

# Participez à un concours sur la Smart City

...

Decembre 1, 2022

# Aperçu

L'objectif de ce premier projet est:

- La prise en main et l'utilisation des outils pour effectuer une analyse de données:
  - Python
  - Jupyter notebook
  - Libraries pour la data science: pandas, numpy, matplotlib etc...
- Effectuer une analyse de données à partir d'un dataset portant sur les arbres de Paris.
- Interpréter les résultats et trouver des suggestions pour végétaliser la ville et optimiser les tournées d'entretien des arbres de la ville.

# Les phases du projet

## Setup l'environnement

L'installation:

- Python
- Ide (editeur)
- Jupyter notebook
- Environnement virtuel python
- Libraries python

## Collecte des données

L'utilisation de pandas:

- Charger les données issues du fichier CSV
- Creation d'un DataFrame (tableau 2D)

## Nettoyage des données

Le nettoyage des données consiste à:

- Traitement des données invalides.
- Suppression des doublons.
- Remplacement de valeurs
- Detecte et filtre les outliers

## Analyse, visualisations et interpretations.

La finalité du projet consiste

- Analyser les différentes donnees
- Avoir une visualisation à l'aide de graphes
- Interpréter les résultats et en tirer des conclusions et faire des suggestions



# Python, Jupyter et les librairies

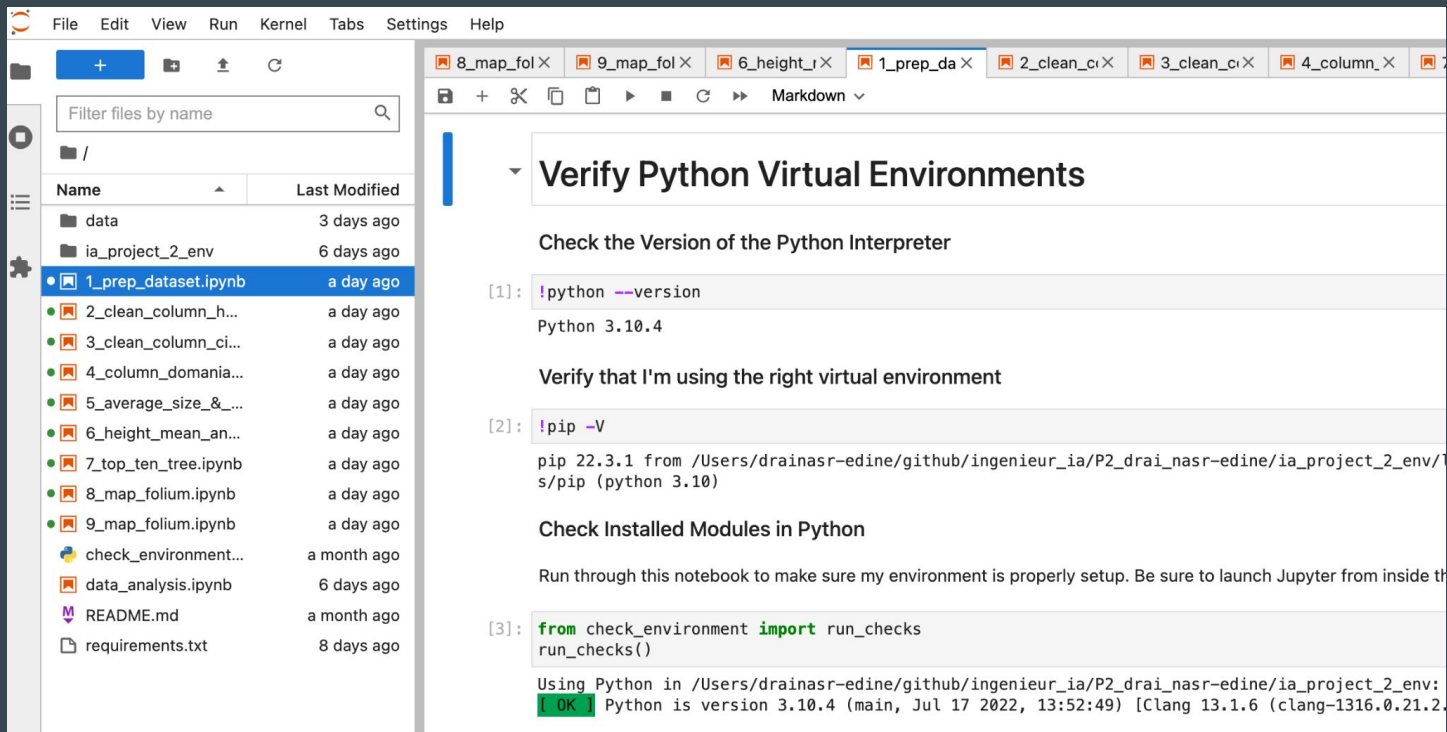


Python étant déjà installé sur mon poste avec une version récente (3.10.4). J'ai fait le choix d'utiliser PIP pour installer les modules nécessaires:

- Jupyterlab
- Pandas
- Matplotlib
- folium

---

# Jupyterlab



The screenshot displays the JupyterLab web interface. On the left is a file browser sidebar with a search bar and a list of files and folders. The main area on the right shows an open notebook titled 'Verify Python Virtual Environments'. The notebook contains three code cells:

- Cell [1]:** A command to check the Python version: `python --version`. The output is `Python 3.10.4`.
- Cell [2]:** A command to check the pip version: `pip -V`. The output is `pip 22.3.1 from /Users/drainasr-edine/github/ingenieur_ia/P2_drai_nasr-edine/ia_project_2_env/.../pip (python 3.10)`.
- Cell [3]:** A command to run a custom check: `from check_environment import run_checks; run_checks()`. The output is `Using Python in /Users/drainasr-edine/github/ingenieur_ia/P2_drai_nasr-edine/ia_project_2_env: [OK] Python is version 3.10.4 (main, Jul 17 2022, 13:52:49) [Clang 13.1.6 (clang-1316.0.21.2...]`.

Jupyter Lab et la nouvelle génération d'interface notebook qui offre d'avantage de fonctionnalités que jupyter notebook classique

# Vérification de l'environnement

# Vérification de l'environnement

## Commande bash dans le notebook:

Jupyter offre la possibilité d'utiliser bash en precedent la commande du symbole '!'

Vérifier python et l'environnement virtuel:

Commande	resultat
<code>!python --version</code>	Python 3.10.4
<code>!pip -V</code>	Chemin du virtual env

```
Check the Version of the Python Interpreter

[1]: !python --version

Python 3.10.4

Verify that I'm using the right virtual environment

[2]: !pip -V

pip 22.3.1 from /Users/drainasr-edine/github/ingenieur_ia/P2_drai_nasr-edine/ia_project_2_env/lib/python3.10/site-packages/pip (python 3.10)

Check Installed Modules in Python

Run through this notebook to make sure my environment is properly setup. Be sure to launch Jupyter from inside the virtual environment.

[3]: from check_environment import run_checks
run_checks()

Using Python in /Users/drainasr-edine/github/ingenieur_ia/P2_drai_nasr-edine/ia_project_2_env:
[ OK ] Python is version 3.10.4 (main, Jul 17 2022, 13:52:49) [Clang 13.1.6 (clang-1316.0.21.2.5)]

[ OK ] jupyterlab
[ OK ] matplotlib
[ OK ] numpy
[ OK ] pandas
[ OK ] seaborn
[ OK ] statsmodels
[ OK ] folium
```

# Verification des dependances

Un script python me permet de vérifier que les dépendances sont bien installés sur mon environnement virtuel

```
# check the requirements
for pkg in requirements:
    try:
        mod = importlib.import_module(pkg)
        print(f"{OK} {mod.__name__}")
    except ImportError:
        print(f"{FAIL} {pkg} not installed.")
```

```
3]: from check_environment import run_checks()
```

Using Python in /Users/drainasr-

[ OK ] Python is version 3.10.4

[ OK ] jupyterlab

[ OK ] matplotlib

[ OK ] numpy

[ OK ] pandas

[ OK ] seaborn

[ OK ] statsmodels

[ OK ] folium



# Collecte des données

# Collecte des données

## `pd.read_csv(path)`

Le fichier de données étant au format CSV, pandas permet de charger les données dans une structure de données appelées DataFrame.

**Le dataframe est un tableau a 2 dimensions, comprenant:**

- Lignes
- colones

```
df = pd.read_csv('data/p2_arbres_fr.csv', sep=';')
```

# Verification du dataframe

## `df.empty`

L'attribut empty permet un premiere examen du dataframe et vérifier qu'il n'est pas vide.

Is it empty?

```
[6]: df.empty
```

```
[6]: False
```

# Presentation des donnees

# Vue générale des données (statistiques)

`df.shape`

- Récupérer les dimensions du tableau
- On a 18 colonnes et 200137 lignes

`df.dtypes`

- Récupérer le type des données
- On a des floats, integers et objects

`df.describe()`

- Récupérer des statistiques sur les données
- Moyenne, écart-type, quartile etc..

`df.info()`

- Sommaire complet du dataframe
- On peut voir une colonne et des valeurs vides

# df.describe()

Permet d'obtenir des statistiques les plus courantes du dataframe.

On a notamment les statistiques suivantes:

- moyenne
- Ecart-type
- Mediane
- Quartile
- Min
- max

▼ ##### Get summary statistics

```
[12]: df.describe()
```

```
[12]:
```

	id	numero	circonference_cm
<b>count</b>	2.001370e+05	0.0	200137.000000
<b>mean</b>	3.872027e+05	NaN	83.380479
<b>std</b>	5.456032e+05	NaN	673.190213
<b>min</b>	9.987400e+04	NaN	0.000000
<b>25%</b>	1.559270e+05	NaN	30.000000
<b>50%</b>	2.210780e+05	NaN	70.000000
<b>75%</b>	2.741020e+05	NaN	115.000000
<b>max</b>	2.024745e+06	NaN	250255.000000

# df.info()

## Un résumé exhaustif du jeu de données.

On peut notamment visualiser les colonnes pour lesquelles il y a des valeurs nulles ou colonnes nulles

```
[13]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200137 entries, 0 to 200136
Data columns (total 18 columns):
#   Column                Non-Null Count  Dtype  
---  --
0   id                     200137 non-null  int64  
1   type_emplacement      200137 non-null  object  
2   domanialite           200136 non-null  object  
3   arrondissement        200137 non-null  object  
4   complement_adresse    30902 non-null   object  
5   numero                 0 non-null       float64 
6   lieu                  200137 non-null  object  
7   id_emplacement        200137 non-null  object  
8   libelle_francais      198640 non-null  object  
9   genre                 200121 non-null  object  
10  espece                198385 non-null  object  
11  variete               36777 non-null   object  
12  circonference_cm       200137 non-null  int64  
13  hauteur_m             200137 non-null  int64  
14  stade_developpement   132932 non-null  object  
15  remarquable           137039 non-null  float64 
16  geo_point_2d_a        200137 non-null  float64 
17  geo_point_2d_b        200137 non-null  float64 
dtypes: float64(4), int64(3), object(11)
memory usage: 27.5+ MB
```

# Le nettoyage des données

# Suppression des colonnes et lignes non pertinentes

On a notamment les cas suivantes:

- Colonne vide
- Colonne content 1 valeur unique
- Colonne contenant peu de valeurs
- Les lignes en doublon
- Filtrer les données hors périmètre

Colonne 'numero' est vide

We have a column `numero` empty. We can drop empty column

```
[15]: print("column is empty ?:", df.numero.isnull().values.all())  
df.drop('numero', axis=1, inplace=True)
```

column is empty ?: True

Le dataframe ne contient pas de doublons

Rows That Contain Duplicate Data

```
[26]: # calculate duplicates  
dups = df.duplicated()  
# report if there are any duplicates  
print(dups.any())
```

False



# Traitement des valeurs invalides

On a notamment les cas suivantes:

- Une hauteur ou circonférence égale à 0
- Une hauteur ou circonférence qui atteint des valeurs non possible

1 étape: on les remplace par Nan

2 étape: On calcule la valeur moyenne selon le genre d'arbre puis on lui impute.

On remplace 0 par Nan

```
df[['hauteur_m', 'circonference_cm']].replace(0, np.nan)
```

On remplace Nan par la valeur moyenne selon le type d'arbre qu'il s'agit.

```
df['hauteur_m'].fillna(df.groupby('genre')['hauteur_m'].transform('mean'),
```

---

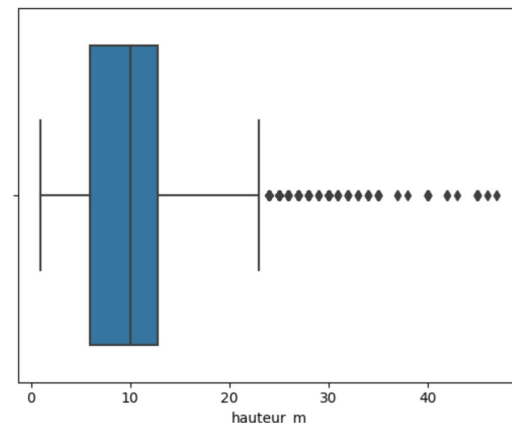
# Traitement des outliers

J'ai utilisé 3 méthodes pour les identifier

- Boxplot de Seaborn
- ecart-type
- interquartile

Ensuite les valeurs sont remplacés par la limite max

## Représentation graphique des outliers



1: Ecart-type

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}}$$

2: Interquartile

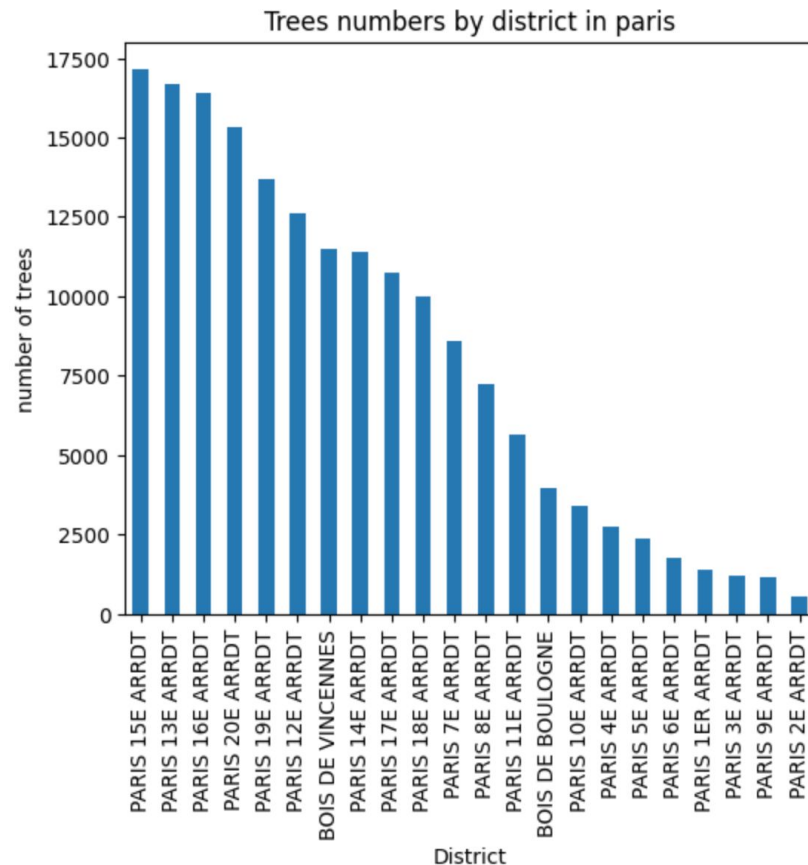
$$\text{IQR} = Q_3 - Q_1$$

# Analyse exploratoire

## Exemple d'analyse univariée

On constate nettement d'après ce graphique:

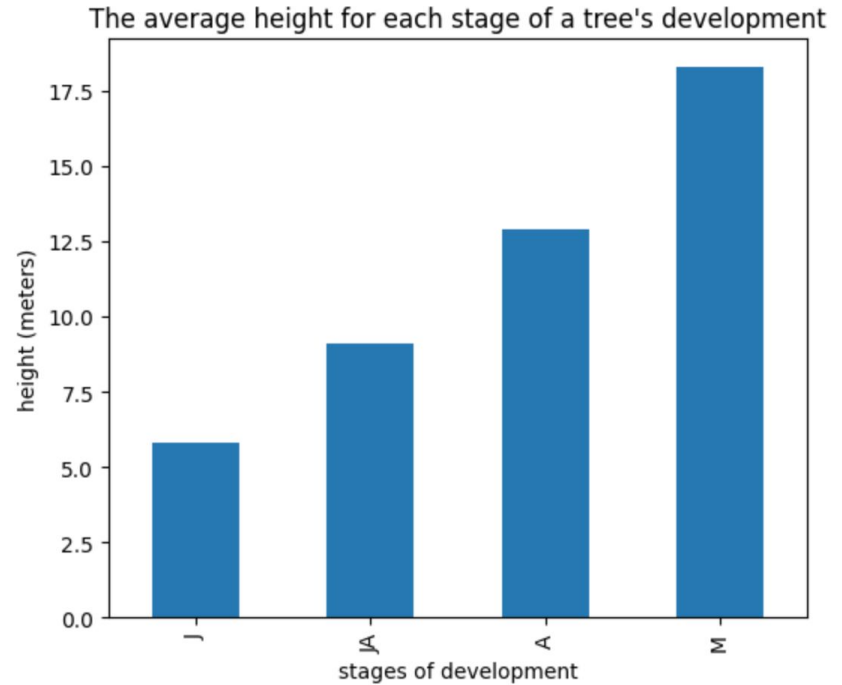
- Les arrondissements au centre de Paris ont une plus faible proportion d'arbre que les arrondissements périphériques



## Exemple d'analyse bivariée

On constate en comparant la hauteur moyenne des arbres au stade de développement:

- Les données fournies sont bien cohérente
- La hauteur des arbres est bien croissantes du stade le plus jeune au plus mature

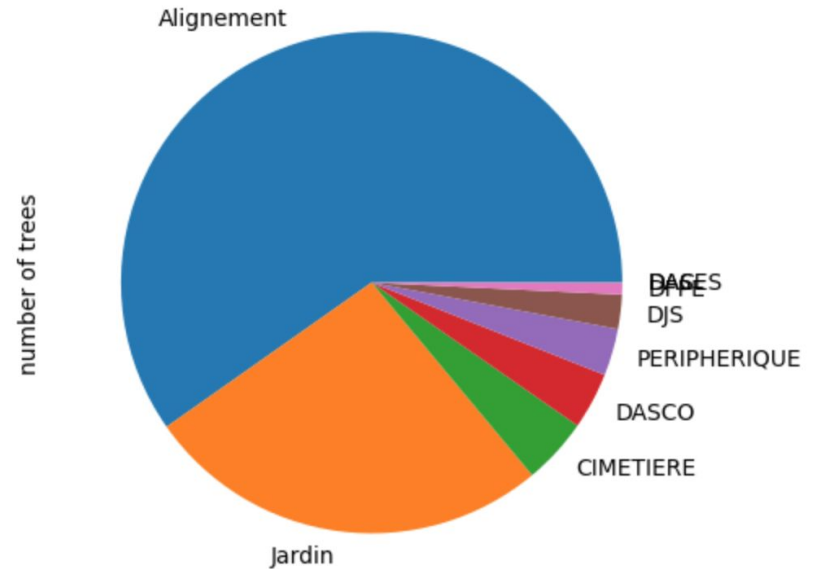


## Diagramme circulaire

On constate d'après ce diagramme:

- Les Alignements sont les espaces qui dominant nettement en nombre d'arbres
- Puis en second, ce sont les jardins

Trees distributions by estate (domanialite) in paris



# Carte

On constate d'après cette carte réalisé avec folium:

- Une concentration importante d'arbres au niveau des grands axes (routes, boulevards). Ce qui confirme notre précédente analyse qui a révélé que les alignements sont les endroits où on trouve le plus d'arbre



# Conclusion



# Conclusion de l'analyse de données

## Points important

Cette analyse a avant tout une portée pédagogique pour permettre à l'étudiant une prise en main des outils et méthodologies d'un data analyst.

Mais elle m'a permis de mettre en évidence les points suivants::

- Le centre de Paris est nettement moins boisé que les arrondissements plus périphériques. Donc il faut davantage planter des arbres dans le cœur de Paris.
- Les Alignements sont les espaces qui contiennent le plus grand nombre d'arbre, donc à prioriser pour les tournées
- La catégorie d'arbre la plus répandue est le Platane, quasiment sur tous les arrondissements. Cet arbre nécessite un entretien comme l'élagage plus particulièrement en automne. Donc recruter davantage de saisonniers par exemple en Octobre, Novembre.

# L'auteur



Nasr-edine Draï, étudiant  
ingénieur IA

---

J'ai obtenu un diplôme de  
développeur d'applications  
python OpenClassrooms et je  
poursuis mes études pour  
devenir ingénieur IA