smart_city2

May 20, 2024

1 Analyse Initiale et Identification des Valeurs Manquantes

Table of contents

[1]:
```
%pip install seaborn
```

```
Requirement already satisfied: seaborn in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=1.2 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from seaborn) (3.8.4)
Requirement already satisfied: contourpy>=1.0.1 in
```

```
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.51.0)
Requirement already satisfied: kiwisolver>=1.3.1 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.0)
Requirement already satisfied: pillow>=8 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: importlib-resources>=3.2.0 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from matplotlib!=3.6.1,>=3.4->seaborn) (6.4.0)
Requirement already satisfied: pytz>=2020.1 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: zipp>=3.1.0 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from importlib-resources>=3.2.0->matplotlib!=3.6.1,>=3.4->seaborn)
(3.18.1)
Requirement already satisfied: six>=1.5 in
e:\openclassroom\ai_engineer\projet_02\work_forlder_2\env_smart_city\lib\site-
packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[5]:
```
%pip freeze
```

```
asttokens==2.4.1
attrs==23.2.0
beautifulsoup4==4.12.3
bleach==6.1.0
colorama==0.4.6
```

```
comm==0.2.2
contourpy==1.2.1
cycler==0.12.1
debugpy==1.8.1
decorator==5.1.1
defusedxml==0.7.1
exceptiongroup==1.2.1
executing==2.0.1
fastjsonschema==2.19.1
fonttools==4.51.0
importlib_metadata==7.1.0
importlib_resources==6.4.0
ipykernel==6.29.4
ipython==8.18.1
jedi==0.19.1
Jinja2==3.1.4
jsonschema==4.22.0
jsonschema-specifications==2023.12.1
jupyter_client==8.6.1
jupyter_core==5.7.2
jupyterlab_pygments==0.3.0
kiwisolver==1.4.5
MarkupSafe==2.1.5
matplotlib==3.8.4
matplotlib-inline==0.1.7
mistune==3.0.2
nbclient==0.10.0
nbconvert==7.16.4
nbformat==5.10.4
nest-asyncio==1.6.0
numpy==1.26.4
packaging==24.0
pandas==2.2.2
pandocfilters==1.5.1
parso==0.8.4
pillow==10.3.0
platformdirs==4.2.1
prompt-toolkit==3.0.43
psutil==5.9.8
pure-eval==0.2.2
Pygments==2.18.0
pyparsing==3.1.2
python-dateutil==2.9.0.post0
pytz==2024.1
pywin32==306
pyzmq==26.0.3
referencing==0.35.1
rpds-py==0.18.1
```

```
seaborn==0.13.2
six==1.16.0
soupsieve==2.5
stack-data==0.6.3
tinycss2==1.3.0
tornado==6.4
traitlets==5.14.3
typing_extensions==4.11.0
tzdata==2024.1
wcwidth==0.2.13
webencodings==0.5.1
zipp==3.18.1
Note: you may need to restart the kernel to use updated packages.
```

[1]:
```
# Importation des librairies Pandas, Matplotlib et Seaborn
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Lecture du fichier de données au format CSV en spécifiant le séparateur ';'
data = pd.read_csv('Paris_tree_dataset.csv', sep=';')
# Affichage des premières lignes pour vérifier la structure des données
data.head()
```

[1]:

	id	type_emplacement	domanialite	arrondissement	complement_addresse	numero	lieu	id_emplacement	libelle_francais	genre	espece	variete	circonference_cm	hauteur_m	stade_developpement	remarquable	geo_point_2d_a	geo_point_2d_b
0	99874	Arbre	Jardin	PARIS 7E ARRDT	NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	19	Marronnier	Aesculus	hippocastanum	NaN	20	5	NaN	0.0	48.857620	2.320962
1	99875	Arbre	Jardin	PARIS 7E ARRDT	NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	20	If	Taxus	baccata	NaN	65					
2	99876	Arbre	Jardin	PARIS 7E ARRDT	NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	21	If	Taxus	baccata	NaN	90					
3	99877	Arbre	Jardin	PARIS 7E ARRDT	NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	22	Erable	Acer	negundo	NaN	60					
4	99878	Arbre	Jardin	PARIS 17E ARRDT	NaN	NaN	PARC CLICHY-BATIGNOLLES-MARTIN LUTHER KING	000G0037	Arbre à miel	Tetradium	daniellii	NaN	38					

```
1          8                  A          NaN       48.857656        2.321031
2         10                  A          NaN       48.857705        2.321061
3          8                  A          NaN       48.857722        2.321006
4          0                NaN          NaN       48.890435        2.315289
```

1.1 Description des Colonnes du Dataset

- **id** : Identifiant unique de l'arbre dans la base de données.
- **type_emplacement** : Type d'emplacement où l'arbre est planté (ex. Arbre, Jardin).
- **domanialite** : Classification du terrain sur lequel l'arbre est situé (ex. Alignement, Jardin).
- **arrondissement** : Zone géographique où l'arbre est localisé.
- **complement_addresse** : Informations complémentaires sur l'adresse.
- **numero** : Numéro associé à l'adresse.
- **lieu** : Adresse de localisation de l'arbre.
- **id_emplacement** : Code identifiant l'emplacement de l'arbre.
- **libelle_francais** : Nom commun de l'arbre en français.
- **genre** : Classification scientifique du genre de l'arbre.
- **espece** : Espèce de l'arbre.
- **varieteoucultivar** : Variété ou cultivar de l'arbre, si applicable.
- **circonference_cm** : Circonférence de l'arbre mesurée en centimètres.
- **hauteur_m** : Hauteur de l'arbre mesurée en mètres.
- **stade_developpement** : Stade de développement de l'arbre (ex. Jeune, Adulte).
- **remarquable** : Indique si l'arbre est classé comme remarquable (1 pour "OUI", 0 pour "NON").
- **geo_point_2d** : Coordonnées géographiques de l'arbre (longitude et latitude).

[2]:
```
# Obtention des dimensions du data frame
dim = data.shape
print("Nombre de lignes dans le data frame :", dim[0])
print("Nombre de colonnes dans le data frame :", dim[1])
```

```
Nombre de lignes dans le data frame : 200137
Nombre de colonnes dans le data frame : 18
```

[3]:
```
# Informations générales sur les colonnes
data.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
```

[4]:
```
# Statistiques descriptives des variables numériques
data.describe()
```

[4]:

	id	numero	circonference_cm	hauteur_m	remarquable	\
count	2.001370e+05	0.0	200137.000000	200137.000000	137039.000000	
mean	3.872027e+05	NaN	83.380479	13.110509	0.001343	
std	5.456032e+05	NaN	673.190213	1971.217387	0.036618	
min	9.987400e+04	NaN	0.000000	0.000000	0.000000	
25%	1.559270e+05	NaN	30.000000	5.000000	0.000000	
50%	2.210780e+05	NaN	70.000000	8.000000	0.000000	
75%	2.741020e+05	NaN	115.000000	12.000000	0.000000	
max	2.024745e+06	NaN	250255.000000	881818.000000	1.000000	

	geo_point_2d_a	geo_point_2d_b
count	200137.000000	200137.000000
mean	48.854491	2.348208
std	0.030234	0.051220
min	48.742290	2.210241
25%	48.835021	2.307530
50%	48.854162	2.351095
75%	48.876447	2.386838
max	48.911485	2.469759

[5]:
```
# Calcul du nombre et du pourcentage de valeurs manquantes pour chaque colonne
missing_data = data.isnull().sum()
missing_percentage = (data.isnull().mean() * 100).round(2)  # Arrondi à deux␣
 ↪chiffres après la virgule pour la clarté

# Création d'un DataFrame pour une meilleure visualisation
missing_info = pd.DataFrame({'Nombre manquant': missing_data, 'Pourcentage␣
 ↪manquant (%)': missing_percentage})
print(missing_info)
```

	Nombre manquant	Pourcentage manquant (%)

id	0	0.00
type_emplacement	0	0.00
domanialite	1	0.00
arrondissement	0	0.00
complement_addresse	169235	84.56
numero	200137	100.00
lieu	0	0.00
id_emplacement	0	0.00
libelle_francais	1497	0.75
genre	16	0.01
espece	1752	0.88
variete	163360	81.62
circonference_cm	0	0.00
hauteur_m	0	0.00
stade_developpement	67205	33.58
remarquable	63098	31.53
geo_point_2d_a	0	0.00
geo_point_2d_b	0	0.00

1.2 Objectif

L'objectif de cette première étape est de comprendre la structure et les caractéristiques de notre jeu de données en effectuant une analyse exploratoire. Nous commençons par importer les données et vérifier leur intégrité. Ensuite, nous identifions les valeurs manquantes et analysons les distributions statistiques des variables.

1.3 Étapes Réalisées

1. **Importation et Vérification des Données**:
   - Nous avons importé les données du fichier CSV et affiché les premières lignes pour vérifier la structure et la cohérence des données.
2. **Description des Colonnes**:
   - Nous avons détaillé chaque colonne pour comprendre la signification de chaque variable présente dans le dataset.
3. **Analyse des Dimensions**:
   - Nous avons obtenu les dimensions du DataFrame, constatant qu'il contient 200,137 lignes et 18 colonnes.
4. **Informations Générales**:
   - Nous avons utilisé `data.info()` pour afficher des informations générales sur les colonnes, y compris les types de données et le nombre de valeurs non nulles.
5. **Statistiques Descriptives**:
   - Avec `data.describe()`, nous avons calculé des statistiques descriptives pour les variables numériques, telles que la moyenne, la médiane, l'écart-type, les valeurs minimales et maximales.
6. **Identification des Valeurs Manquantes**:
   - Nous avons calculé le nombre et le pourcentage de valeurs manquantes pour chaque colonne, créant un DataFrame `missing_info` pour une visualisation claire.

1.4 Conclusions Tirées

- **Colonnes avec des Valeurs Manquantes Élevées**:
  - Nous avons identifié plusieurs colonnes avec un pourcentage élevé de valeurs manquantes, notamment `numero` (100%), `complement_addresse` (84.56%), et `variete` (81.62%). Ces colonnes ont été considérées pour suppression dans les étapes de nettoyage.
- **Colonnes à Faible Pourcentage de Valeurs Manquantes**:
  - Certaines colonnes, comme `espece` (0.88%) et `libelle_francais` (0.75%), ont un faible pourcentage de valeurs manquantes. Nous avons décidé de les conserver dans le dataset sans imputation pour ne pas altérer les analyses futures.
- **Colonnes Importantes avec des Valeurs Manquantes**:
  - `stade_developpement` et `remarquable` ont respectivement 33.58% et 31.53% de valeurs manquantes. Étant donné leur importance potentielle pour les analyses futures, nous avons décidé d'imputer ces valeurs.

1.5 Justification pour l'Étape Suivante

Les observations et les conclusions tirées de cette analyse initiale nous ont permis d'identifier les colonnes problématiques et de justifier nos décisions de nettoyage. Les prochaines étapes se concentreront sur le nettoyage des données pour s'assurer que notre jeu de données est prêt pour des analyses plus détaillées et précises.

En effectuant une analyse initiale rigoureuse, nous posons les bases pour un travail de qualité qui repose sur des données fiables et bien comprises.

2 Nettoyage des Données

[6]:
```
# Définir des seuils pour identifier les valeurs aberrantes basées sur les␣
 ↪records du monde
circonference_seuil = 3110  # 31.1 mètres convertis en cm
hauteur_seuil = 83.8        # En mètres

# Identification des valeurs aberrantes selon les nouveaux seuils
valeurs_aberrantes_circonference = data[data['circonference_cm'] >␣
 ↪circonference_seuil]
valeurs_aberrantes_hauteur = data[data['hauteur_m'] > hauteur_seuil]

# Méthode Interquartile pour identifier les outliers
Q1_circonference = data['circonference_cm'].quantile(0.25)
Q3_circonference = data['circonference_cm'].quantile(0.75)
IQR_circonference = Q3_circonference - Q1_circonference
borne_inf_circonference = Q1_circonference - 1.5 * IQR_circonference
borne_sup_circonference = Q3_circonference + 1.5 * IQR_circonference
outliers_circonference = data[(data['circonference_cm'] <␣
 ↪borne_inf_circonference) | (data['circonference_cm'] >␣
 ↪borne_sup_circonference)]

Q1_hauteur = data['hauteur_m'].quantile(0.25)
```

```
Q3_hauteur = data['hauteur_m'].quantile(0.75)
IQR_hauteur = Q3_hauteur - Q1_hauteur
borne_inf_hauteur = Q1_hauteur - 1.5 * IQR_hauteur
borne_sup_hauteur = Q3_hauteur + 1.5 * IQR_hauteur
outliers_hauteur = data[(data['hauteur_m'] < borne_inf_hauteur) |␣
 ↪(data['hauteur_m'] > borne_sup_hauteur)]

# Fusion des outliers identifiés par les deux méthodes
outliers_combined = pd.concat([valeurs_aberrantes_circonference,␣
 ↪outliers_circonference]).drop_duplicates()
outliers_combined = pd.concat([outliers_combined, valeurs_aberrantes_hauteur,␣
 ↪outliers_hauteur]).drop_duplicates()

# Suppression des outliers combinés
data_cleaned = data[~data.index.isin(outliers_combined.index)]

# Suppression des valeurs de circonférence et de hauteur égales à zéro
data_cleaned = data_cleaned[(data_cleaned['circonference_cm'] > 0) &␣
 ↪(data_cleaned['hauteur_m'] > 0)]

# Gestion des valeurs manquantes
colonnes_a_supprimer = ['numero', 'complement_addresse', 'variete']
data_cleaned = data_cleaned.drop(columns=colonnes_a_supprimer)
data_cleaned['remarquable'] = data_cleaned['remarquable'].fillna(0)

# Imputation des valeurs manquantes pour 'stade_developpement' par la valeur la␣
 ↪plus fréquente (mode)
mode_stade = data_cleaned['stade_developpement'].mode()[0]
data_cleaned['stade_developpement'] = data_cleaned['stade_developpement'].
 ↪fillna(mode_stade)

# Vérification des modifications
print("Dimensions des données après gestion des valeurs manquantes :",␣
 ↪data_cleaned.shape)
```

```
Dimensions des données après gestion des valeurs manquantes : (154626, 15)
```

2.1 Objectif

L'objectif de cette étape est de préparer notre jeu de données pour une analyse précise et fiable en éliminant les valeurs aberrantes et en gérant les valeurs manquantes.

2.2 Étapes Réalisées

1. **Identification des Valeurs Aberrantes**:
   - **Approche métier** : Nous avons défini des seuils basés sur les records mondiaux pour la circonférence et la hauteur des arbres, en utilisant les caractéristiques du séquoia "Général Sherman" comme référence.

- **Méthode Interquartile (IQR)** : Nous avons utilisé cette méthode statistique pour identifier les valeurs aberrantes basées sur les quartiles de la distribution.

2. **Suppression des Valeurs Aberrantes**:
   - Nous avons combiné les outliers identifiés par les deux méthodes et les avons supprimés du jeu de données.
3. **Suppression des Valeurs Égales à Zéro**:
   - Nous avons également supprimé les valeurs de circonférence et de hauteur égales à zéro, car elles ne sont pas réalistes pour des arbres.
4. **Gestion des Valeurs Manquantes**:
   - **Suppression des Colonnes** : Nous avons supprimé les colonnes avec un pourcentage très élevé de valeurs manquantes (`numero`, `complement_addresse`, `variete`).
   - **Imputation des Valeurs Manquantes** :
     - `remarquable` : Remplacement des valeurs manquantes par 0.
     - `stade_developpement` : Imputation par la valeur la plus fréquente (mode).

2.3 Conclusions Tirées

- **Réduction des Valeurs Aberrantes** :
  - En supprimant les valeurs aberrantes, nous avons amélioré la qualité de notre jeu de données, ce qui permettra d'obtenir des analyses plus précises.
- **Nettoyage des Zéros** :
  - La suppression des valeurs de circonférence et de hauteur égales à zéro a permis d'éliminer des anomalies évidentes.
- **Gestion Efficace des Valeurs Manquantes** :
  - En supprimant les colonnes avec un pourcentage élevé de valeurs manquantes et en imputant les valeurs manquantes pour des colonnes importantes, nous avons assuré que notre jeu de données est complet et utilisable pour des analyses ultérieures.

2.4 Justification pour l'Étape Suivante

Grâce à ce nettoyage rigoureux, nous avons maintenant un jeu de données fiable et prêt pour des analyses plus détaillées. Cette préparation nous permet de passer à l'analyse univariée des variables, en sachant que les données sont exemptes de valeurs aberrantes et que les valeurs manquantes ont été gérées de manière appropriée.

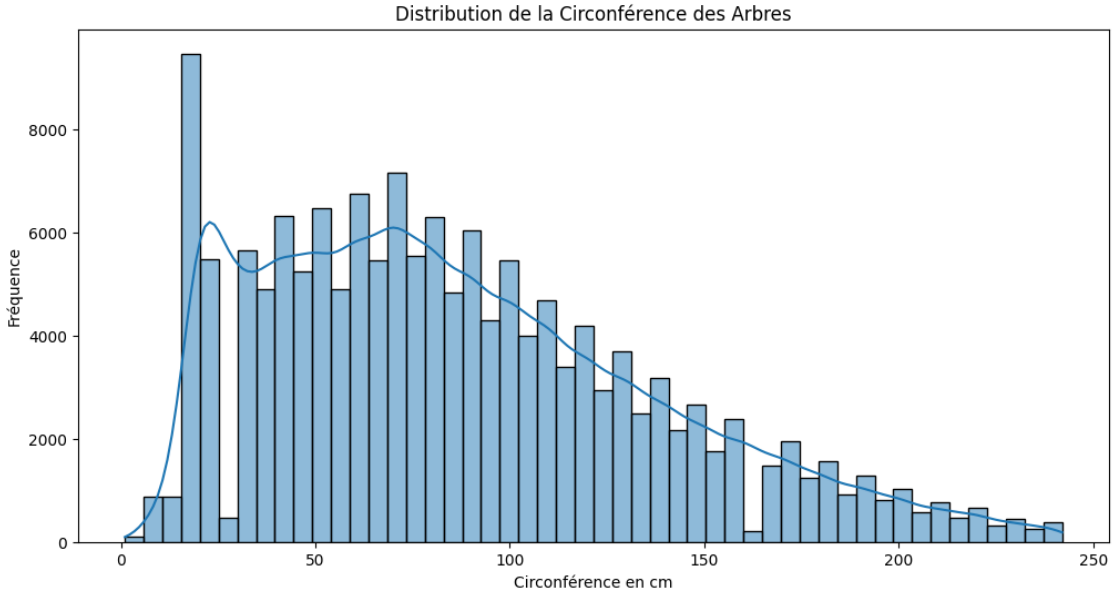
En nettoyant les données de manière exhaustive, nous nous assurons que les conclusions tirées des analyses futures seront précises et pertinentes.

3 Analyse Univariée et Conclusion

[8]:
```
# Calcul de la médiane pour les variables pertinentes
variables_pertinentes = ['circonference_cm', 'hauteur_m']
for col in variables_pertinentes:
    median_value = data_cleaned[col].median()
    print(f"Médiane de {col}: {median_value}")

# Calcul de l'écart-type pour les variables pertinentes
for col in variables_pertinentes:
```

```
    std_dev = data_cleaned[col].std()
    print(f"Écart-type de {col}: {std_dev}")

# Définir la taille des bins
bin_size = 50

# Créer l'histogramme après suppression des doublons
plt.figure(figsize=(12, 6))
sns.histplot(data=data_cleaned, x='circonference_cm', bins=bin_size, kde=True)
plt.title('Distribution de la Circonférence des Arbres')
plt.xlabel('Circonférence en cm')
plt.ylabel('Fréquence')
plt.show()
```

```
Médiane de circonference_cm: 80.0
Médiane de hauteur_m: 10.0
Écart-type de circonference_cm: 50.36696114075837
Écart-type de hauteur_m: 4.575605263329693
```

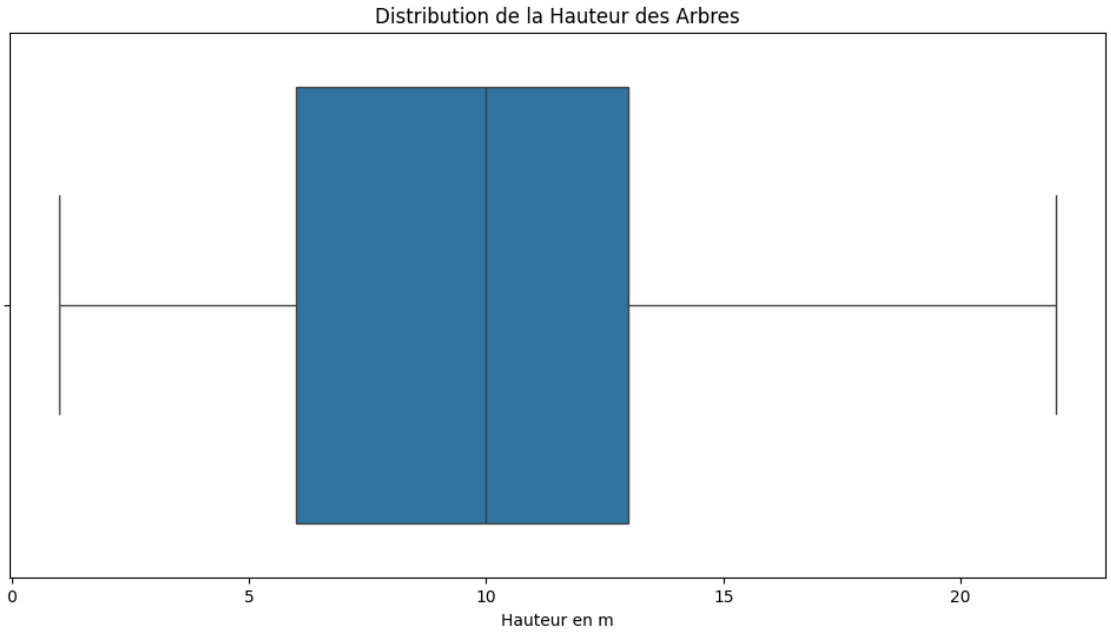


3.1 Observations de la Distribution de la Circonférence des Arbres

- **Concentration autour de 20 cm** : On observe un pic important autour de 20 cm, indiquant que beaucoup d'arbres ont une circonférence proche de cette valeur.
- **Distribution étendue** : La distribution est relativement large, avec une concentration notable autour de 70 à 80 cm.
- **Diminution progressive** : Après 100 cm, la fréquence diminue progressivement, mais il existe encore des arbres avec des circonférences allant jusqu'à 250 cm.

Ce graphique nous aide à visualiser comment les tailles des arbres varient dans l'ensemble des données, avec une tendance centrale mais aussi une diversité significative.

[9]:
```
# Visualisation de la distribution de la hauteur
plt.figure(figsize=(12, 6))
sns.boxplot(x=data_cleaned['hauteur_m'])
plt.title('Distribution de la Hauteur des Arbres')
plt.xlabel('Hauteur en m')
plt.show()
```



3.2 Observations de la Distribution de la Hauteur des Arbres

- **Médiane** : La médiane de la hauteur des arbres est de 10 m, indiquant que 50 % des arbres ont une hauteur inférieure ou égale à 10 m et 50 % ont une hauteur supérieure ou égale à 10 m.
- **Plage interquartile (IQR)** : La majorité des hauteurs des arbres se situe entre 6 m et 12.5 m, ce qui correspond à la plage entre le premier quartile (Q1) et le troisième quartile (Q3).
- **Valeurs extrêmes** : Les valeurs minimales et maximales sans être considérées comme des valeurs aberrantes vont de 1 m à 22 m.

Ce boxplot illustre bien la répartition des hauteurs des arbres, montrant une concentration autour de la médiane et quelques valeurs plus extrêmes.

3.3 Objectif

L'objectif de cette étape est de comprendre la distribution des variables clés après le nettoyage des données. Nous nous concentrons sur les mesures de tendance centrale (médiane) et de dispersion

(écart-type), ainsi que sur la visualisation des distributions de la circonférence et de la hauteur des arbres.

3.4 Étapes Réalisées

1. **Calcul de la Médiane et de l'Écart-Type**:
   - Nous avons calculé la médiane pour les variables `circonference_cm` et `hauteur_m`, ce qui nous donne une mesure de la tendance centrale des données.
   - L'écart-type a également été calculé pour ces variables, fournissant une mesure de la dispersion des données autour de la moyenne.
2. **Visualisation des Distributions**:
   - **Circonférence** : Nous avons utilisé un histogramme avec une courbe de densité pour visualiser la distribution de la circonférence des arbres. Cela nous a permis de voir comment les valeurs sont réparties après le nettoyage.
   - **Hauteur** : Un boxplot a été utilisé pour visualiser la distribution de la hauteur des arbres, ce qui a permis de mettre en évidence les éventuelles valeurs extrêmes et la dispersion des données.

3.5 Conclusions Tirées

- **Circonférence des Arbres** :
  - La médiane de la circonférence est de 80 cm, avec un écart-type de 50.37 cm. L'histogramme montre une distribution relativement étendue avec une concentration autour de la médiane.
- **Hauteur des Arbres** :
  - La médiane de la hauteur est de 10 m, avec un écart-type de 4.58 m. Le boxplot montre que la majorité des arbres ont une hauteur qui se situe autour de cette médiane, avec quelques valeurs extrêmes.

3.6 Résumé et Conclusion

Ce projet a suivi une série d'étapes méthodiques pour analyser les données des arbres de Paris :

1. **Analyse Initiale et Identification des Valeurs Manquantes** :
   - Nous avons importé et vérifié les données, décrit les colonnes, et identifié les valeurs manquantes.
2. **Nettoyage des Données** :
   - Nous avons géré les valeurs aberrantes et les valeurs manquantes pour obtenir un dataset propre et fiable.
3. **Analyse Univariée** :
   - Nous avons effectué une analyse détaillée des variables clés après le nettoyage, calculé la médiane et l'écart-type, et visualisé les distributions.

3.7 Perspectives pour des Analyses Futures

Grâce à ce travail de nettoyage et d'analyse univariée, nous disposons désormais d'un jeu de données propre et fiable, prêt pour des analyses plus approfondies. Les prochaines étapes pourraient inclure des analyses bivariées pour explorer les relations entre différentes variables, ainsi que des modélisations pour prédire des caractéristiques spécifiques des arbres.

En conclusion, ce projet a démontré l'importance d'un nettoyage rigoureux des données et d'une analyse initiale approfondie pour garantir des résultats fiables et pertinents. Les outils et méthodes utilisés ici peuvent être appliqués à d'autres jeux de données pour obtenir des insights précieux et prendre des décisions informées.