



# Collecte, manipulation et traitement des données avec Python

Année académique 2024-2025

M2 DI:

Enseignant : David Rhenals

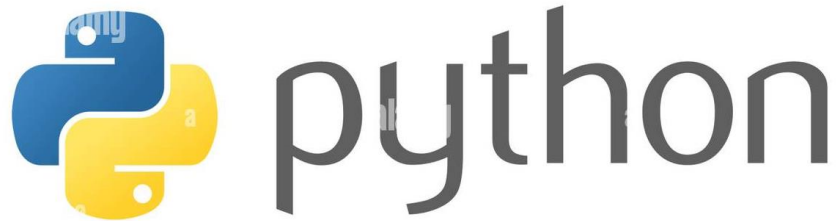
**Clause de confidentialité :** Ce cours est à usage unique des étudiants de l'ENSITECH la diffusion externe de tout contenu de ce cours est strictement interdite sans l'autorisation écrite préalable de l'enseignant.

# Objectifs du Cours

- Apprendre à utiliser les fonctionnalités avancées de Pandas (comme l'agrégation, le regroupement, les jointures, etc.) pour manipuler et analyser facilement des ensembles de données sur Python.
- Acquérir une maîtrise des techniques de prétraitement des données avec Python(**Scikit-learn**) afin d'optimiser les entrées pour le développement de modèles d'apprentissage automatique

- Installation et configuration de l'environnement de travail (Python)
  - Anaconda
  - Python (installation de librairies – optionnel)
    - Jupyter Notebook /Jupyter Lab (Environnement de Développement Interactif)
    - Spyder (Environnement de Développement Intégré)
  - Quelques rappels basiques de programmation python
  - Test simple de notions basiques de python
- Manipulation, nettoyage et transformation de données structurées sur python à la aide des bibliothèques Pandas et Scikit-Learn
  - Structures de données gérées par pandas (Series et Data Frames) ?
  - Principales attributs des séries et des Data frames en pandas
  - Indexation et découpage en pandas
  - Principales fonctionnalités de pandas pour la gestion d'opérations sur les séries et les data frames
    - Synthèse des principales fonctionnalités de pandas
  - Qu'est-ce que scikit-learn et quels sont ses principaux avantages (bibliothèque utilisée pour le traitement et la mise en œuvre des modèles de ML)
  - Nettoyage et transformation de données (phase de prétraitement) à l'aide de scikitlearn
    - Importance et principales objectifs
    - Tableau de synthèse de transformation de données
- Quelques exemples illustratifs de manipulation de données structurées et prétraitement
  - Création de séries et de data frames (listes, dictionnaires, import de fichiers)
  - Filtrage, regroupement et agrégation
  - Traitement de données manquantes, encodage, normalisation et définition de pipelines de traitement
- Mise en pratique de manipulation, analyse et traitement de données à travers des exercices simples sur Jupyter Notebooks

# Installation et configuration de l'environnement Python



- ✓ Langage de programmation interprété et polyvalent (relativement facile à lire et à écrire – exécution ligne par ligne)
- ✓ Multiplateforme (Windows, macOS et Linux)
- ✓ Large bibliothèque standard pour manipulation de tâches courantes (fichiers, opérations mathématiques, gestion protocoles internet, etc.)
- ✓ Programmation orienté objet (POO) et fonctionnelle
- ✓ Applications diverses (Développement web, Développement logiciel, Administration de systèmes, **Data Science, Analyse de données, intelligence artificielle, machine learning**, etc)
- ✓ Entre autres



- ✓ Distribution de python (et R) avec des packages ou librairies préinstallées. Elle est préconçue pour la science de données (cela ne veut pas dire que celle-ci ne peut pas être étendue à d'autres tâches).
- ✓ Inclut non seulement l'interpréteur python, mais aussi des outils comme Jupyter, Spyder (IDE pour python), anaconda prompt (terminal) et des nombreux packages pour la science de données (Numpy, Pandas, Matplotlib, Scipy, Scikit-learn, entre autres)
- ✓ Permet de démarrer plus facilement des projets de science de données grâce à sa configuration initiale par rapport à python où il serait nécessaire l'installation de toutes les librairies

# Installation et configuration de l'environnement Python (Anaconda)

- Télécharger la distribution Anaconda depuis le site [anaconda](https://anaconda.org/). Les instructions d'installation pour les différents systèmes d'exploitation ([Windows, macOS et Linux](#)).
- Fournir l'adresse email afin d'avoir accès au fichier exécutable .exe (windows) ou .pkg (mac)
- Exécuter le fichier .exe ou .pkg avec les options par défaut
- Finalisée l'installation, le navigateur Anaconda devrait être disponible dans le menu windows et mac.

# Installation et configuration de l'environnement Python

**Navigateur anaconda**

**Terminal anaconda**

**Jupyter**

**Spyder**

**Gestionnaire d'environnements**

**Gestionnaire de packages**

The image shows the Anaconda Navigator interface. On the left, a Windows Start menu search results for 'Anaconda3 (64-bit)' are shown, with a red box highlighting the 'Anaconda Navigator' application. Arrows point from labels 'Navigateur anaconda', 'Terminal anaconda', 'Jupyter', and 'Spyder' to the corresponding icons in the Start menu. The main window of Anaconda Navigator is displayed, showing the 'Environments' tab. A blue box highlights the 'Environments' list, with arrows pointing to labels 'création', 'clonage', 'Import', 'Sauvegarde', and 'Suppression' above the 'Create', 'Clone', 'Import', 'Backup', and 'Remove' buttons respectively. An orange box highlights the 'Channels' tab, showing a list of installed packages. An arrow points from the 'Gestionnaire de packages' label to this tab.

Name	Description	Version
✓ _anaconda_depends	Simplifies package management and deployment of anaconda	2024.02
✓ absl-cpp	Abseil common libraries (c++)	2021110
✓ aiobotocore	Async client for aws services using botocore and aiohttp	2.7.0
✓ aiohttp	Async http client/server framework (asyncio)	3.9.3
✓ aioitertools	Asyncio version of the standard multiprocessing module	0.7.1
✓ aiosignal	Aiosignal: a list of registered asynchronous callbacks	1.2.0
✓ alabaster	Lightweight, configurable sphinx theme	0.7.12
✓ altair	A declarative statistical visualization library for python	5.0.1
✓ anaconda-anon-us...	Basic anonymous telemetry for conda clients	0.4.3
✓ anaconda-catalogs	Client library to interface with anaconda cloud catalogs service	0.2.0
✓ anaconda-client	Anaconda.org command line client library	1.12.3
✓ anaconda-cloud-a...	A client auth library for anaconda.cloud apis	0.5.1
✓ anaconda-project	Tool for encapsulating, running, and reproducing data science projects	0.11.1
✓ annotated-types	Reusable constraint types to use with typing annotated	0.6.0

524 packages available

# Composantes basiques – Jupyter NB-LAB-Spyder

Diagram illustrating the basic components of three Python development environments: Spyder, Jupyter Lab, and Jupyter Notebook.

**Labels and Components:**

- Navigateur de fichiers** (File Explorer): Points to the file browser in Spyder, Jupyter Lab, and Jupyter Notebook.
- Editeur de code python** (Python Code Editor): Points to the code editor in Spyder.
- Console python** (Python Console): Points to the IPython console in Spyder.
- Explorateur variables-graphes** (Variable Explorer): Points to the variable explorer in Spyder.
- Navigateur de fichiers** (File Browser): Points to the file browser in Jupyter Lab and Jupyter Notebook.
- Volet d'affichage de notebooks, texte et code python** (Notebook, Text, and Python Code Display Panel): Points to the main content area in Jupyter Lab and Jupyter Notebook.

**Interfaces:**

- Spyder**: A desktop application with a menu bar (Fichier, Édition, Recherche, Source, Exécution, Déboguer, Console, Projets, Outils, Affichage, Aide), a file explorer, a code editor, a variable explorer, and a console.
- Jupyter Lab**: A web-based interface with a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help), a file browser, a notebook editor, and a console.
- Jupyter Notebook**: A web-based interface with a menu bar (File, View, Settings, Help), a file browser, and a notebook editor.

# Tableau comparatif Jupyter Notebook/Lab/Spyder

Caractéristique	Jupyter Notebook	Jupyter Lab	Spyder
<b>Type</b>	Environnement de notebook interactif	Environnement de développement web interactif	IDE scientifique pour Python
<b>Interface</b>	Basée sur des cellules, simple et intuitive	Basée sur des onglets, personnalisable, interface web moderne	Interface classique d'un IDE, avec éditeur, console, explorateur de variables
<b>Flexibilité</b>	Moins flexible, mais suffisante pour de nombreux cas d'utilisation	Très flexible, hautement extensible grâce aux extensions	Moins flexible que JupyterLab, mais plus structuré que Jupyter Notebook
<b>Collaboration</b>	Facile à partager, mais moins collaboratif en temps réel	Très collaboratif grâce aux fonctionnalités de partage et de co-édition	Moins axé sur la collaboration en temps réel, mais peut être utilisé en réseau
<b>Visualisation</b>	Intègre bien les bibliothèques de visualisation (Matplotlib, Seaborn, etc.)	Intègre bien les bibliothèques de visualisation, plus de possibilités de personnalisation	Intègre bien les bibliothèques de visualisation, interface dédiée à l'inspection des données
<b>Débogage</b>	Débogage cellulaire, mais moins complet qu'un IDE classique	Débogage plus avancé, avec points d'arrêt, inspection de variables, etc.	Débogage complet avec toutes les fonctionnalités d'un IDE classique
<b>Profiling</b>	Possibilité de profiler le code pour optimiser les performances	Possibilité de profiler le code	Possibilité de profiler le code
<b>Extensions</b>	Moins d'extensions disponibles	Nombreuses extensions disponibles pour personnaliser l'interface et ajouter des fonctionnalités	Moins d'extensions disponibles que JupyterLab, mais une communauté active
<b>Intégration avec d'autres outils</b>	S'intègre bien avec d'autres outils de la stack data science (Git, Docker, etc.)	S'intègre bien avec d'autres outils de la stack data science	S'intègre bien avec d'autres outils de la stack data science
<b>Utilisation typique</b>	Exploration de données, prototypage rapide, création de rapports interactifs	Développement de projets data science plus complexes, collaboration en équipe, enseignement	Développement scientifique en Python, analyse de données, prototypage



# Création d'environnement virtuel (Anaconda-Python) par ligne de commandes

- Clonant anaconda environnement de base  
**conda create --name <myenv> -- clone base**
- Création d'environnement python depuis anaconda  
**conda create --name <myenv> python=3.8**
- Activation de l'environnement python  
**conda activate myenv**
- Désactivation de l'environnement python  
**conda deactivate**
- Installation avec un fichier requirements.txt  
**pip install -r requirements.txt**
- Définition du workspace jupyter lab et jupyter notebook  
**jupyter notebook --notebook-dir <workdir>**  
**jupyter lab --notebook-dir <workdir>**

# Quelques rappels basiques de programmation Python

- Quelques notions de base en programmation sur python sont nécessaires pour continuer le cours. ces notions incluent les concepts:
  - Type de données
  - Structures de control
  - Built-in structures
  - modules
  - fonctions
- Pour rappel, voici quelques exemples simples rappelant ces concepts (cliquer [ici](#))

# Manipulation de données structurées avec Python (Pandas)

# Qu'est-ce que Pandas

- **Pandas** est une bibliothèque open-source en Python, largement utilisée pour la manipulation et l'analyse de données. Elle est construite sur **NumPy** et fournit des structures de données et des outils performants pour travailler avec des données tabulaires et structurées.
- Principales utilisations de pandas :
  - **Lire et écrire des données** dans différents formats (CSV, Excel, SQL, JSON, etc.).
  - **Manipuler des données** (filtrage, transformation, agrégation, etc.)
  - **Nettoyer des données** (traitement des valeurs manquantes, gestion des doublons, etc.).
  - **Effectuer des analyses statistiques descriptives** (moyennes, médianes, corrélations, etc.).

# Structures de données gérées par pandas (Series et Data Frames)

- Une **série** est une structure unidimensionnelle qui ressemble à un tableau (ou une liste). C'est l'équivalent d'une colonne de données dans une feuille de calcul ou une base de données.
- Caractéristiques principales des séries pandas:
  - **Indexées**: chaque élément d'une série est associé à une étiquette (un index) qui peut être utilisé pour accéder aux valeurs.
  - **Homogènes**: contient un seul type de données (entiers, chaînes de caractères, flottants, etc.).
  - **Equivalence à Numpy**: Une série pandas peut être vue comme un tableau Numpy avec des étiquettes d'index.

# Structures de données gérées par pandas (Series et Data Frames)

- Un **DataFrame** est une structure bidimensionnelle, similaire à une feuille de calcul Excel ou à une table dans une base de données. Il est composé de plusieurs séries (colonnes) ayant chacune son propre type de données.
- Caractéristiques principales des DataFrames pandas:
  - **Indexés**: les lignes et les colonnes sont étiquetées avec des indices.
  - **Agrégation de séries**: chaque colonne est une série, mais les colonnes peuvent avoir des types de données différents.
  - **Hétérogénéité**: les data frames pandas peuvent contenir des données hétérogènes (par exemple, des nombres, des chaînes de caractères, des booléens, etc.).

# Principales attributs des séries et des Data frames en pandas

## Principales attributs des séries Pandas

Attribut	Description
<code>index</code>	Renvoie les indices de la série sous forme d'un objet Index.
<code>values</code>	Renvoie les valeurs contenues dans la série sous forme de tableau NumPy.
<code>name</code>	Nom de la série (chaîne de caractères), peut être défini ou modifié.
<code>dtype</code>	Type de données des éléments de la série (int64, float64, object, etc.).
<code>size</code>	Nombre total d'éléments dans la série.
<code>shape</code>	Forme de la série (nombre de lignes, puisque c'est unidimensionnel).
<code>nbytes</code>	Quantité de mémoire utilisée par la série en octets.
<code>is_unique</code>	Indique si les valeurs de la série sont uniques (True ou False).
<code>is_monotonic</code>	Indique si les valeurs de la série sont monotones croissantes.
<code>empty</code>	Indique si la série est vide (True ou False).
<code>hasnans</code>	Indique la présence de valeurs manquantes (NaN) dans la série.

## Principales attributs des Data Frames Pandas

Attribut	Description
<code>index</code>	Renvoie les indices du DataFrame sous forme d'un objet Index.
<code>columns</code>	Renvoie les noms des colonnes sous forme d'un objet Index.
<code>values</code>	Renvoie les données du DataFrame sous forme de tableau NumPy.
<code>dtypes</code>	Renvoie les types de données de chaque colonne du DataFrame.
<code>shape</code>	Forme du DataFrame sous forme de tuple (nombre de lignes, nombre de colonnes).
<code>size</code>	Nombre total d'éléments dans le DataFrame (lignes * colonnes).
<code>empty</code>	Indique si le DataFrame est vide (True ou False).
<code>ndim</code>	Nombre de dimensions du DataFrame (toujours 2).
<code>axes</code>	Renvoie une liste des axes (l'index et les colonnes).
<code>T</code>	Transpose le DataFrame (inverse les lignes et les colonnes).

# Indexation et découpage en pandas

- **Indexing** (indexation) en Pandas fait référence à la manière d'accéder à des éléments spécifiques d'une série ou d'un DataFrame en utilisant des **étiquettes d'index** ou des **positions**. Cela permet d'accéder à des lignes, des colonnes ou des éléments individuels.
- **Slicing** (découpage) consiste à extraire une sous-partie des données, généralement un sous-ensemble de lignes ou de colonnes, en utilisant une plage d'étiquettes ou d'indices.



# Indexation et découpage en pandas

<i>Opération</i>	<i>Description</i>
<code>df['colonne']</code>	Sélectionne une colonne spécifique par son nom (retourne une Series).
<code>df[['colonne1', 'colonne2']]</code>	Sélectionne plusieurs colonnes en spécifiant une liste de noms de colonnes.
<code>df.iloc[0]</code>	Sélectionne la première ligne du DataFrame en utilisant l'index numérique.
<code>df.iloc[0:3]</code>	Sélectionne les lignes de la 0 à la 2 (slicing avec indices numériques).
<code>df.loc['index']</code>	Sélectionne une ligne spécifique en utilisant une étiquette d'index.
<code>df.loc['index1':'index3']</code>	Sélectionne plusieurs lignes par une plage d'étiquettes d'index (inclusif).
<code>df.iloc[:, 0]</code>	Sélectionne toutes les lignes de la première colonne (par indice numérique de colonne).
<code>df.loc[:, 'colonne']</code>	Sélectionne toutes les lignes d'une colonne par son nom (slicing par étiquette).
<code>df.iloc[0:3, 0:2]</code>	Sélectionne un sous-ensemble de lignes et colonnes en utilisant des indices numériques.
<code>df.loc['index1':'index3', 'colonne1':'colonne2']</code>	Sélectionne un sous-ensemble de lignes et de colonnes en utilisant des étiquettes (slicing étiquettes).
<code>df.at['index', 'colonne']</code>	Sélectionne un seul élément en utilisant une étiquette de ligne et une étiquette de colonne.
<code>df.iat[0, 1]</code>	Sélectionne un seul élément en utilisant des indices numériques pour la ligne et la colonne.
<code>df[df['colonne'] &gt; 5]</code>	Filtre les lignes où les valeurs de la colonne spécifiée sont supérieures à une condition.
<code>df.loc[df['colonne'] == 'valeur']</code>	Filtre les lignes où les valeurs de la colonne sont égales à une valeur spécifique.
<code>df.iloc[-1]</code>	Sélectionne la dernière ligne du DataFrame en utilisant l'index négatif.
<code>df['colonne'].iloc[0:5]</code>	Sélectionne les 5 premières lignes d'une colonne spécifique.
<code>df.iloc[:, -1]</code>	Sélectionne toutes les lignes de la dernière colonne.
<code>df.loc[df['colonne'].isin([1, 2])]</code>	Sélectionne les lignes où les valeurs de la colonne appartiennent à une liste donnée (condition multiple).

# Principales fonctionnalités de pandas pour la gestion d'opérations sur les séries et les data frames

- Pandas possède un grand nombre de fonctionnalités pour la gestion de diverses opérations qui peuvent être appliquées sur les séries et les data frames. Ces fonctionnalités peuvent être classifiées par thématiques:
  - Lecture/Écriture de fichiers : Importation et exportation de données.
  - Sélection et filtrage : Extraction de données spécifiques.
  - Manipulation et transformation de données : Transformation et fusion de DataFrames.
  - Gestion des valeurs manquantes : Traitement des valeurs nulles (NaN)
  - Opérations sur les colonnes : Gestion des colonnes et de l'index.
  - Statistiques et agrégation : Calculs statistiques et regroupement.
  - Fonctions avancées : Fonctions plus complexes pour la manipulation des données.

# Synthèse des principales fonctionnalités de pandas

## Lecture et écriture de fichiers

Fonction	Appel	Description
<code>read_csv()</code>	<code>pd.read_csv('fichier.csv')</code>	Lit un fichier CSV et renvoie un Data Frame.
<code>read_excel()</code>	<code>pd.read_excel('fichier.xlsx')</code>	Lit un fichier xlsx et renvoie un Data Frame
<code>to_csv()</code>	<code>df.to_csv('fichier.csv')</code>	Exporte le Data Frame vers un fichier CSV.
<code>to_excel()</code>	<code>df.to_excel('fichier.xlsx')</code>	Exporte le Data Frame vers un fichier Excel.

## Sélection et filtrage

Fonction	Appel	Description
<code>head()</code>	<code>df.head(n)</code>	Affiche les premières `n` lignes d'un Data Frame.
<code>tail()</code>	<code>df.tail(n)</code>	Affiche les dernières `n` lignes d'un Data Frame.
<code>iloc[]</code>	<code>df.iloc[i, j]</code>	Sélectionne des données par position (index numérique).
<code>loc[]</code>	<code>df.loc['index_label', 'col']</code>	Sélectionne des données par étiquette (nom de ligne/colonne).
<code>query()</code>	<code>df.query('condition')</code>	Filtre les lignes d'un Data Frame en utilisant une condition exprimée sous forme de chaîne de caractères (requête SQL-like).

## Opérations sur les colonnes et les index

Fonction	Appel	Description
<code>info</code>	<code>df.info()</code>	Affiche un résumé concis du Data Frame, incluant les types de données et les valeurs nulles
<code>columns</code>	<code>df.columns</code>	Renvoie les noms des colonnes du Data Frame.
<code>index</code>	<code>df.index</code>	Renvoie l'index (étiquettes des lignes) du Data Frame.
<code>astype()</code>	<code>df['colonne'].astype(dtype)</code>	Change le type de données d'une colonne.
<code>set_index()</code>	<code>df.set_index('colonne')</code>	Définit une colonne comme index du Data Frame.
<code>reset_index()</code>	<code>df.reset_index()</code>	Réinitialise l'index du Data Frame, en le ramenant à l'index par défaut (numérique).

## Gestion de données manquantes

Fonction	Appel	Description
<code>fillna()</code>	<code>df.fillna(valeur)</code>	Remplace les valeurs manquantes ('NaN') par une valeur donnée.
<code>isna()</code>	<code>df.isna()</code>	Renvoie un Data Frame booléen indiquant les valeurs manquantes ('NaN').
<code>dropna()</code>	<code>df.dropna()</code>	Supprime les lignes ou colonnes contenant des valeurs manquantes.

## Manipulation et transformation de données

Fonction	Appel	Description
<code>drop()</code>	<code>df.drop('colonne', axis=1)</code>	Supprime des colonnes ou des lignes d'un Data Frame.
<code>groupby()</code>	<code>df.groupby('colonne')</code>	Grouper les données par une ou plusieurs colonnes et appliquer des opérations comme `sum()`, `mean()`, etc.
<code>merge()</code>	<code>pd.merge(df1, df2, on='col')</code>	Fusionne deux Data Frames en fonction d'une ou plusieurs colonnes.
<code>concat()</code>	<code>pd.concat([df1, df2], axis=0)</code>	Concatène deux ou plusieurs DataFrames le long des lignes ou des colonnes.
<code>sort_values()</code>	<code>df.sort_values('colonne')</code>	Trie les lignes d'un Data Frame par les valeurs d'une colonne.
<code>replace()</code>	<code>df.replace(a, b)</code>	Remplace des valeurs spécifiques dans un DataFrame par d'autres.
<code>pivot()</code>	<code>df.pivot(index, columns, values)</code>	Restructure un Data Frame en fonction d'un ou plusieurs index, colonnes et valeurs.
<code>melt()</code>	<code>pd.melt(df, id_vars, value_vars)</code>	Transforme un DataFrame large en un format long.

## Statistiques et agrégation

Fonction	Appel	Description
<code>describe()</code>	<code>df.describe()</code>	Fournit des statistiques descriptives des colonnes numériques du Data Frame.
<code>mean()</code>	<code>df['colonne'].mean()</code>	Calcule la moyenne de la colonne
<code>median()</code>	<code>df['colonne'].median()</code>	
<code>pivot_table()</code>	<code>df.pivot_table(values, index)</code>	Crée un tableau croisé dynamique à partir des données du Data Frame.
<code>value_counts()</code>	<code>df['colonne'].value_counts()</code>	Compte les occurrences uniques des valeurs dans une colonne.
<code>corr()</code>	<code>df.corr()</code>	Calcule la matrice de corrélation des colonnes numériques.
<code>cumsum()</code>	<code>df['colonne'].cumsum()</code>	Calcule la somme cumulée des valeurs sur une colonne.
<code>cumprod()</code>	<code>df['colonne'].cumprod()</code>	Calcule le produit cumulé des valeurs sur une colonne.

# Synthèse des principales fonctionnalités de pandas

## Fonctions avancées

Fonction	Appel	Description
<code>apply()</code>	<code>df.apply(func)</code>	Applique une fonction à chaque élément d'une colonne ou ligne du DataFrame.
<code>duplicated()</code>	<code>df.duplicated()</code>	Renvoie un booléen indiquant si une ligne est dupliquée.
<code>drop_duplicates()</code>	<code>df.drop_duplicates()</code>	Supprime les lignes dupliquées dans un DataFrame.
<code>resample()</code>	<code>df.resample('freq')</code>	Regroupe les données selon une certaine fréquence (par exemple, pour des données temporelles).
<code>rolling()</code>	<code>df['colonne'].rolling(window=n)</code>	Applique des calculs sur des fenêtres glissantes de longueur définie (par exemple, moyenne mobile).
<code>expanding()</code>	<code>df['colonne'].expanding()</code>	Applique des calculs cumulatifs en considérant toutes les valeurs jusqu'à un point donné.
<code>applymap()</code>	<code>df.applymap(func)</code>	Applique une fonction élémentaire à chaque cellule d'un DataFrame.
<code>nunique()</code>	<code>df['colonne'].nunique()</code>	Renvoie le nombre de valeurs uniques dans une colonne.

# Nettoyage et transformation de données à l'aide de scikitlearn

# Qu'est-ce que Scikit-learn et pourquoi l'utiliser

- Scikit-learn est une bibliothèque open-source de Python, principalement utilisée pour l'apprentissage automatique (machine learning) .
- Elle fournit des outils simples et efficaces pour le prétraitement, la modélisation et l'évaluation des performances des modèles de machine learning
- Compatible avec des bibliothèques Python populaires comme Pandas, NumPy, SciPy et Matplotlib
- Scikit-learn bénéficie d'une communauté large et d'une documentation complète qui permettent une prise en main rapide et une aide en cas de besoin.
- Scikit-learn est conçu pour être rapide et performant, même avec des jeux de données relativement importants, tout en restant accessible
- Scikit-learn peut être intégré à des bibliothèques spécialisées dans le traitement à grande échelle, telles que Dask-ML

# Nettoyage et transformation de données (Prétraitement)

- Le prétraitement des données est une étape cruciale dans le développement de modèles de Machine Learning. Elle consiste à préparer les données brutes afin de les rendre adaptées à l'apprentissage automatique.
- Les principales objectifs de la phase de prétraitement :
  - Nettoyage des données
    - Permet d'éliminer les données manquantes qui peuvent fausser les résultats. Les imputer (remplacer) ou les supprimer permet d'avoir un jeu de données complet
    - Permet d'identifier et de corriger les erreurs de saisie, les doublons ou les incohérences afin de garantir l'intégrité des données
  - Transformations nécessaires (mise à l'échelle des variables numériques et encodage des variables catégorielles). La sélection de variables pourrait également faire partie de cette phase, même si certains modèles permettent d'obtenir l'importance des variables dans les prédictions de la variable cible.
  - Permet l'entraînement des estimateurs et peut améliorer les performances du modèle

# Nettoyage et transformation de données (Prétraitement)

## ■ Nettoyage des Données

### ■ Gestion de données manquantes

- Imputation moyenne, médiane, valeur constante ou KNN pour des variables numériques
- Imputation pour la mode (valeur la plus fréquente dans la variable) ou valeur constante pour les variables catégorielles
- Suppression des valeurs manquantes (à utiliser avec attention!)
- Utilisation de modèles prédictifs (prédire les valeurs manquantes)

### ■ Suppression de doublons (réduction du nombre des lignes avec des informations redondantes)

## ■ Transformation des Données






- Normalisation: Mise à l'échelle des variables prédictives dans une intervalle spécifique (Généralement dans l'intervalle  $[0, 1]$ )
- Standardisation: Transformation des variables prédictives à des nouvelles variables centrées-réduites (variables supposées normalement distribuées  $\mu = 0$  et  $\sigma = 1$ ).
- Encodage des variables catégorielles
  - Encodage One-Hot : Pour chaque catégorie de la variable, celui-ci crée une colonne binaire
  - Encodage Ordinal : Attribue des entiers sur chaque catégorie basé dans la relation d'ordre de la variable catégorielle.

**Tableau de synthèse de transformation de données (cliquer [ici](#))**






# Quelques exemples illustratifs des fonctionnalités Pandas appliquées sur des séries et des data frames (Création de séries et de data frames)

## Tableau d'exemples introductifs à Pandas

Exemple	Lien html
Création de série à partir d'une liste aléatoire	
Création d'un data frame à partir d'un dictionnaire et changement d'indices	
Création d'un data frame à partir d'un fichier csv et changement d'indices	
Méthodologies de filtrage d'un data frame	
Regroupement et agrégation dans un data frame	

# Quelques exemples illustratifs de traitement de données avec scikit-learn

## Tableau d'exemples introductifs au traitement des données avec scikit-learn

Exemple	Lien html
Imputation de données numériques et catégorielles ordinales et nominales	
Encodage de variables catégorielles nominales et ordinales	
Normalisation de variables numériques (comparaison des méthodes personnalisée et sklearn)	
Traitement des données à l'aide d'un pipeline sklearn	