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

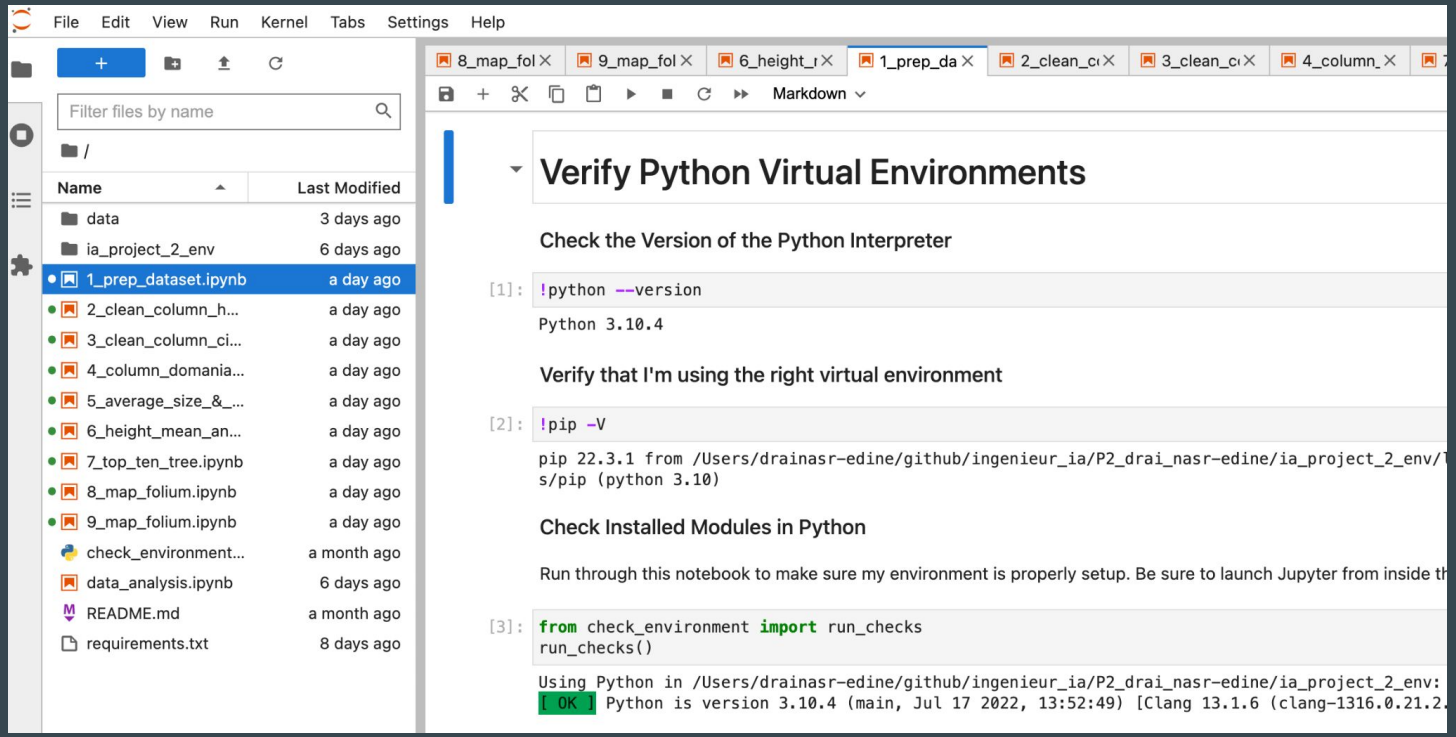
- Analyser les differentes donnees
- Avoir une visualisation à l'aide de graphes
- Interpréter les résultats et en tirer des conclusions et faire des suggestions

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



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

Vérification de l'environnement

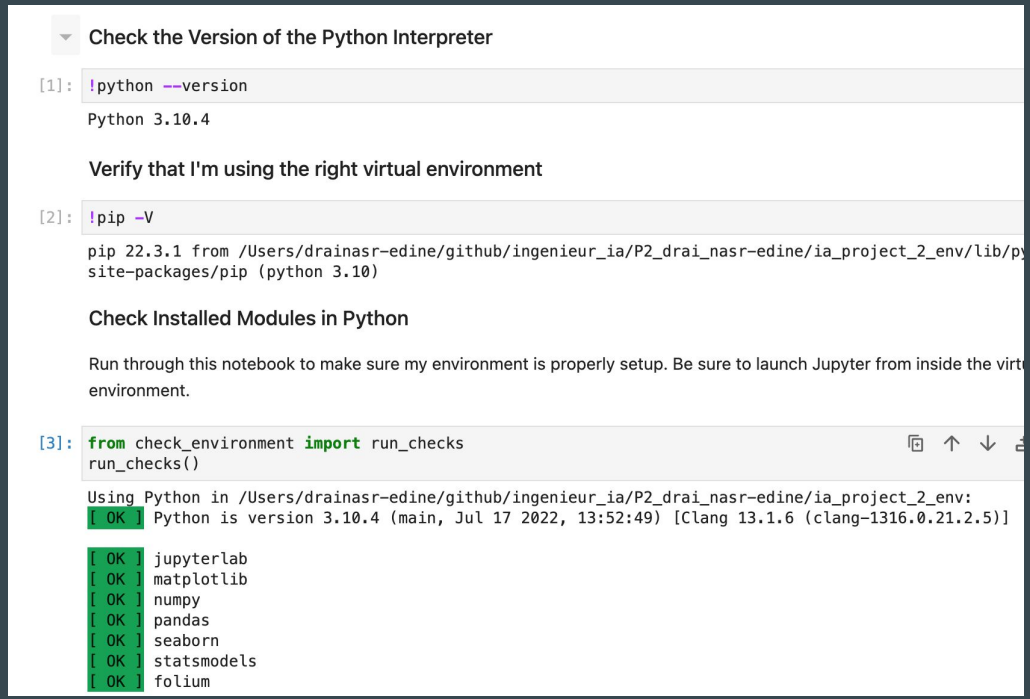
Vérification de l'environnement

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
!python --version	Python 3.10.4
!pip -V	Chemin du virtual env



Verification des dependances

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

```python
# 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 ru
    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:

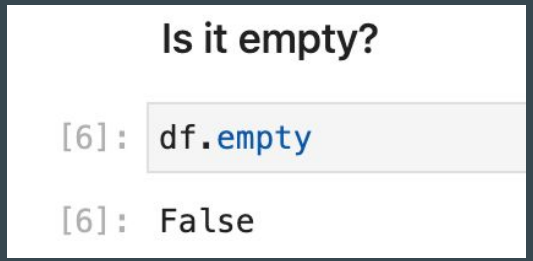
- Lignes
- colones

```
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.



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 • Et avoir une vue sur les ordres de grandeur
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. Et également d'avoir des ordres de grandeur des variables .

On note notamment les éléments suivants:

- On a une valeur minimum égale à 0
- On a une valeur max de 881818 m

Les ordres de grandeur min max ne semblent pas réaliste. Donc on pourra les traiter dans la phase de nettoyage

```
count    175689.000000
mean         14.272504
std        2103.900313
min           0.000000
25%           5.000000
50%           9.000000
75%          12.000000
max      881818.000000
Name: hauteur_m, dtype: float64
```

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_addresse  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
```

Les ordres de grandeur

On avait constaté à l'aide de la fonction describe de Pandas qu'on avait des ordres de grandeur qui semble surréaliste. Pour les traiter, j'ai procédé comme suit:

- Je me suis basé sur les reccords connu de la taille des arbres dont j'ai mis la reference dans mon notebook ainsi que dans le lien ci-dessous: https://www.monumentaltrees.com/fr/recordsdhauteur/fra/paris/
- Puis je les ai remplacés de la même façon que les autres valeurs invalides.

Quelques recherches sur Google m'ont permit de constater que l'arbre le plus grand de Paris est un séquoia qui mesure 31m

Records d'hauteur des arbres en Paris

Hauteur

Dans ce tableau des mesures d'hauteur en Paris seulement des mesures faites par laser (co considérées. Évidemment cet inventaire contient seulement des arbres dont les mesures s

Nr	Espèce de l'arbre	Hauteur	L
1	*Sequoiadendron giganteum* (Séquoia géant)	**31 m**	B
2	*Liriodendron tulipifera* (Tulipier de Virginie)	**26 m**	B
3	*Ginkgo biloba* (Ginkgo)	**25 m**	J

En me basant sur les références ci-dessus. J'ai remplacé les valeurs dépassant les records connus par Nan.

```
[4]: # count the number of values greather than 50
     len(df[df['hauteur_m'] >= 50])
```

```
[4]: 448
```

```
[5]: # replace values greather than 50m with 'nan'
     df.loc[df.hauteur_m >= 50, 'hauteur_m'] = np.nan
```

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'),
```

Les ordres de grandeurs:

La description de la colonne hauteur laisse apparaître des valeurs min et max incohérentes pour les dimensions d'un arbre.

- 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.

```
count    175689.000000
mean         14.272504
std        2103.900313
min           0.000000
25%           5.000000
50%           9.000000
75%          12.000000
max      881818.000000
Name: hauteur_m, dtype: float64
```

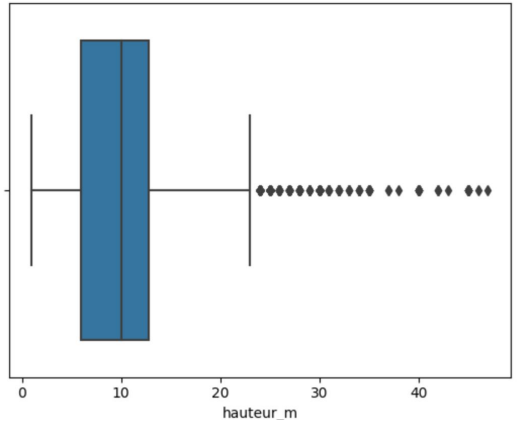
Traitement des outliers

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

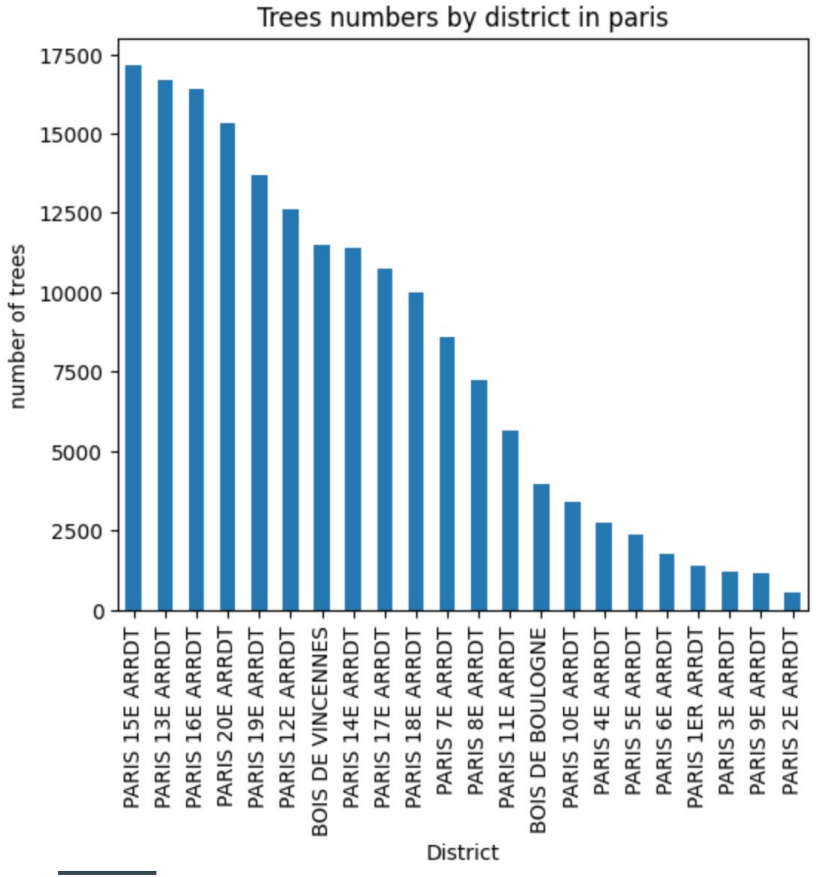
$$\mathrm{IQR} = Q_3 - Q_1$$

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

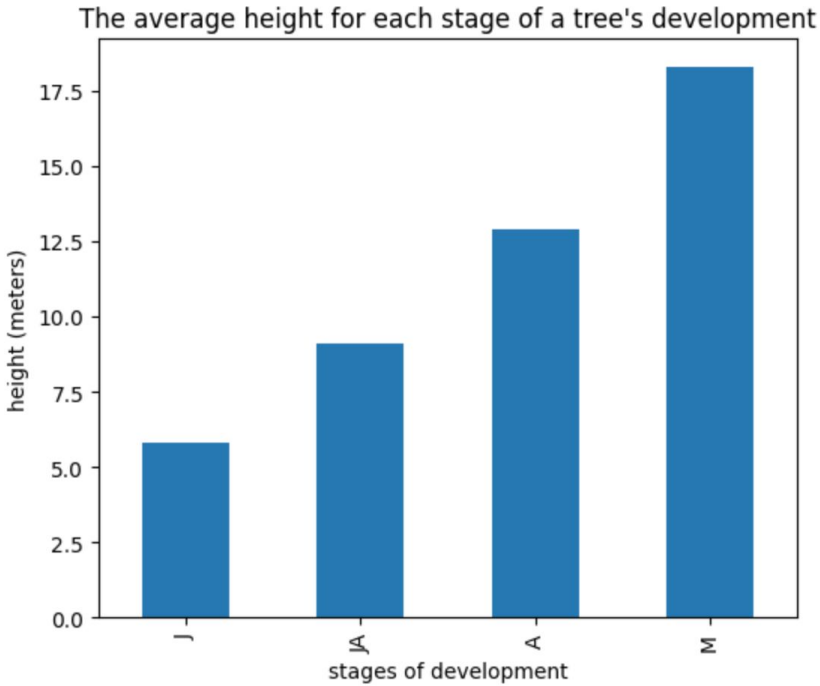
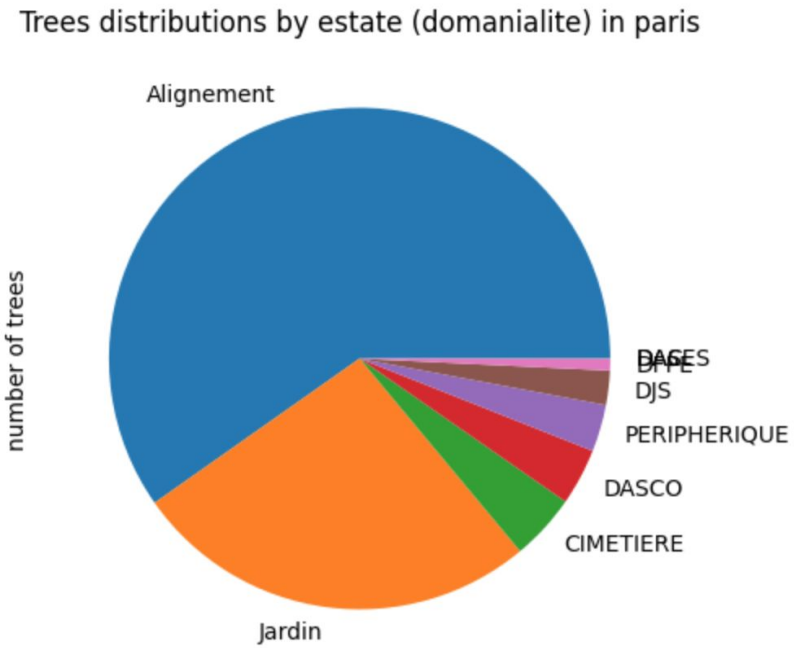


Diagramme circulaire

On constate d'après ce diagramme:

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



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 ou 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 permit 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 Drai, etudiant ingénieur IA

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