

# Prétraitement et analyse de données

**Ousman Hamit Hassani**

Dakar Institute of Technology



Master 1 - Intelligence artificielle

February 17, 2023

- 1 Introduction
- 2 Installation d'environnements de travail
- 3 Nettoyage de données(data cleaning)
- 4 Transformation de données(feature scaling)
- 5 Réduction de données
- 6 Visualisation de données

# Objectifs

## Prétraitement de données


✍ Ce cours vous permettra d'acquérir les compétences requises pour nettoyer, transformer et mettre au bon format des données afin d'assurer leur cohérence et fiabilité pour des analyses futures.

## Visualisation de données


📊 Vous pourrez également créer des résumés statistiques pour comprendre les tendances et les relations entre variables, présenter des résultats de manière synthétique et pertinente à l'aide de graphiques et de tableaux etc.

# Objectifs

## Prétraitement de données

 Ce cours vous permettra d'acquérir les compétences requises pour nettoyer, transformer et mettre au bon format des données afin d'assurer leur cohérence et fiabilité pour des analyses futures.

## Visualisation de données

 Vous pourrez également créer des résumés statistiques pour comprendre les tendances et les relations entre variables, présenter des résultats de manière synthétique et pertinente à l'aide de graphiques et de tableaux etc.

## Environnements virtuels

ANACONDA DISTRIBUTION



ANACONDA®

CONDA®

## Installation et création d'environnements

- ❶ <https://www.anaconda.com/products/distribution>
- ❷ `sudo apt-get update && sudo apt-get upgrade -y`
- ❸ `bash Anaconda3-2022.10-Linux-x86_64.sh`
- ❹ `conda create --name ml-env3.9 python3.9`
- ❺ `conda activate ml-env3.9 / conda deactivate ml-env3.9`
- ❻ `conda env list / conda list`
- ❼ `conda install PACKAGE_NAME / conda remove PACKAGE_NAME`
- ❽ `conda update --all`
- ❾ <https://urlz.fr/kErb>

## Installation et création d'environnements

- ❶ <https://www.anaconda.com/products/distribution>
- ❷ **`sudo apt-get update && sudo apt-get upgrade -y`**
- ❸ `bash Anaconda3-2022.10-Linux-x86_64.sh`
- ❹ `conda create --name ml-env3.9 python3.9`
- ❺ `conda activate ml-env3.9 / conda deactivate ml-env3.9`
- ❻ `conda env list / conda list`
- ❼ `conda install PACKAGE_NAME / conda remove PACKAGE_NAME`
- ❽ `conda update --all`
- ❾ <https://urlz.fr/kErb>

## Installation et création d'environnements

- ❶ <https://www.anaconda.com/products/distribution>
- ❷ `sudo apt-get update && sudo apt-get upgrade -y`
- ❸ `bash Anaconda3-2022.10-Linux-x86_64.sh`
- ❹ `conda create --name ml-env3.9 python3.9`
- ❺ `conda activate ml-env3.9 / conda deactivate ml-env3.9`
- ❻ `conda env list / conda list`
- ❼ `conda install PACKAGE_NAME / conda remove PACKAGE_NAME`
- ❽ `conda update --all`
- ❾ <https://urlz.fr/kErb>



## Installation et création d'environnements

- ❶ <https://www.anaconda.com/products/distribution>
- ❷ `sudo apt-get update && sudo apt-get upgrade -y`
- ❸ `bash Anaconda3-2022.10-Linux-x86_64.sh`
- ❹ **`conda create --name ml-env3.9 python3.9`**
- ❺ `conda activate ml-env3.9 / conda deactivate ml-env3.9`
- ❻ `conda env list / conda list`
- ❼ `conda install PACKAGE_NAME / conda remove PACKAGE_NAME`
- ❽ `conda update --all`
- ❾ <https://urlz.fr/kErb>

## Installation et création d'environnements

- ❶ <https://www.anaconda.com/products/distribution>
- ❷ `sudo apt-get update && sudo apt-get upgrade -y`
- ❸ `bash Anaconda3-2022.10-Linux-x86_64.sh`
- ❹ `conda create --name ml-env3.9 python3.9`
- ❺ **`conda activate ml-env3.9 / conda deactivate ml-env3.9`**
- ❻ `conda env list / conda list`
- ❼ `conda install PACKAGE_NAME / conda remove PACKAGE_NAME`
- ❽ `conda update --all`
- ❾ <https://urlz.fr/kErb>

## Installation et création d'environnements

- ❶ <https://www.anaconda.com/products/distribution>
- ❷ `sudo apt-get update && sudo apt-get upgrade -y`
- ❸ `bash Anaconda3-2022.10-Linux-x86_64.sh`
- ❹ `conda create --name ml-env3.9 python3.9`
- ❺ `conda activate ml-env3.9 / conda deactivate ml-env3.9`
- ❻ **`conda env list / conda list`**
- ❼ `conda install PACKAGE_NAME / conda remove PACKAGE_NAME`
- ❽ `conda update --all`
- ❾ <https://urlz.fr/kErb>

## Installation et création d'environnements

- ❶ <https://www.anaconda.com/products/distribution>
- ❷ `sudo apt-get update && sudo apt-get upgrade -y`
- ❸ `bash Anaconda3-2022.10-Linux-x86_64.sh`
- ❹ `conda create --name ml-env3.9 python3.9`
- ❺ `conda activate ml-env3.9 / conda deactivate ml-env3.9`
- ❻ `conda env list / conda list`
- ❼ **`conda install PACKAGE_NAME / conda remove PACKAGE_NAME`**
- ❽ `conda update --all`
- ❾ <https://urlz.fr/kErb>

## Installation et création d'environnements

- ❶ <https://www.anaconda.com/products/distribution>
- ❷ `sudo apt-get update && sudo apt-get upgrade -y`
- ❸ `bash Anaconda3-2022.10-Linux-x86_64.sh`
- ❹ `conda create --name ml-env3.9 python3.9`
- ❺ `conda activate ml-env3.9 / conda deactivate ml-env3.9`
- ❻ `conda env list / conda list`
- ❼ `conda install PACKAGE_NAME / conda remove PACKAGE_NAME`
- ❽ `conda update --all`
- ❾ <https://urlz.fr/kErb>

## Installation et création d'environnements

- ❶ <https://www.anaconda.com/products/distribution>
- ❷ `sudo apt-get update && sudo apt-get upgrade -y`
- ❸ `bash Anaconda3-2022.10-Linux-x86_64.sh`
- ❹ `conda create --name ml-env3.9 python3.9`
- ❺ `conda activate ml-env3.9 / conda deactivate ml-env3.9`
- ❻ `conda env list / conda list`
- ❼ `conda install PACKAGE_NAME / conda remove PACKAGE_NAME`
- ❽ `conda update --all`
- ❾ <https://urlz.fr/kErb>

## Alternative à anaconda3

- ❶ `sudo apt install python3-virtualenv`
- ❷ `python3 -m venv .venv`
- ❸ `source .venv/bin/activate`
- ❹ `pip list / pip3 list`
- ❺ `pip install jupyter pandas numpy seaborn matplotlib`
- ❻ `pip uninstall PACKAGE_NAME`
- ❼ `jupyter notebook`

## Alternative à anaconda3

- ❶ `sudo apt install python3-virtualenv`
- ❷ `python3 -m venv .venv`
- ❸ `source .venv/bin/activate`
- ❹ `pip list / pip3 list`
- ❺ `pip install jupyter pandas numpy seaborn matplotlib`
- ❻ `pip uninstall PACKAGE_NAME`
- ❼ `jupyter notebook`



## Alternative à anaconda3

- 1 `sudo apt install python3-virtualenv`
- 2 `python3 -m venv .venv`
- 3 `source .venv/bin/activate`
- 4 `pip list / pip3 list`
- 5 `pip install jupyter pandas numpy seaborn matplotlib`
- 6 `pip uninstall PACKAGE_NAME`
- 7 `jupyter notebook`

## Alternative à anaconda3

- ❶ `sudo apt install python3-virtualenv`
- ❷ `python3 -m venv .venv`
- ❸ `source .venv/bin/activate`
- ❹ `pip list / pip3 list`
- ❺ `pip install jupyter pandas numpy seaborn matplotlib`
- ❻ `pip uninstall PACKAGE_NAME`
- ❼ `jupyter notebook`

## Alternative à anaconda3

- ❶ `sudo apt install python3-virtualenv`
- ❷ `python3 -m venv .venv`
- ❸ `source .venv/bin/activate`
- ❹ `pip list / pip3 list`
- ❺ `pip install jupyter pandas numpy seaborn matplotlib`
- ❻ `pip uninstall PACKAGE_NAME`
- ❼ `jupyter notebook`

## Alternative à anaconda3

- ❶ `sudo apt install python3-virtualenv`
- ❷ `python3 -m venv .venv`
- ❸ `source .venv/bin/activate`
- ❹ `pip list / pip3 list`
- ❺ `pip install jupyter pandas numpy seaborn matplotlib`
- ❻ `pip uninstall PACKAGE_NAME`
- ❼ `jupyter notebook`

## Alternative à anaconda3

- ❶ `sudo apt install python3-virtualenv`
- ❷ `python3 -m venv .venv`
- ❸ `source .venv/bin/activate`
- ❹ `pip list / pip3 list`
- ❺ `pip install jupyter pandas numpy seaborn matplotlib`
- ❻ `pip uninstall PACKAGE_NAME`
- ❼ `jupyter notebook`



## Étapes à suivre

- 👉 Description et compréhension de la problématique
- 👉 **Identification et suppression de données dupliquées**
- 👉 Correction de données erronées
- 👉 Traiter les valeurs manquantes
- 👉 Traiter les valeurs aberrantes
- 👉 Encodage des variables(qualitatives)
- 👉 Normaliser ou standardiser les variables(numériques)
- 👉 Sélectionner les caractéristiques les plus importantes
- 👉 Conversion des types
- 👉 etc.





## Étapes à suivre

- 👉 Description et compréhension de la problématique
- 👉 Identification et suppression de données dupliquées
- 👉 Correction de données erronées
- 👉 **Traiter les valeurs manquantes**
- 👉 Traiter les valeurs aberrantes
- 👉 Encodage des variables(qualitatives)
- 👉 Normaliser ou standardiser les variables(numériques)
- 👉 Sélectionner les caractéristiques les plus importantes
- 👉 Conversion des types
- 👉 etc.





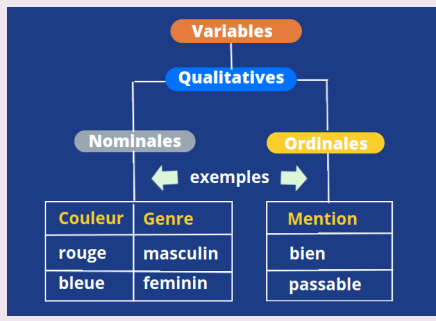




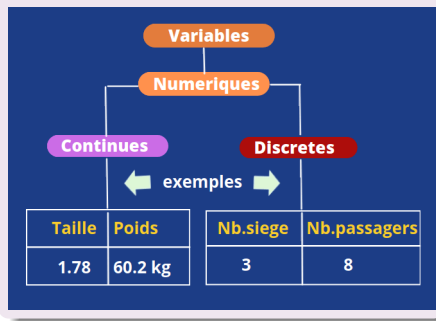


- 👉 Description et compréhension de la problématique
- 👉 Identification et suppression de données dupliquées
- 👉 Correction de données erronées
- 👉 Traiter les valeurs manquantes
- 👉 Traiter les valeurs aberrantes
- 👉 Encodage des variables(qualitatives)
- 👉 Normaliser ou standardiser les variables(numériques)
- 👉 Sélectionner les caractéristiques les plus importantes
- 👉 Conversion des types
- 👉 etc.

## Variables qualitatives



## Variables quantitatives





## Exemple de dataset à nettoyer

<b>Nom</b>	<b>Taille</b>	<b>Poids</b>	<b>Age</b>	<b>Performance</b>
Ibrahim	1m78	70 kg	28 ans	Bonne
Ali	1.80	65.5	27	Insuffisant
Diop	1.80cm	65.5 kg	28.5	Moyenne
moussa	0.65	65.5	32 ANS	Faible
Haroun	1,65	63.0	29,7	Moyen
James	3.65	200 kg	33	faible

## Google colab

👉 <https://colab.research.google.com/#create=true>

## Explorer pour comprendre(méthodes et attributs)

```
■ df = pd.read_csv("data.csv")
```

## Explorer pour comprendre(méthodes et attributs)

- **df = pd.read\_csv("data.csv")**
- **df.head(), df.tail(), df.shape, df.shape[0] et df.shape[1])**
- df.columns, df.info(), df.dtypes(), df.describe()
- df.drop(columns = ["col1", "col2", ... , "col\_n"], inplace = True)
- df.rename(columns = {"ColName": "NewName"}, inplace = True)
- df.insert(position, colname, values)
- pd.set\_option("display.max\_rows", None)
- pd.set\_option("display.max\_columns", None)

 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)



## Explorer pour comprendre(méthodes et attributs)

- `df = pd.read_csv("data.csv")`
- `df.head()`, `df.tail()`, `df.shape`, `df.shape[0]` et `df.shape[1]`
- `df.columns`, `df.info()`, `df.dtypes()`, `df.describe()`
- `df.drop(columns = ["col1", "col2", ... , "col_n"], inplace = True)`
- `df.rename(columns = {"ColName": "NewName"}, inplace = True)`
- `df.insert(position, colname, values)`
- `pd.set_option("display.max_rows", None)`
- `pd.set_option("display.max_columns", None)`

 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)

## Explorer pour comprendre(méthodes et attributs)

- `df = pd.read_csv("data.csv")`
- `df.head()`, `df.tail()`, `df.shape`, `df.shape[0]` et `df.shape[1]`
- `df.columns`, `df.info()`, `df.dtypes()`, `df.describe()`
- `df.drop(columns = ["col1", "col2", ... , "col_n"], inplace = True)`
- `df.rename(columns = {"ColName": "NewName"}, inplace = True)`
- `df.insert(position, colname, values)`
- `pd.set_option("display.max_rows", None)`
- `pd.set_option("display.max_columns", None)`

 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)

## Explorer pour comprendre(méthodes et attributs)

- **df = pd.read\_csv("data.csv")**
- **df.head(), df.tail(), df.shape, df.shape[0] et df.shape[1]**
- **df.columns, df.info(), df.dtypes(), df.describe()**
- **df.drop(columns = ["col1", "col2", ... , "col\_n"], inplace = True)**
- **df.rename(columns = {"ColName": "NewName"}, inplace = True)**
- **df.insert(position, colname, values)**
- `pd.set_option("display.max_rows", None)`
- `pd.set_option("display.max_columns", None)`

 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)



## Explorer pour comprendre(méthodes et attributs)

- `df = pd.read_csv("data.csv")`
- `df.head()`, `df.tail()`, `df.shape`, `df.shape[0]` et `df.shape[1]`
- `df.columns`, `df.info()`, `df.dtypes()`, `df.describe()`
- `df.drop(columns = ["col1", "col2", ... , "col_n"], inplace = True)`
- `df.rename(columns = {"ColName": "NewName"}, inplace = True)`
- `df.insert(position, colname, values)`
- `pd.set_option("display.max_rows", None)`
- `pd.set_option("display.max_columns", None)`

 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)

## Explorer pour comprendre(méthodes et attributs)

- `df = pd.read_csv("data.csv")`
- `df.head()`, `df.tail()`, `df.shape`, `df.shape[0]` et `df.shape[1]`
- `df.columns`, `df.info()`, `df.dtypes()`, `df.describe()`
- `df.drop(columns = ["col1", "col2", ... , "col_n"], inplace = True)`
- `df.rename(columns = {"ColName": "NewName"}, inplace = True)`
- `df.insert(position, colname, values)`
- `pd.set_option("display.max_rows", None)`
- `pd.set_option("display.max_columns", None)`

 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)

- `df = pd.read_csv("data.csv")`
- `df.head()`, `df.tail()`, `df.shape`, `df.shape[0]` et `df.shape[1]`
- `df.columns`, `df.info()`, `df.dtypes()`, `df.describe()`
- `df.drop(columns = ["col1", "col2", ... , "col_n"], inplace = True)`
- `df.rename(columns = {"ColName": "NewName"}, inplace = True)`
- `df.insert(position, colname, values)`
- `pd.set_option("display.max_rows", None)`
- `pd.set_option("display.max_columns", None)`

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

## Identifier et supprimer les données dupliquées

### ■ **df.loc[df.duplicated()]**

- `df.drop_duplicates(keep = {'first', 'last', False}, inplace = True)`
- `df.drop_duplicates(subset = ['cols'], keep = ... , inplace = True)`

🌐 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)

## Identifier et supprimer les données dupliquées

- **`df.loc[df.duplicated()]`**
- **`df.drop_duplicates(keep = {'first', 'last', False}, inplace = True)`**
- `df.drop_duplicates(subset = ['cols'], keep = ... , inplace = True)`

🌐 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)

## Identifier et supprimer les données dupliquées

- **df.loc[df.duplicated()]**
- **df.drop\_duplicates(keep = {'first', 'last', False}, inplace = True)**
- **df.drop\_duplicates(subset = ['cols'], keep = ... , inplace = True)**

🌐 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)

## Identifier et supprimer les données dupliquées

- **`df.loc[df.duplicated()]`**
- **`df.drop_duplicates(keep = {'first', 'last', False}, inplace = True)`**
- **`df.drop_duplicates(subset = ['cols'], keep = ... , inplace = True)`**

👉 [https://pandas.pydata.org/docs/getting\\_started/index.html](https://pandas.pydata.org/docs/getting_started/index.html)

## Explorer les données manquantes

- **`df.isnull().sum(), df.isna().sum(), df.isnull().sum().sum()`**
- `round(df.isna().sum() / df.shape[0], 2)*100`
- `df.style.highlight_null(color = "red")`
- `plt.figure(figsize = (10, 4))`
- `sns.heatmap(df.isnull(), cmap= "viridis", cbar = False);`



## Explorer les données manquantes

- **`df.isnull().sum(), df.isna().sum(), df.isnull().sum().sum()`**
- **`round(df.isna().sum() / df.shape[0], 2)*100`**
- `df.style.highlight_null(color = "red")`
- `plt.figure(figsize = (10, 4))`
- `sns.heatmap(df.isnull(), cmap= "viridis", cbar = False);`

## Explorer les données manquantes

- **`df.isnull().sum(), df.isna().sum(), df.isnull().sum().sum()`**
- **`round(df.isna().sum() / df.shape[0], 2)*100`**
- **`df.style.highlight_null(color = "red")`**
- `plt.figure(figsize = (10, 4))`
- `sns.heatmap(df.isnull(), cmap= "viridis", cbar = False);`

## Explorer les données manquantes

- `df.isnull().sum(), df.isna().sum(), df.isnull().sum().sum()`
- `round(df.isna().sum() / df.shape[0], 2)*100`
- `df.style.highlight_null(color = "red")`
- `plt.figure(figsize = (10, 4))`
- `sns.heatmap(df.isnull(), cmap= "viridis", cbar = False);`

## Explorer les données manquantes

- `df.isnull().sum(), df.isna().sum(), df.isnull().sum().sum()`
- `round(df.isna().sum() / df.shape[0], 2)*100`
- `df.style.highlight_null(color = "red")`
- `plt.figure(figsize = (10, 4))`
- `sns.heatmap(df.isnull(), cmap= "viridis", cbar = False);`

## Supprimer ou remplacer les NaNs

### 👉 Les supprimer ou ne rien faire ?

- 👉 Si oui, supprimer les lignes ou colonnes ?
- 👉 Les remplacer par (**mean, median, most\_frequent, constant, ffill, bfill**)
- 👉 Pour quelle conclusion ?



## Supprimer ou remplacer les NaNs

- 👉 Les supprimer ou ne rien faire ?
- 👉 Si oui, supprimer les lignes ou colonnes ?
- 👉 Les remplacer par (**mean, median, most\_frequent, constant, ffill, bfill**)
- 👉 Pour quelle conclusion ?

## Supprimer ou remplacer les NaNs

- 👉 Les supprimer ou ne rien faire ?
- 👉 Si oui, supprimer les lignes ou colonnes ?
- 👉 Les remplacer par (**mean**, **median**, **most\_frequent**, **constant**, **ffill**, **bfill**)
- 👉 Pour quelle conclusion ?

















## Remplacer les **NaN** par la **MEDIANE** de la colonne considérée

On peut utiliser la **MEDIANE** si:

- 👉 Les données sont fortement non-gaussiennes
  - 👉 Les données comportent de nombreux outliers
  - 👉 Le total de NaN est faible par rapport à la taille de la colonne considérée
- ```
df["colname"].fillna(df["colname"].median(), inplace = True)  
df.fillna(df.median(), inplace = True)
```
- ✓ Dans le cas contraire, il est recommandé d'utiliser d'autres méthodes pour traiter les valeurs manquantes.



## Remplacer les **NaN** par la **MEDIANE** de la colonne considérée

On peut utiliser la **MEDIANE** si:

- 👉 Les données sont fortement non-gaussiennes
- 👉 Les données comportent de nombreux outliers
- 👉 Le total de **NaN** est faible par rapport à la taille de la colonne considérée

```
df["colname"].fillna(df["colname"].median(), inplace = True)
```

```
df.fillna(df.median(), inplace = True)
```

- ✓ Dans le cas contraire, il est recommandé d'utiliser d'autres méthodes pour traiter les valeurs manquantes.





## Remplacer les **NaN** par la **MEDIANE** de la colonne considérée

On peut utiliser la **MEDIANE** si:

- 👉 Les données sont fortement non-gaussiennes
- 👉 Les données comportent de nombreux outliers
- 👉 Le total de **NaN** est faible par rapport à la taille de la colonne considérée

- `df["colname"].fillna(df["colname"].median(), inplace = True)`
- `df.fillna(df.median(), inplace = True)`

✓ Dans le cas contraire, il est recommandé d'utiliser d'autres méthodes pour traiter les valeurs manquantes.



## Remplacer les **NaN** par le **MODE** de la colonne considérée

On peut utiliser le **MODE** si:

- 👉 Les données sont de type **nominales** ou **ordinales**
- 👉 Le total de **NaN** est faible par rapport à la taille de la colonne considérée

```
df.fillna(df.value_counts().index[0], inplace = True)
features = ['col_1', 'col_2', ... , 'col_n']
for feature in features:
    df[feature] = df[feature].fillna(df[feature].value_counts().index[0])
```

- ✓ Notez que la segmentation peut contribuer à améliorer les performances de votre futur modèle pour les données de type **nominales** ou **ordinales**.

## Remplacer les NaN par le MODE de la colonne considérée

On peut utiliser le **MODE** si:

- Les données sont de type **nominales** ou **ordinales**
- Le total de **NaN** est faible par rapport à la taille de la colonne considérée

```
■ df.fillna(df.value_counts().index[0], inplace = True)
```





## Remplacer les **NaN** par le **MODE** de la colonne considérée

On peut utiliser le **MODE** si:

- 👉 Les données sont de type **nominales** ou **ordinales**
- 👉 Le total de **NaN** est faible par rapport à la taille de la colonne considérée

```
■ df.fillna(df.value_counts().index[0], inplace = True)
■ features = ['col_1', 'col_2', ... , 'col_n']
for feature in features:
    df[feature] = df[feature].fillna(df[feature].value_counts().index[0])
```

- ✓ Notez que la segmentation peut contribuer à améliorer les performances de votre futur modèle pour les données de type **nominales** ou **ordinales**.

## Remplacer les NaN par une **CONSTANTE**

Les raisons du choix d'une constante doivent être justifiées

### ■ **df["colname"].fillna(0, inplace = True)**

■ df['colname'] = df['colname'].replace(np.nan, 0)

■ df["colname"].fillna("Absent", inplace = True)

■ df.fillna("Absent", inplace = True)

■ features = ['col\_1', 'col\_2', ... , 'col\_n']

for feature in features:

    df[feature] = df[feature].fillna('Absent')

❖ Zéro, absent ou tout autre valeur de votre choix

## Remplacer les NaN par une **CONSTANTE**

Les raisons du choix d'une constante doivent être justifiées

- `df["colname"].fillna(0, inplace = True)`
- `df['colname'] = df['colname'].replace(np.nan, 0)`
- `df["colname"].fillna("Absent", inplace = True)`
- `df.fillna("Absent", inplace = True)`
- `features = ['col_1', 'col_2', ... , 'col_n']`  
for feature in features:  
    `df[feature] = df[feature].fillna('Absent')`
- ❖ Zéro, absent ou tout autre valeur de votre choix

## Remplacer les NaN par une **CONSTANTE**

Les raisons du choix d'une constante doivent être justifiées

- `df["colname"].fillna(0, inplace = True)`
- `df['colname'] = df['colname'].replace(np.nan, 0)`
- `df["colname"].fillna("Absent", inplace = True)`
- `df.fillna("Absent", inplace = True)`
- `features = ['col_1', 'col_2', ... , 'col_n']`  
for feature in features:  
    `df[feature] = df[feature].fillna('Absent')`
- ❖ Zéro, absent ou tout autre valeur de votre choix

## Remplacer les NaN par une **CONSTANTE**

Les raisons du choix d'une constante doivent être justifiées

- `df["colname"].fillna(0, inplace = True)`
- `df['colname'] = df['colname'].replace(np.nan, 0)`
- `df["colname"].fillna("Absent", inplace = True)`
- **`df.fillna("Absent", inplace = True)`**

```
features = ['col_1', 'col_2', ... , 'col_n']  
for feature in features:  
    df[feature] = df[feature].fillna('Absent')
```

❖ Zéro, absent ou tout autre valeur de votre choix

## Remplacer les NaN par une **CONSTANTE**

Les raisons du choix d'une constante doivent être justifiées

- `df["colname"].fillna(0, inplace = True)`
- `df['colname'] = df['colname'].replace(np.nan, 0)`
- `df["colname"].fillna("Absent", inplace = True)`
- `df.fillna("Absent", inplace = True)`
- **`features = ['col_1', 'col_2', ... , 'col_n']`**  
**for feature in features:**  
**`df[feature] = df[feature].fillna('Absent')`**

❖ Zéro, absent ou tout autre valeur de votre choix

## Remplacer les NaN par une **CONSTANTE**

Les raisons du choix d'une constante doivent être justifiées

- `df["colname"].fillna(0, inplace = True)`
- `df['colname'] = df['colname'].replace(np.nan, 0)`
- `df["colname"].fillna("Absent", inplace = True)`
- `df.fillna("Absent", inplace = True)`
- `features = ['col_1', 'col_2', ... , 'col_n']`  
for feature in features:  
    `df[feature] = df[feature].fillna('Absent')`

❖ Zéro, absent ou tout autre valeur de votre choix

Remplacer les **NaN** par le **BFILL** ou **FFILL** de la colonne considérée

On peut utiliser le **BFILL**(backfill) ou **FFILL**(forwardfill) si:

- 👉 Les données sont de type **séries temporelles**(time series data)
- Le total de NaN est faible par rapport à la taille de la colonne considérée
- `df["colname"] = df["colname"].fillna(method = 'bfill')`
- `df["colname"] = df["colname"].fillna(method = 'ffill')`
- `df.fillna(method = 'ffill', inplace = True)`
- `df.fillna(method = 'bfill', limit = 3, inplace = True)`



Remplacer les **NaN** par le **BFILL** ou **FFILL** de la colonne considérée

On peut utiliser le **BFILL**(backfill) ou **FFILL**(forwardfill) si:

- 👉 Les données sont de type **séries temporelles**(time series data)
- 👉 Le total de **NaN** est faible par rapport à la taille de la colonne considérée

Remplacer les **NaN** par le **BFILL** ou **FFILL** de la colonne considérée

On peut utiliser le **BFILL**(backfill) ou **FFILL**(forwardfill) si:

- 👉 Les données sont de type **séries temporelles**(time series data)
- 👉 Le total de **NaN** est faible par rapport à la taille de la colonne considérée

```
■ df["colname"] = df["colname"].fillna(method = 'bfill')
```







## Suppression de NaN

🚫 La suppression de données sera la méthode utilisée en dernier recours car elle engendre une perte importante d'informations

## Suppression de NaN

👉 La suppression de données sera la méthode utilisée en dernier recours car elle engendre une perte importante d'informations

```
■ df.dropna(how = 'any', axis = "rows", inplace = True)
```

## Suppression de NaN

📌 La suppression de données sera la méthode utilisée en dernier recours car elle engendre une perte importante d'informations

- **df.dropna(how = 'any', axis = "rows", inplace = True)**
- **df.dropna(how = 'all', axis = 0, inplace = True)**
- df.dropna(subset = ['col1', 'col2', ..., 'col\_n'], inplace = True)
- df.dropna(axis = 1, inplace = True)



## Suppression de NaN

📌 La suppression de données sera la méthode utilisée en dernier recours car elle engendre une perte importante d'informations

- **`df.dropna(how = 'any', axis = "rows", inplace = True)`**
- **`df.dropna(how = 'all', axis = 0, inplace = True)`**
- **`df.dropna(subset = ['col1', 'col2', ..., 'col_n'], inplace = True)`**
- `df.dropna(axis = 1, inplace = True)`


📌 Il est préférable de supprimer les lignes et **non les colonnes**

## Suppression de NaN

📌 La suppression de données sera la méthode utilisée en dernier recours car elle engendre une perte importante d'informations

- `df.dropna(how = 'any', axis = "rows", inplace = True)`
- `df.dropna(how = 'all', axis = 0, inplace = True)`
- `df.dropna(subset = ['col1', 'col2', ..., 'col_n'], inplace = True)`
- `df.dropna(axis = 1, inplace = True)`

## Suppression de NaN

 La suppression de données sera la méthode utilisée en dernier recours car elle engendre une perte importante d'informations

- **`df.dropna(how = 'any', axis = "rows", inplace = True)`**
- **`df.dropna(how = 'all', axis = 0, inplace = True)`**
- **`df.dropna(subset = ['col1', 'col2', ..., 'col_n'], inplace = True)`**
- **`df.dropna(axis = 1, inplace = True)`**

 Il est préférable de supprimer les lignes et **non les colonnes**

## Valeurs aberrantes

Quelles méthodes choisir ?

## Valeurs aberrantes

## Quelles méthodes choisir ?



