La méthode `fillna()` nous permet de remplacer les cellules vides par une valeur.

Exemple

Le code suivant permet de remplacer les valeurs nulles par le nombre 2 :

```
import pandas as pd
df = pd.read_csv('data.csv')
df.fillna(2, inplace = True)
```

Pour remplacer uniquement les valeurs vides d'une colonne, il faut indiquer le nom de la colonne du DataFrame.

Exemple

L'exemple permet de remplacer les valeurs nulles de la colonne "Calories" avec la valeur 2:

```
import pandas as pd
df = pd.read_csv('data.csv')
df["Calories"].fillna(3, inplace = True)
```

Remplacer en utilisant la moyenne, la médiane ou le mode.

Il est possible de remplacer les cellules vides par des valeurs calculées comme la valeur moyenne, médiane ou mode de la colonne.

Pandas utilise les méthodes `mean()`, `median()` et `mode()` pour calculer les valeurs respectives d'une colonne spécifiée.

- `mean()` : représente la moyenne des valeurs d'une colonne, c'est la somme de toutes les valeurs divisée par le nombre de valeurs).
- `median()` : représente la valeur du milieu, après avoir trié toutes les valeurs par ordre croissant.
- `mode()` : représente la valeur qui apparaît le plus fréquemment.

Exemple

Ce code permet de calculer la moyenne des valeurs d'une colonne, et remplacez toutes les valeurs vides par celle-ci :

```
import pandas as pd
df = pd.read_csv('data.csv')
moyenne = df["Calories"].mean()
df["Calories"].fillna(x, inplace = True)
```

Exemple

Ce code permet de calculer la médiane des valeurs d'une colonne, et remplacez toutes les valeurs vides par celle-ci :

```
import pandas as pd
df = pd.read_csv('data.csv')
x = df["Calories"].median()
df["Calories"].fillna(x, inplace = True)
```

Exemple

Ce code permet de calculer le mode des valeurs d'une colonne, et remplacez toutes les valeurs vides par celle-ci :

```
import pandas as pd
df = pd.read_csv('data.csv')
x = df["Calories"].mode()[0]
df["Calories"].fillna(x, inplace = True)
```

7. Traitement des données de format incorrect

Les cellules contenant des données de mauvais format peuvent rendre difficile, voire impossible, l'analyse des données.

Pour y remédier, vous avez deux possibilités : supprimer les lignes ou convertir toutes les cellules des colonnes au même format.

Par exemple, une colonne date doit contenir des valeurs de type date, si une valeur de cette colonne ne relève pas du type date, pandas dispose d'une méthode utile pour convertir cette valeur en type date : `to_datetime()`.

Le code suivant illustre l'utilisation de la méthode `to_datetime()`.

```
import pandas as pd
df = pd.read_csv('data.csv')
df['Date'] = pd.to_datetime(df['Date'])
print(df.to_string())
```

8. Correction des données erronées:

Les "données erronées" ne sont pas nécessairement des "cellules vides" ou un "format incorrect", elles peuvent simplement être erronées, par exemple remplir la valeur "20.22" à la place de "2022".

Parfois, si nous avons une idée préalable du domaine d'étude, nous pouvons repérer les données erronées. Une façon de corriger les valeurs erronées est de les remplacer par quelque chose d'autre.

Par exemple, une colonne des notes d'étudiants (la note varie entre 0 et 20), si on trouve une valeur qui n'appartient pas à cet intervalle, nous pouvons la remplacer.

Si par exemple cette valeur se trouve à la ligne 7, nous corriger la note :
`df.loc[7, 'note'] = i` et i représente une valeur arbitraire.

Pour les petits jeux de données, nous pouvons remplacer les données erronées une par une, seulement pour les grands datasets, c'est plus compliqué et il existe d'autres possibilités.

Pour remplacer des données erronées des grands datasets, vous pouvez créer des règles, par exemple en fixant des limites pour les valeurs correctes et en remplaçant toutes les valeurs qui sont en dehors de ces limites.
Par exemple pour des notes, si la note dépasse 20, on la remplace par la valeur 10.

```
for x in df.index:
  if df.loc[x, "note"] > 120:
    df.loc[x, "note"] = 120
```

9. Suppression des doublons :
Pour découvrir les doublons, nous pouvons utiliser la méthode duplicated(). La méthode `duplicated()` renvoie une valeur booléenne pour chaque ligne :

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.duplicated())
```

Pour supprimer les doublons : `df.drop_duplicates(inplace = True)`

10. Corrélations des données
Un aspect important du module Pandas est la méthode `corr()`. La méthode `corr()` évalue la relation entre chaque colonne du jeu de données: `df.corr()`. Cette méthode ignore les colonnes non numériques

Voilà le résultat de l'exécution de cette méthode sur le dataset :
Le résultat de la méthode `corr()` est un tableau contenant un grand nombre de chiffres qui représentent la qualité de la relation entre deux colonnes. Le nombre varie de -1 à 1.

1 signifie qu'il existe une relation de 1 à 1 (une corrélation parfaite), et pour cet ensemble de données, chaque fois qu'une valeur augmente dans la première colonne, l'autre augmente également.

0,9 est également une bonne relation, et si vous augmentez une valeur, l'autre augmentera probablement aussi.

-0,9 est une relation tout aussi bonne que 0,9, mais si vous augmentez une valeur, l'autre diminuera probablement.

0,2 signifie qu'il ne s'agit pas d'une bonne relation, c'est-à-dire que si une valeur augmente, cela ne signifie pas que l'autre augmentera aussi.

11. Le plotting
Pandas utilise la méthode `plot()` pour créer des diagrammes. Nous pouvons utiliser Pyplot, un sous-module de la bibliothèque `Matplotlib` pour visualiser le diagramme à l'écran.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('data.csv')
df.plot()
plt.show()
```

Nous verrons la librairie avec plus de detail.