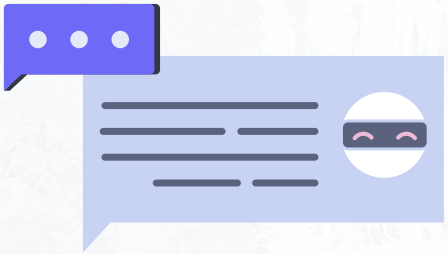
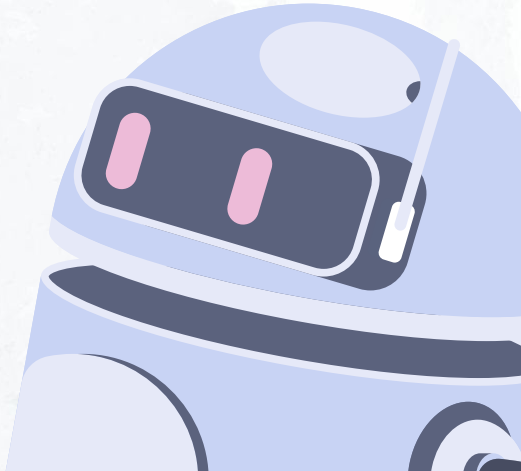
Participez à un concours sur la Smart City →

(IA)





Sommaire

01 → Rappel du contexte

02 → Démarche méthodologique d’analyse de données

03 → Présentation générale du jeu de données

04 → Synthèse

01 →

Rappel du contexte

(IA)

Rappel du contexte

Vous êtes fraîchement établi en tant qu'expert indépendant spécialisé en intelligence artificielle. Vous participez régulièrement à des concours pour vous **faire la main sur de nouveaux sujets**.

Vous avez décidé de participer à un challenge proposé par la ville de Paris !

Vos résultats contribueront à une optimisation des tournées pour l'entretien des arbres de la ville.

Règles du challenge

- Installez votre environnement de développement sur votre ordinateur, et créez un environnement virtuel dédié à ce challenge.
- Les données doivent être explorées à l'aide de Python et de ses librairies.
- Vous soumettrez votre analyse sous forme de présentation, contenant les informations suivantes :
    - Présentation générale du jeu de données
    - Démarche méthodologique d'analyse de données
    - Synthèse de l'analyse de données
- Le second livrable prendra la forme d'un Notebook Jupyter. Le Notebook sera documenté pour expliciter les différents traitements, calculs ou graphiques que vous effectuez.

Objectifs de l’analyse exploratoire

- Contribuer à une optimisation des tournées pour l’entretien des arbres de la ville.

02 →

Démarche méthodologique d’analyse de données

(IA)

TABLE DES MATIÈRES

01 | Paramétrage → Cette section sert à importer ou installer les librairies nécessaires, vérifier les versions.

02 | Chargement du jeu de données → Cette section sert à notre fichier plat.

03 | Exploration du jeu de données → Cette section sert à décrire et comprendre le jeu de données

04 | Nettoyage du jeu de données → Cette section sert à traiter les valeurs aberrantes et manquantes.

05 | Analyse univariée et représentations graphiques → Cette section vise à effectuer des analyses univariée et bivariée pour créer des représentations graphiques pour explorer les caractéristiques individuelles des données.

06 | Conclusion → Cette section vise à tirer des conclusions dans notre analyse et à mettre en lumière d'éventuelles solutions pour atteindre l'objectif

03 →

Présentation générale du jeu de données

(IA)

A. Affichage des cinq premières lignes de notre DataFrame :

Entrée [8]:
```
data.head()
```

Out[8]:

	id	type_emplacement	domanialite	arrondissement	complement_addresse	numero	lieu	id_emplacement	libelle_francais	genre	es
0	99874	Arbre	Jardin	PARIS 7E ARRDT	NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	19	Marronnier	Aesculus	hippocast
1	99875	Arbre	Jardin	PARIS 7E ARRDT	NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	20	If	Taxus	ba
2	99876	Arbre	Jardin	PARIS 7E ARRDT	NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	21	If	Taxus	ba
3	99877	Arbre	Jardin	PARIS 7E ARRDT	NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	22	Erable	Acer	neg
4	99878	Arbre	Jardin	PARIS 17E ARRDT	NaN	NaN	PARC CLICHY-BATIGNOLLES-MARTIN LUTHER KING	000G0037	Arbre à miel	Tetradium	da

B. Affichage du nombre de lignes et de colonnes dans le Data Frame :

Entrée [9]:
```
print("Nombre de colonnes :", data.shape[1])
print("Nombre de lignes :", data.shape[0])
```

```
Nombre de colonnes : 18
Nombre de lignes : 200137
```

C. Affichage des types de données de chaque colonne :

Entrée [10]:
```
print("Types de données :")
data.dtypes
```

```
Types de données :
```

Out[10]:
```
id                     int64
type_emplacement      object
domanialite           object
arrondissement        object
complement_addresse   object
numero               float64
lieu                  object
id_emplacement        object
libelle_francais      object
genre                 object
espece                object
variete               object
circonference_cm       int64
hauteur_m              int64
stade_developpement   object
remarquable          float64
geo_point_2d_a       float64
geo_point_2d_b       float64
dtype: object
```

D. Affichage des statistiques descriptives pour chaque colonne numérique (moyenne, médiane, min, max, écart-type, etc.) :

Entrée [11]:
```
data.describe()
```

Out[11]:

	id	numero	circonference_cm	hauteur_m	remarquable	geo_point_2d_a	geo_point_2d_b
count	2.001370e+05	0.0	200137.000000	200137.000000	137039.000000	200137.000000	200137.000000
mean	3.872027e+05	NaN	83.380479	13.110509	0.001343	48.854491	2.348208
std	5.456032e+05	NaN	673.190213	1971.217387	0.036618	0.030234	0.051220
min	9.987400e+04	NaN	0.000000	0.000000	0.000000	48.742290	2.210241
25%	1.559270e+05	NaN	30.000000	5.000000	0.000000	48.835021	2.307530
50%	2.210780e+05	NaN	70.000000	8.000000	0.000000	48.854162	2.351095
75%	2.741020e+05	NaN	115.000000	12.000000	0.000000	48.876447	2.386838
max	2.024745e+06	NaN	250255.000000	881818.000000	1.000000	48.911485	2.469759

E. Affichage du nombres de valeurs uniques pour chaque colonne

Entrée [12]:
```
data.nunique()
```

Out[12]:
```
id                    200137
type_emplacement           1
domanialite                9
arrondissement            25
complement_addresse     3795
numero                     0
lieu                    6921
id_emplacement         69040
libelle_francais         192
genre                    175
espece                   539
variete                  436
circonference_cm         531
hauteur_m                143
stade_developpement        4
remarquable                2
geo_point_2d_a        200107
geo_point_2d_b        200114
dtype: int64
```

F. Affichage du nombre de valeurs uniques pour chaque colonne catégorielle :

Entrée [13]:
```
data.select_dtypes(include=['object']).apply(pd.Series.value_counts)
```

Out[13]:

	type_emplacement	domanialite	arrondissement	complement_addresse	lieu	id_emplacement	libelle_francais	genre	espece	variete	stade_de
0	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	
00010238b	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	
00010264 /	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	
00010270 /	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	
00010271 /	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	
à côté de l'arrêt de bus appartement 29e	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	
à côté passerelle	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	

G. Affichage du pourcentage de valeurs manquantes par colonne :

Entrée [14]:
```
(data.isnull().mean() * 100).astype(int)
```

Out[14]:
```
id                     0
type_emplacement       0
domanialite            0
arrondissement         0
complement_addresse    84
numero                 100
lieu                   0
id_emplacement         0
libelle_francais       0
genre                  0
espece                 0
variete                81
circonference_cm       0
hauteur_m              0
stade_developpement    33
remarquable            31
geo_point_2d_a         0
geo_point_2d_b         0
dtype: int32
```

H. Affichage du nombre de lignes qui ont des valeurs manquantes :

Entrée [15]:
```
(data.isnull().sum(axis=1) > 0).sum()
```

Out[15]:
```
200137
```

I. Vérification de l'équilibre de notre DataFrame :

Notre objectif est d'optimiser les tournées d'entretien des arbres. Je me suis donc rapproché de professionnelles des espaces verts afin de voir avec eux quels features sont importants.

Après ses entretiens ces features semblent les plus importantes pour réaliser notre objectif :

- Espèce (pour l'entretien)
- Hauteur (Pour l'élagage)
- Stade de développement (Pour l'arrosage)
- Circonférence du tronc (Pour l'élagage)
- Arrondissement (pour la localisation)

Pour cela, il est important de comprendre comment nos données sont réparties en matière d'espèce, de hauteur, stade de développement, circonférence du tronc et d'arrondissement. Un déséquilibre dans ces données peut avoir un impact sur la planification des tournées.

Nous allons analyser la fréquence de chaque espèce d'arbre dans notre jeu de données. Une répartition inégale pourrait indiquer que certaines espèces d'arbres nécessitent plus d'entretien que d'autres.

Entrée [16]:
```
# Compter la fréquence de chaque espèce d'arbre
freq_espece = data['espece'].value_counts()

# Créer un DataFrame pour stocker les fréquences des espèces d'arbres
stats_espece = pd.DataFrame({'Espèce': freq_espece.index,
                             'Fréquence': freq_espece.values})

stats_espece
```

Out[16]:

	Espèce	Fréquence
0	x hispanica	36409
1	hippocastanum	20039
2	japonica	11822
3	n. sp.	9063
4	tomentosa	8962
...	...	...
534	polycarpa	1
535	x gondouinii	1
536	lusitanica subsp.azorica	1
537	oliveri	1
538	delavayi subsp. potaninii	1

539 rows × 2 columns

Nous allons examiner la distribution de la hauteur des arbres. Des arbres plus grands peuvent nécessiter un entretien plus fréquent ou plus spécialisé.

Nous allons examiner la distribution de la hauteur des arbres. Des arbres plus grands peuvent nécessiter un entretien plus fréquent ou plus spécialisé.

Entrée [17]:
```
# Nombre total d'arbres
total_arbres = data['hauteur_m'].count()

# Hauteur minimale des arbres
hauteur_min = data['hauteur_m'].min()

# Hauteur maximale des arbres
hauteur_max = data['hauteur_m'].max()

# Hauteur moyenne des arbres
hauteur_moy = data['hauteur_m'].mean()
```

Entrée [18]:
```
# Création d'un DataFrame pour stocker les statistiques de la hauteur des arbres
stats_hauteur = pd.DataFrame({'Total d\'arbres': [total_arbres],
                              'Hauteur minimale (m)': [hauteur_min],
                              'Hauteur maximale (m)': [hauteur_max],
                              'Hauteur moyenne (m)': [hauteur_moy]})

stats_hauteur
```

Out[18]:

	Total d'arbres	Hauteur minimale (m)	Hauteur maximale (m)	Hauteur moyenne (m)
0	200137	0	881818	13.110509

Nous allons examiner le stade de devloppement

Entrée [19]:
```
# Créer un dictionnaire de correspondance pour les stades de développement
correspondance_stades = {'A': 'Adulte', 'JA': 'Jeune adulte', 'J': 'Jeune', 'M': 'Mature'}

# Mapper les valeurs des stades de développement avec les noms complets
data['stade_complet'] = data['stade_developpement'].map(correspondance_stades)

# Compter le nombre d'arbres par stade de développement
arbres_par_stade = data['stade_complet'].value_counts()

# Créer un DataFrame pour stocker les nombres d'arbres par stade de développement
stats_stade_developpement = pd.DataFrame({'Stade de développement': arbres_par_stade.index,
                                          'Nombre d\'arbres': arbres_par_stade.values})

stats_stade_developpement
```

	Stade de développement	Nombre d'arbres
0	Adulte	64438
1	Jeune adulte	35444
2	Jeune	26937
3	Mature	6113

Nous allons examiner la circonferance des trons

Entrée [20]:
```
# Création d'un DataFrame pour stocker les statistiques de la circonférence des troncs
stats_circonf = pd.DataFrame({'Statistiques': ['Minimum', 'Maximum', 'Moyenne'],
                              'Valeurs': [data['circonference_cm'].min(), data['circonference_cm'].max(), data['circonference_cm
stats_circonf
```

Out[20]:

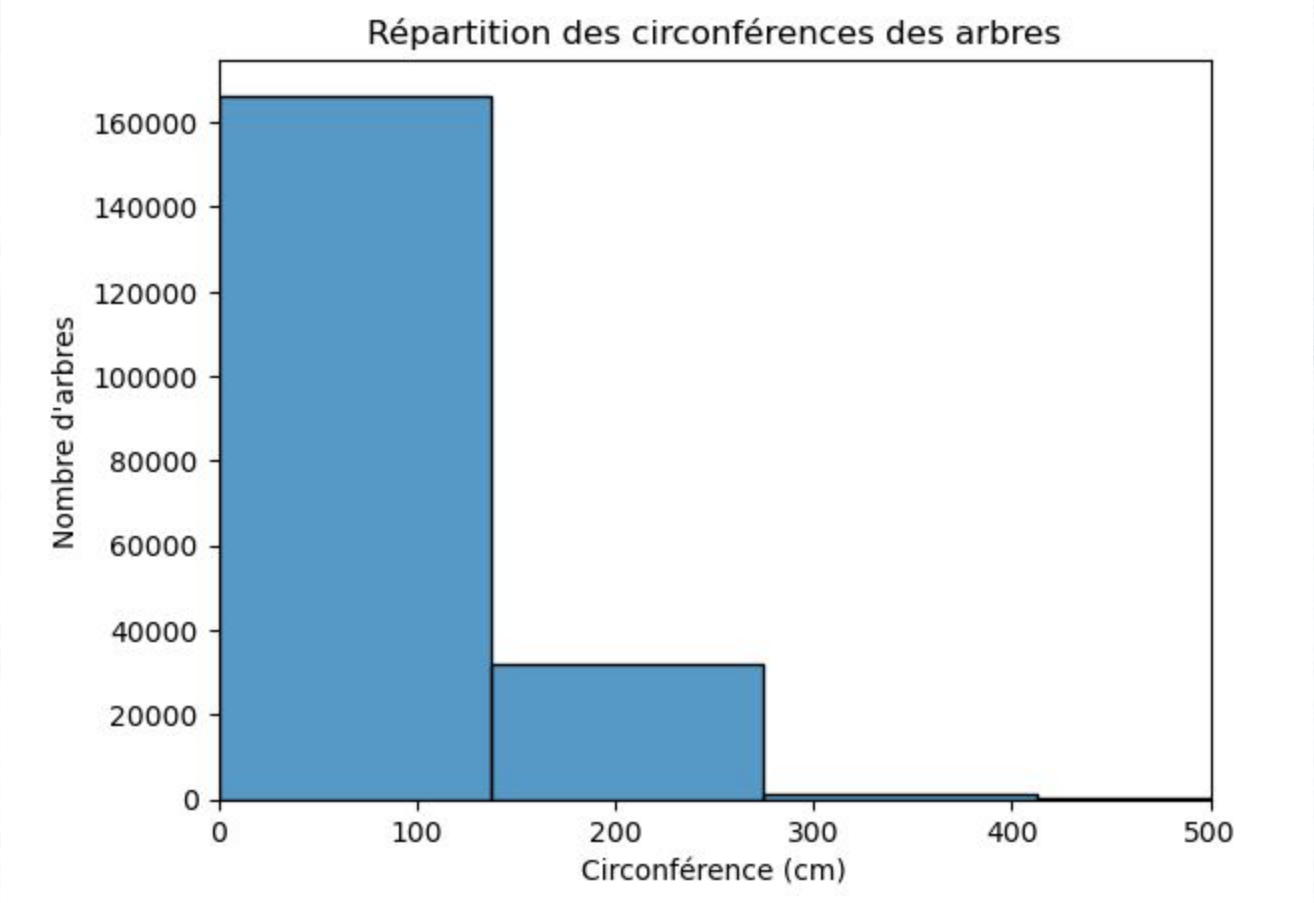
	Statistiques	Valeurs
0	Minimum	0.000000
1	Maximum	250255.000000
2	Moyenne	83.380479

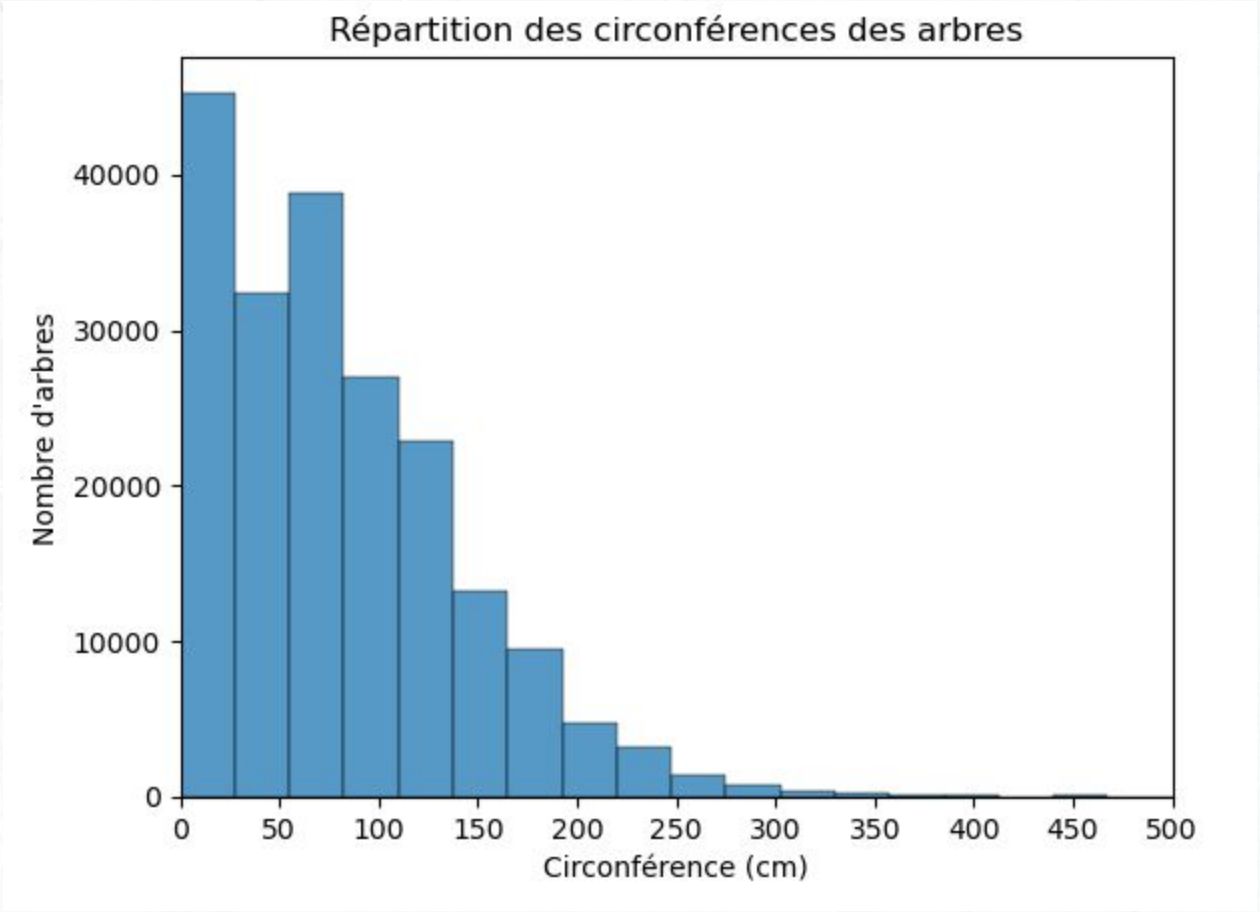
Nous allons compter le nombre d'arbres dans chaque arrondissement. Un arrondissement avec un grand nombre d'arbres peut nécessiter plus de tournées d'entretien.

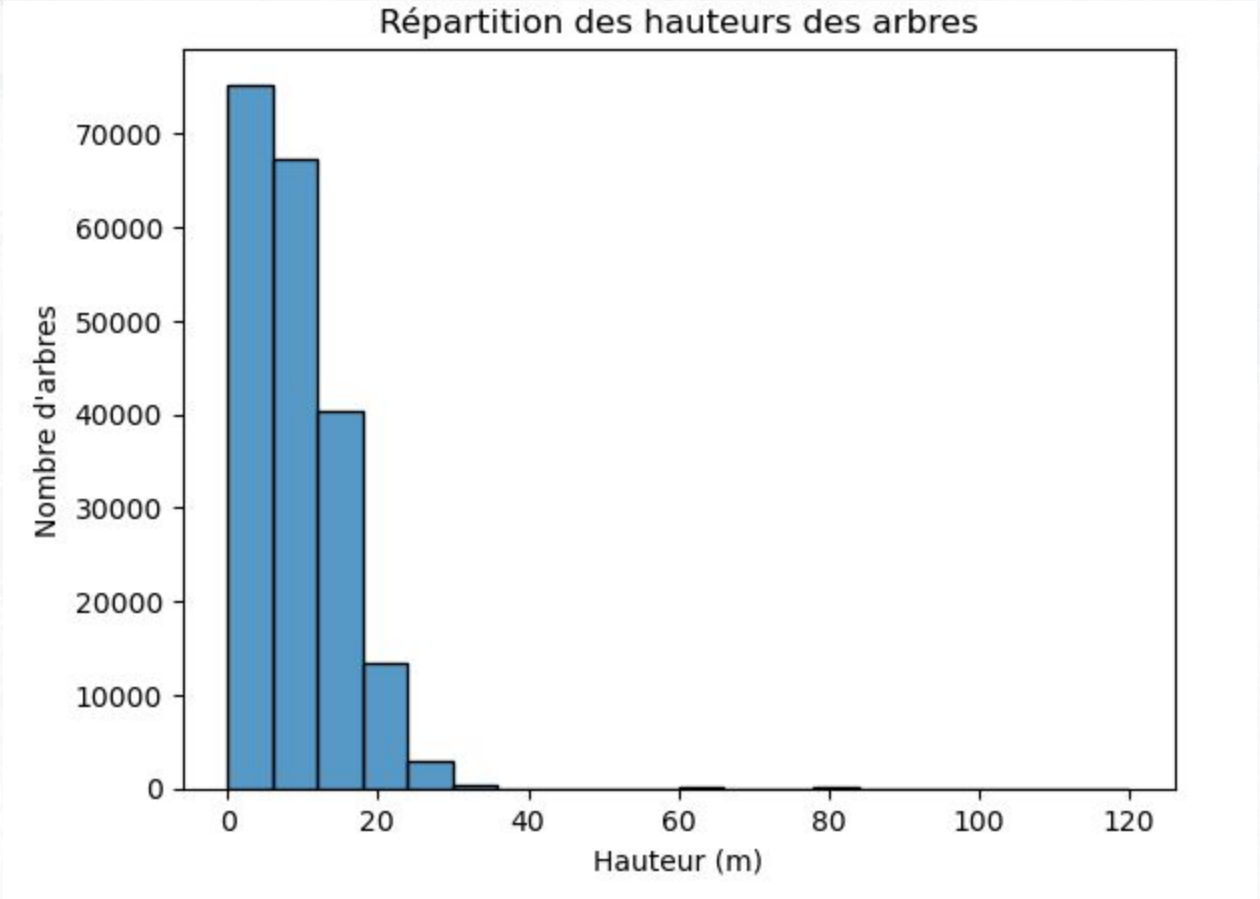
Entrée [21]:
```
# Compter la fréquence de chaque arrondissement
freq_arrondissement = data['arrondissement'].value_counts()

# Créer un DataFrame pour stocker les fréquences des arrondissements
stats_arrondissement = pd.DataFrame({'Arrondissement': freq_arrondissement.index,
                                     'Fréquence': freq_arrondissement.values})

stats_arrondissement
```

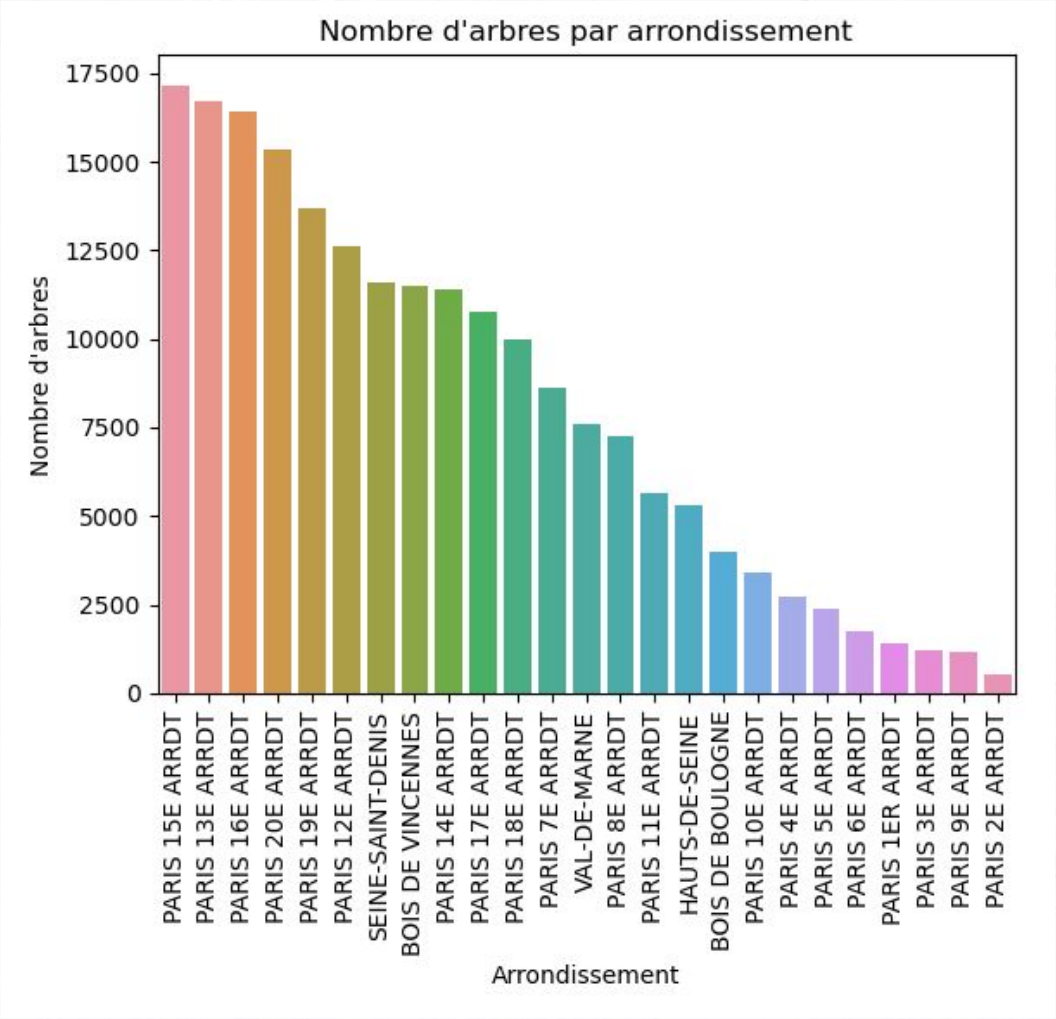
Out[21]:

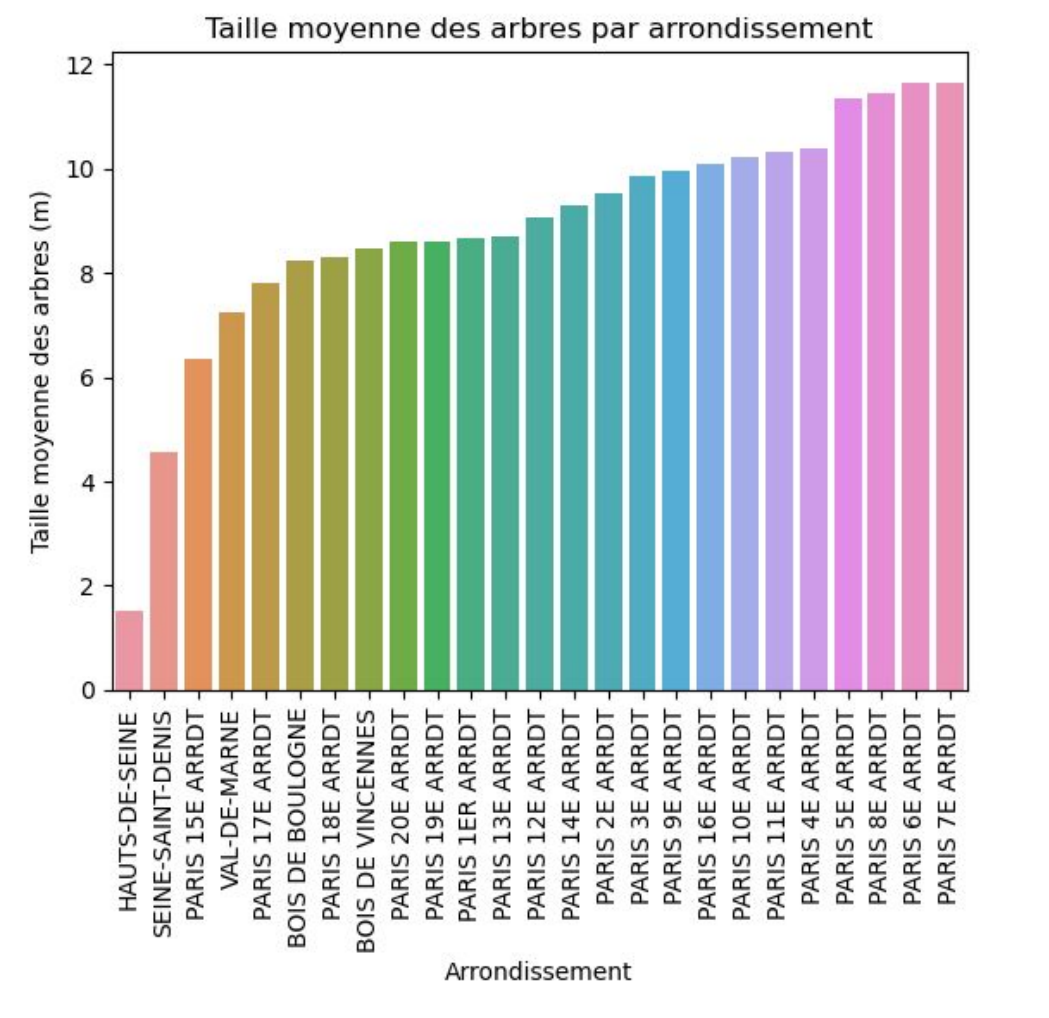
	Arrondissement	Fréquence
0	PARIS 15E ARRDT	17151
1	PARIS 13E ARRDT	16712
2	PARIS 16E ARRDT	16403
3	PARIS 20E ARRDT	15340
4	PARIS 19E ARRDT	13709
5	PARIS 12E ARRDT	12600
6	SEINE-SAINT-DENIS	11570
7	BOIS DE VINCENNES	11510
8	PARIS 14E ARRDT	11399
9	PARIS 17E ARRDT	10762
10	PARIS 18E ARRDT	10011
11	PARIS 7E ARRDT	8617
12	VAL-DE-MARNE	7580
13	PARIS 8E ARRDT	7245
14	PARIS 11E ARRDT	5658
15	HAUTS-DE-SEINE	5298
16	BOIS DE BOULOGNE	3978
17	PARIS 10E ARRDT	3385
18	PARIS 4E ARRDT	2740
19	PARIS 5E ARRDT	2368
20	PARIS 6E ARRDT	1764
21	PARIS 1ER ARRDT	1413
22	PARIS 3E ARRDT	1209
23	PARIS 9E ARRDT	1167
24	PARIS 2E ARRDT	548

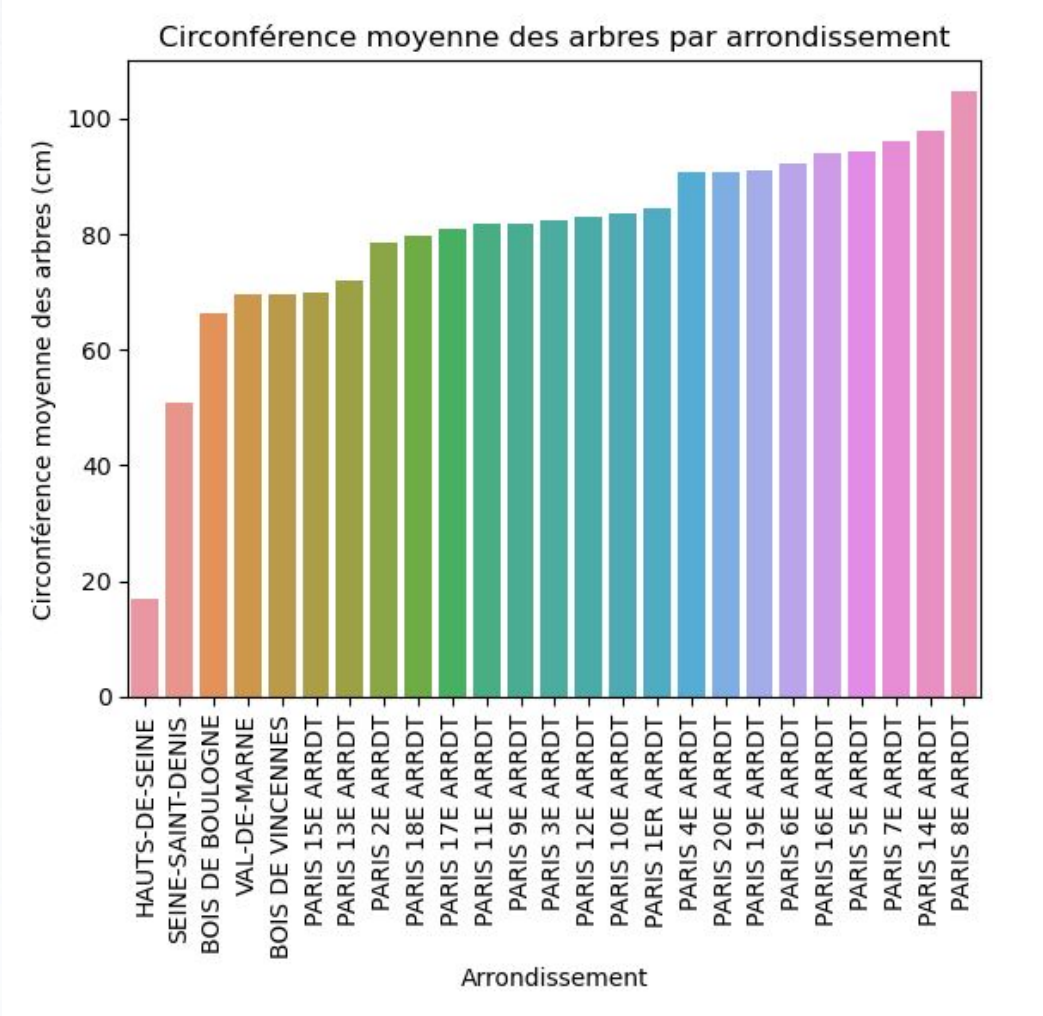

Répartition des circonférences des arbres
Nombre d'arbres
160000
140000
120000
100000
80000
60000
40000
20000
0
0
100
200
300
400
500
Circonférence (cm)

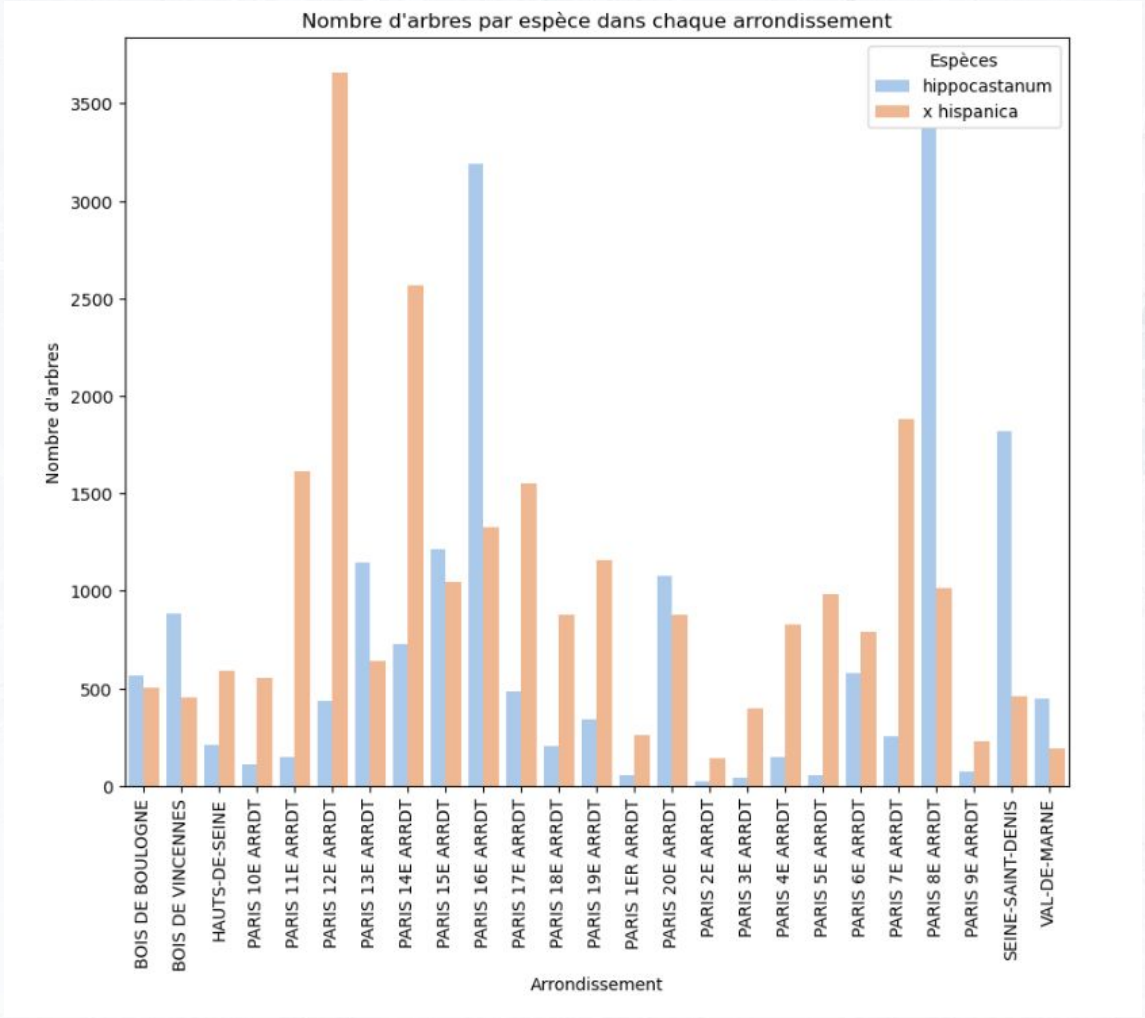

Répartition des circonférences des arbres
Nombre d'arbres
0
10000
20000
30000
40000
0
50
100
150
200
250
300
350
400
450
500
Circonférence (cm)

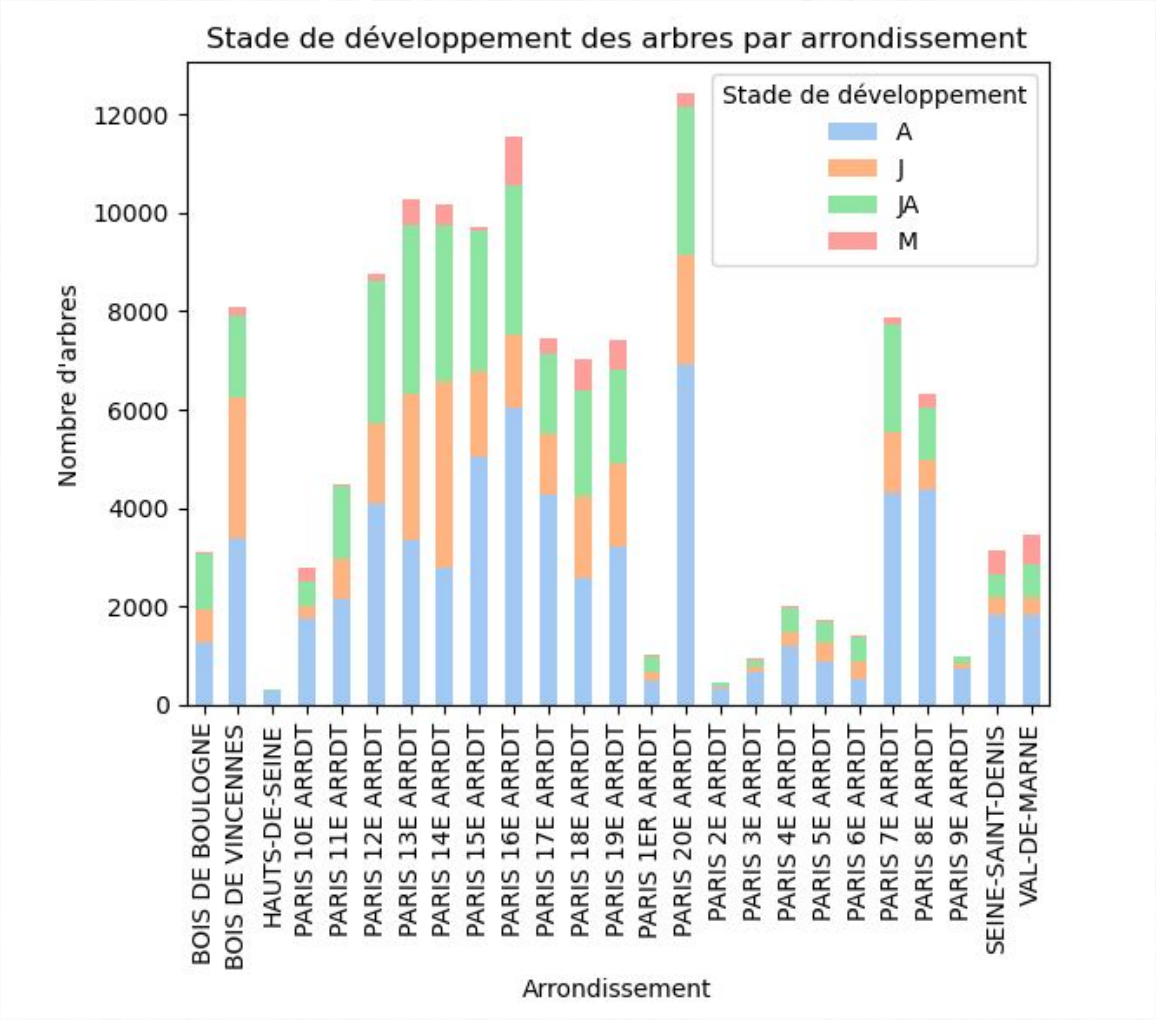

Répartition des hauteurs des arbres
Nombre d'arbres
70000
60000
50000
40000
30000
20000
10000
0
0
20
40
60
80
100
120
Hauteur (m)


Nombre d'arbres par arrondissement
Nombre d'arbres
17500
15000
12500
10000
7500
5000
2500
0
PARIS 15E ARRDT
PARIS 13E ARRDT
PARIS 16E ARRDT
PARIS 20E ARRDT
PARIS 19E ARRDT
PARIS 12E ARRDT
SEINE-SAINT-DENIS
BOIS DE VINCENNES
PARIS 14E ARRDT
PARIS 17E ARRDT
PARIS 18E ARRDT
PARIS 7E ARRDT
VAL-DE-MARNE
PARIS 8E ARRDT
PARIS 11E ARRDT
HAUTS-DE-SEINE
BOIS DE BOULOGNE
PARIS 10E ARRDT
PARIS 4E ARRDT
PARIS 5E ARRDT
PARIS 6E ARRDT
PARIS 1ER ARRDT
PARIS 3E ARRDT
PARIS 9E ARRDT
PARIS 2E ARRDT
Arrondissement


Taille moyenne des arbres par arrondissement
Taille moyenne des arbres (m)
0
2
4
6
8
10
12
HAUTS-DE-SEINE
SEINE-SAINT-DENIS
PARIS 15E ARRDT
VAL-DE-MARNE
PARIS 17E ARRDT
BOIS DE BOULOGNE
PARIS 18E ARRDT
BOIS DE VINCENNES
PARIS 20E ARRDT
PARIS 19E ARRDT
PARIS 1ER ARRDT
PARIS 13E ARRDT
PARIS 12E ARRDT
PARIS 14E ARRDT
PARIS 2E ARRDT
PARIS 3E ARRDT
PARIS 9E ARRDT
PARIS 16E ARRDT
PARIS 10E ARRDT
PARIS 11E ARRDT
PARIS 4E ARRDT
PARIS 5E ARRDT
PARIS 8E ARRDT
PARIS 6E ARRDT
PARIS 7E ARRDT
Arrondissement


Circonférence moyenne des arbres par arrondissement
Circonférence moyenne des arbres (cm)
0
20
40
60
80
100
HAUTS-DE-SEINE
SEINE-SAINT-DENIS
BOIS DE BOULOGNE
VAL-DE-MARNE
BOIS DE VINCENNES
PARIS 15E ARRDT
PARIS 13E ARRDT
PARIS 2E ARRDT
PARIS 18E ARRDT
PARIS 17E ARRDT
PARIS 11E ARRDT
PARIS 9E ARRDT
PARIS 3E ARRDT
PARIS 12E ARRDT
PARIS 10E ARRDT
PARIS 1ER ARRDT
PARIS 4E ARRDT
PARIS 20E ARRDT
PARIS 19E ARRDT
PARIS 6E ARRDT
PARIS 16E ARRDT
PARIS 5E ARRDT
PARIS 7E ARRDT
PARIS 14E ARRDT
PARIS 8E ARRDT
Arrondissement


Nombre d'arbres par espèce dans chaque arrondissement
Espèces
hippocastanum
x hispanica
Nombre d'arbres
0
500
1000
1500
2000
2500
3000
3500
BOIS DE BOULOGNE
BOIS DE VINCENNES
HAUTS-DE-SEINE
PARIS 10E ARRDT
PARIS 11E ARRDT
PARIS 12E ARRDT
PARIS 13E ARRDT
PARIS 14E ARRDT
PARIS 15E ARRDT
PARIS 16E ARRDT
PARIS 17E ARRDT
PARIS 18E ARRDT
PARIS 19E ARRDT
PARIS 1ER ARRDT
PARIS 20E ARRDT
PARIS 2E ARRDT
PARIS 3E ARRDT
PARIS 4E ARRDT
PARIS 5E ARRDT
PARIS 6E ARRDT
PARIS 7E ARRDT
PARIS 8E ARRDT
PARIS 9E ARRDT
SEINE-SAINT-DENIS
VAL-DE-MARNE
Arrondissement


Stade de développement des arbres par arrondissement
Stade de développement
A
J
JA
M
Nombre d'arbres
0
2000
4000
6000
8000
10000
12000
BOIS DE BOULOGNE
BOIS DE VINCENNES
HAUTS-DE-SEINE
PARIS 10E ARRDT
PARIS 11E ARRDT
PARIS 12E ARRDT
PARIS 13E ARRDT
PARIS 14E ARRDT
PARIS 15E ARRDT
PARIS 16E ARRDT
PARIS 17E ARRDT
PARIS 18E ARRDT
PARIS 19E ARRDT
PARIS 1ER ARRDT
PARIS 20E ARRDT
PARIS 2E ARRDT
PARIS 3E ARRDT
PARIS 4E ARRDT
PARIS 5E ARRDT
PARIS 6E ARRDT
PARIS 7E ARRDT
PARIS 8E ARRDT
PARIS 9E ARRDT
SEINE-SAINT-DENIS
VAL-DE-MARNE
Arrondissement

04 →

Synthèse

(IA)

Au terme de cette analyse exploratoire des données, nous avons pu mettre en lumière plusieurs caractéristiques essentielles pour l'optimisation de nos tournées d'entretien des arbres.

En me rapprochants de professionnelles, j'ai pus mettre ne place une approche métier cohérente et ainsi connaitres les features importantes à savoir :

- l'espèce
- la hauteur
- le stade de développement
- la circonférence du tronc
- l'arrondissement

L'espèce de l'arbre est un élément déterminant dans la nature de l'entretien nécessaire. En effet, chaque espèce a des besoins spécifiques en termes de soins et d'entretien. La hauteur de l'arbre et la circonférence de son tronc sont également des indicateurs précieux pour l'élagage, puisqu'ils permettent de déterminer l'équipement nécessaire et le temps que prendra cette intervention.

Le stade de développement de l'arbre nous renseigne sur ses besoins en arrosage : un arbre jeune nécessitera un arrosage plus fréquent qu'un arbre mature. Enfin, l'arrondissement dans lequel se trouve l'arbre est crucial pour la planification des tournées d'entretien. En regroupant les interventions par arrondissement, nous pouvons optimiser nos déplacements et ainsi économiser du temps et des ressources.

Ces caractéristiques seront donc nos points de repère pour la mise en place d'un réseau de neurones.

Afin de choisir le réseau de neurones le plus adéquat pour réaliser notre objectif ainsi que le type d'apprentissage nous avons mené des recherches.

Lors de mes recherches j'ai pus observer certains corelas avec le Problème du voyageur de commerce (TSP - Traveling Salesman Problem) ou Problème de routage de véhicules (VRP - Vehicle Routing Problem)

En étudiant plus en profondeur le problème du voyageur de commerce , j'ai pus observer que l'une des méthodes de résolution de celui- ci était l'algorithme de colonies de fourmis.

Au cours de l'approfondissement de mes investigations, j'ai découvert une publication scientifique qui éclaire pertinemment la problématique à l'étude.

Cette étude utilise des réseaux de neurones en rétro-propagation afin de pouvoir résoudre le problème du voyageur de commerce.

Toutefois, l'application des réseaux de neurones pour résoudre ce problème a également été étudiée.

Par exemple, dans le cas du problème de routage de véhicules (VRP), l'approche hybride qui utilise le réseau de neurones auto-organisateur (SOM) comme une recherche locale dans un algorithme mémétique basé sur la population a été présentée

Mais aussi avec la méthode de l'algorithme génétique.

Cette méthode, inspirée du processus de sélection naturelle, recherche un espace pour une solution approximative à des problèmes avec de multiples solutions.

Un exemple d'application est la recherche de l'itinéraire optimal dans le cadre du problème VRP.

Cependant, il convient de souligner que la résolution de ce problème peut se faire de plusieurs façons et ne nécessite pas nécessairement l'implémentation d'un réseau de neurones.

Parallèlement, des méthodes mathématiques ont été largement utilisées pour résoudre le problème du voyageur de commerce (TSP) et le VRP.

Le TSP, par exemple, a été formulé pour la première fois comme un problème de trouver l'itinéraire le plus court à travers un graphe qui visite chaque nœud exactement une fois.

Plusieurs techniques d'optimisation importantes ont été développées à partir de l'étude du TSP, notamment les méthodes de plans de coupe et les algorithmes de branch and bound.

Dans le cas du VRP, la notion de capacité est directement impliquée, où la demande en ressources fixes des clients d'une part, et le nombre de clients qui peuvent être desservis par un seul véhicule d'autre part, sont contraints par la capacité du véhicule.

Cela a introduit à la fois la composante de routage (similaire au TSP) et la composante d'attribution (les clients doivent être partitionnés en plusieurs routes)

Il conviendra donc d'effectuer une analyse comparative en complément de cette analyse exploratoire, qui prendrait en compte une approche algorithmique pure et une approche avec des modèles de réseaux de neurones.

Sources

- Synthèse du problème de routage de véhicules
- Le problème du voyageur de commerce
- Utilisation des réseaux de neurones pour le tuning des Algorithmes d’optimisation par colonies de fourmis Application aux chaînes logistiques
- Du TSP au VRP dynamique : une application des réseaux de neurones dans la métaheuristique basée sur la population
- Algorithmes génétiques sélectionnés pour la résolution de problèmes de routage de véhicules
- Un examen des approches de modélisation des problèmes de routage de véhicules appliqués
- A review of approaches to modeling applied vehicle routing problems

Merci

Avez-vous des questions ?

GitHub

Cette présentation est en liens avec le noteBook :

“NoteBook_P2_Participez_à_un_concours_sur_la_Smart_City”