L'objectif de ce projet est d'estimer les valeurs foncières des biens qui se sont vendus 2022, en utilisant des algorithmes d'apprentissages supervisées incluant des modèles de classification et de régression. Dans un premier temps il a fallu élaborer un modèle de classification afin de prédire le mieux possibles les types de locaux (manquants) vendus en 2022. Dans un second temps, il a fallu construire un modèle de régression pour prédire la valeur foncières des biens vendus en 2022. Ce rapport a pour but d'expliciter notre démarche d'analyse: exploration des donnnées, transformation des données, construction et interprétation des modèles.

Exploration des données et statistiques descriptives

Lorsqu'on a concaténer l'ensemble des données de 2018 à 2021, nous avons rajouter une nouvelle variable qui renseigne le nombre d'habitants pour chaque département, que l'on appelera "total". Données issus de l'Insee: https://www.insee.fr/fr/statistiques/2012713#tableau-TCRD_004_tab1_regions2016.
Puis nous avons supprimé toutes les ventes qui présentaient des doublons à la même date et la même adresse. Enfin, nous avons supprimés toutes les ventes dont la valeur foncière étaient inférieur ou égal à 1€.
Nous disposons à présent de 46 variables de type numériques (int et float) et de type chaîne de caractère (object) et 3 676 922 observations:

Out[38]:

	Identifiant de document	Reference document	1 Articles CGI	2 Articles CGI	3 Articles CGI	4 Articles CGI	5 Articles CGI	No disposition	Da mutati
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	10/01/20
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2	12/01/20
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	04/01/20
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	11/01/20
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1	17/01/20

5 rows × 46 columns

Regardons le taux de données manquantes dans les données:

Out[23]:

Identifiant de document	100.000000
Reference document	100.000000
1 Articles CGI	100.000000
2 Articles CGI	100.000000
3 Articles CGI	100.000000
4 Articles CGI	100.000000
5 Articles CGI	100.000000
No disposition	0.000000
Date mutation	0.000000
Nature mutation	0.000000
Valeur fonciere	0.000000
No voie	33.004181
B/T/Q	95.620495
Type de voie	33.916085
Code voie	0.000490
Voie	0.002393
Code postal	0.005521
Commune	0.000000
Code departement	0.000000
Code commune	0.000000
Prefixe de section	95.644210
Section	0.003481
No plan	0.000000
No Volume	99.814573
1er lot	72.592076
Surface Carrez du 1er lot	87.744423
2eme lot	90.190164
Surface Carrez du 2eme lot	96.857915
3eme lot	98.130284
Surface Carrez du 3eme lot	99.637958
4eme lot	99.382065
Surface Carrez du 4eme lot	99.910550
5eme lot	99.717862
Surface Carrez du 5eme lot	99.966303
Nombre de lots	0.000000
Code type local	36.479343
Type local	36.479343
Identifiant local	100.000000
Surface reelle bati	36.544914
Nombre pieces principales	36.544914
Nature culture	27.910274
Nature culture speciale	97.169780
Surface terrain	27.910274
code_departement	1.047588
departement	1.047588
total	1.047588

dtype: float64

Nous constatons que beaucoup de variables ont un taux de données manquantes de plus de 90%. Nous décidons de supprimer toutes les variables qui ont un taux de données manquantes supérieur à 65%.

Out[41]:

	No disposition	Date mutation	Nature mutation	Valeur fonciere	No voie	Type de voie	Code voie	Voie	Code postal	Co
0	2	10/01/2018	Vente	3150.0	NaN	NaN	B077	PONT D AIN	1160.0	
1	2	12/01/2018	Vente	2100.0	NaN	NaN	B135	SOUS LE BOIS GIROUD	1250.0	JAS
2	1	04/01/2018	Vente	67000.0	12.0	ALL	3044	DE LA PETITE REYSSOUZE	1000.0	B E
3	1	11/01/2018	Vente	76200.0	5.0	RUE	2690	MOLIERE	1000.0	B E
4	1	17/01/2018	Vente	1000.0	NaN	NaN	B112	VACAGNOLE	1340.0	AT

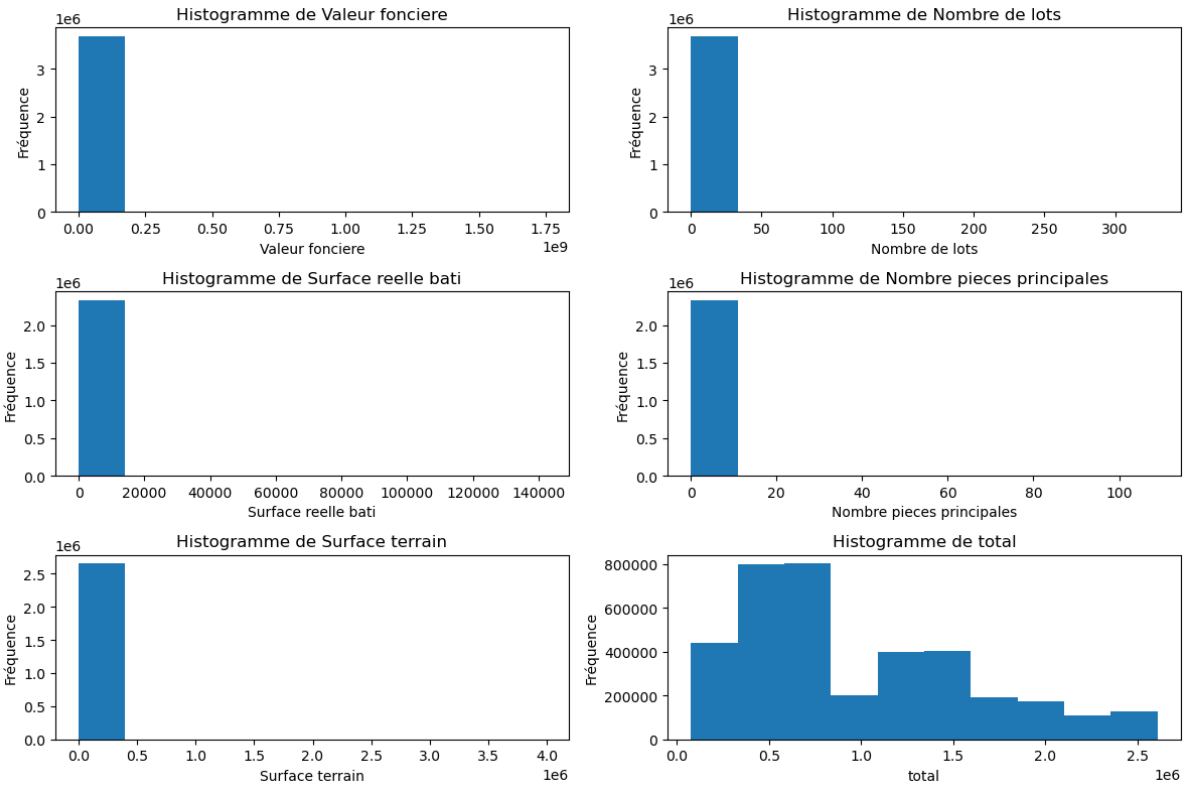
5 rows × 24 columns

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3676922 entries, 0 to 3676921
Data columns (total 24 columns):
 #   Column                      Dtype
---  ------                      -----
 0   No disposition              int64
 1   Date mutation               object
 2   Nature mutation             object
 3   Valeur fonciere             float64
 4   No voie                     float64
 5   Type de voie                object
 6   Code voie                   object
 7   Voie                        object
 8   Code postal                 object
 9   Commune                     object
 10  Code departement            object
 11  Code commune                object
 12  Section                     object
 13  No plan                     int64
 14  Nombre de lots              int64
 15  Code type local             object
 16  Type local                  object
 17  Surface reelle bati         float64
 18  Nombre pieces principales   float64
 19  Nature culture              object
 20  Surface terrain             float64
 21  code_departement            object
 22  departement                 object
 23  total                       float64
dtypes: float64(6), int64(3), object(15)
memory usage: 673.3+ MB
```

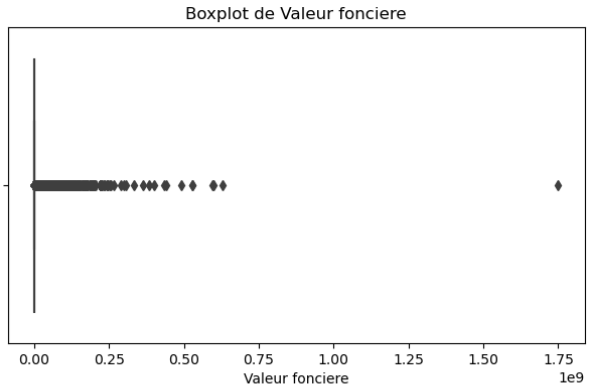
Analyse exploratoire et distribution des variables numériques d'intérêts: Nous allons commencer par l'analyse de la distribution des variables quantitatives qui nous intéressent.

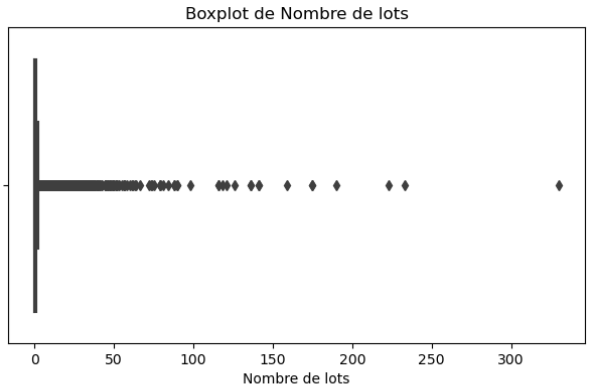
Out[47]:

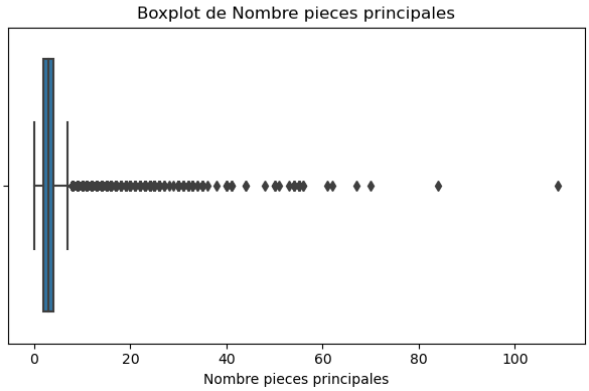
	No disposition	Valeur fonciere	No voie	No plan	Nombre de lots	Surface reelle bati	Nombre pieces principales	Surface terrain	total
0	2	3150.0	NaN	461	0	NaN	NaN	126.0	647634.0
1	2	2100.0	NaN	2066	0	NaN	NaN	197.0	647634.0
2	1	67000.0	12.0	227	1	45.0	1.0	NaN	647634.0
3	1	76200.0	5.0	152	2	68.0	3.0	NaN	647634.0
4	1	1000.0	NaN	106	0	NaN	NaN	5093.0	647634.0



Nous devons vérifier la présence des outliers dans les variables quantitatives, que nous allons rectifier par la suite







Nous observons la présence de nombreux outliers, à l'extérieur de la boîte (qui représente l'intervalle interquartile), au- delà des valeurs maximum représentées bar la barre verticale à l'extrémité de la boîte. Nous devrons rectifier la présence des ces outliers par la suite, car ils risquent de biaiser nos modèles.

Nous transformons la valeur fonciere en log et nous observons la distribution de cette variable.

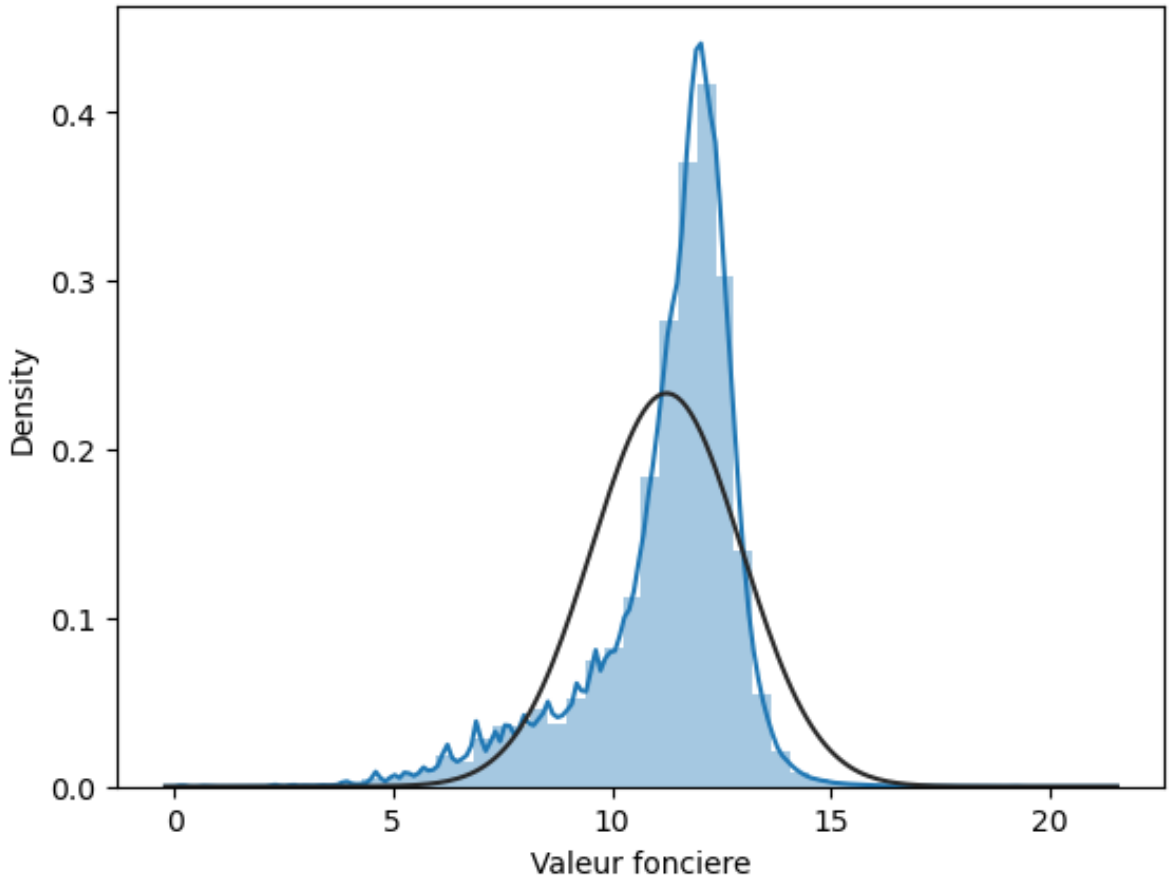
```
/var/folders/sj/kc5dv8dd4_18gtz7tjx528p40000gn/T/ipykernel_1956/20
16357502.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn
v0.14.0.

Please adapt your code to use either `displot` (a figure-level fun
ction with
similar flexibility) or `histplot` (an axes-level function for his
tograms).

For a guide to updating your code to use the new functions, please
see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
(https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(y, fit=norm);
```

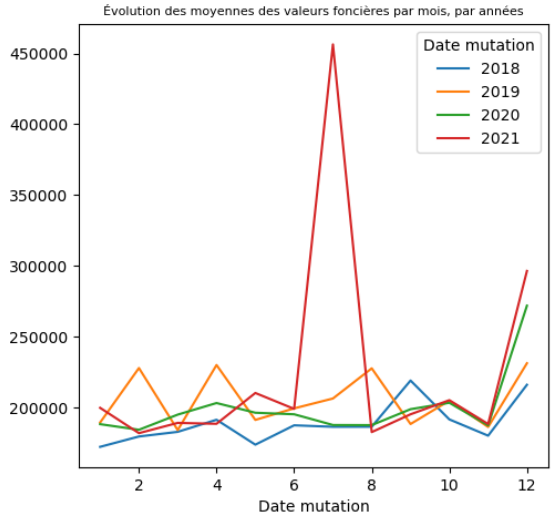


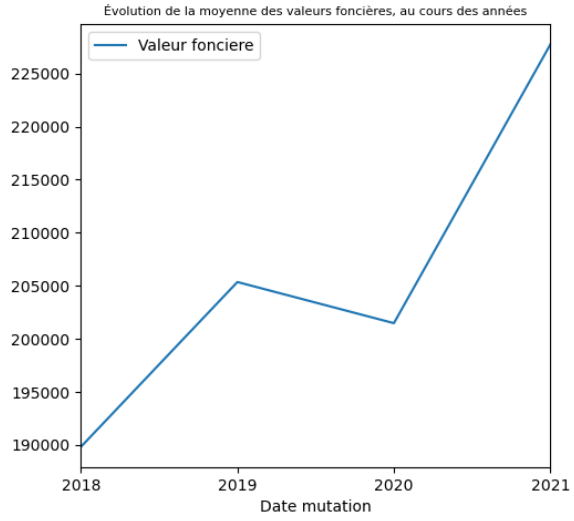

```
<Figure size 640x480 with 0 Axes>
```

Nous observons une hétérogénéité dans les valeurs foncières: on observe une distribution décalée à droite de la médiane et une queue de distribution décalée vers la gauche: l'hétérogénité des valeurs foncières peut s'expliquer par les différents types de biens vendus (appartement, maison, local et dépendance) qui n'ont pas les mêmes valeurs immobilières.

Nous voulons observer l'évolution des valeurs foncières en fonction du temps:

```
/var/folders/sj/kc5dv8dd4_18gtz7tjx528p40000gn/T/ipykernel_16692/1
351706847.py:1: UserWarning: Parsing dates in DD/MM/YYYY format wh
en dayfirst=False (the default) was specified. This may lead to in
consistently parsed dates! Specify a format to ensure consistent p
arsing.
  df['Date mutation'] = pd.to_datetime(df['Date mutation'])
```



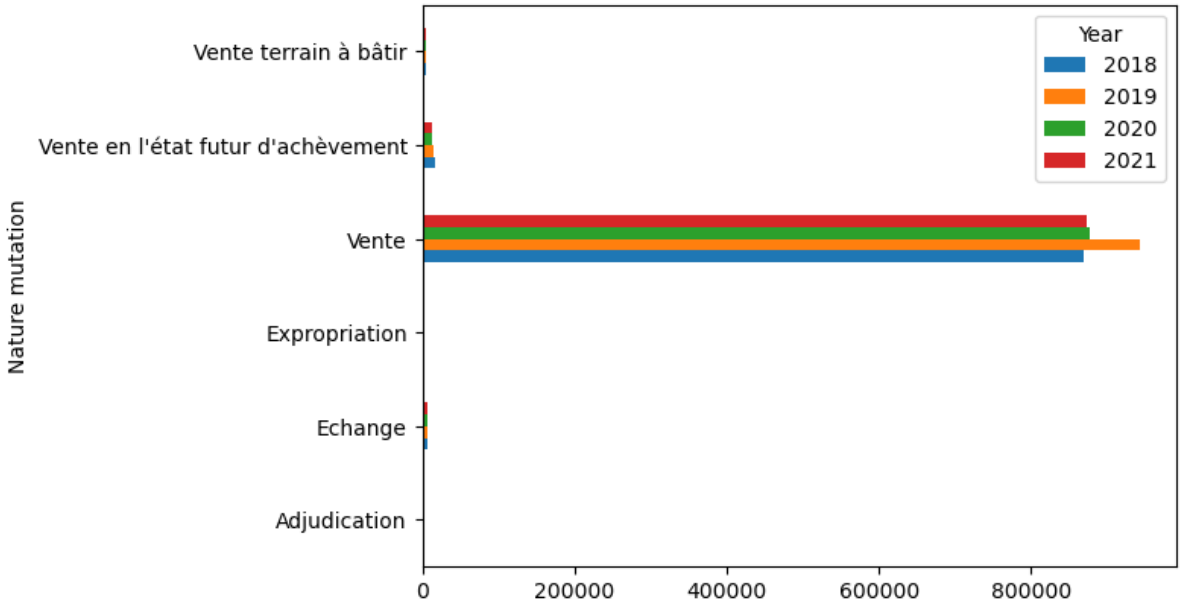


Le plot de droite représente l'évolution de la moyenne des valeurs foncières par mois, pour chaque année de 2018 à 2021. On remarque un pic important de valeurs foncières élevées au mois de juillet pour l'année 2021.
D'après le plot de droite, on remarque que la moyenne des valeurs foncières, tout types de biens confondus, augmentent au cours des années. Une légère diminution de la moyenne des valeurs foncières s'observent pour 2020, peut-être en raison des confinements en période de COVID.

Analyse exploratoire et répartition des variables catégorielles d'intérêts:
Nous allons maintenant analyser la répartition des variables qualitatives qui nous intéressent

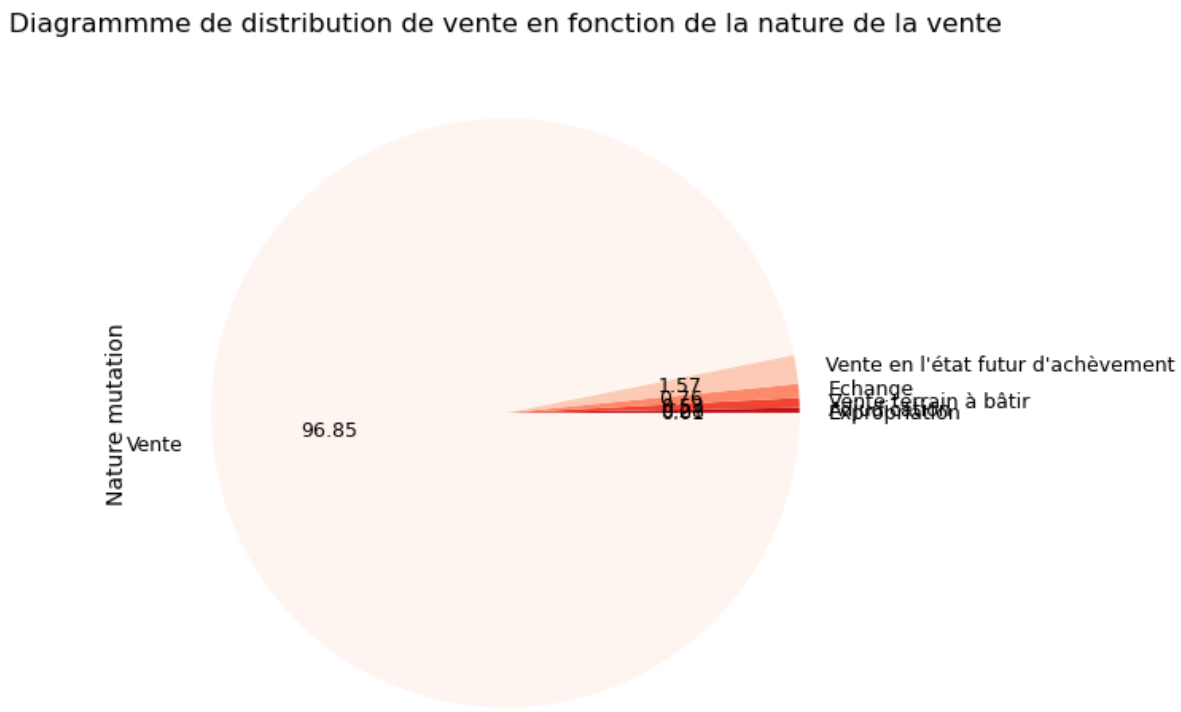
On sintéresse à présent au type de ventes, renseigné par la variable "Nature mutation":

Out[102]: Text(0.5, 1.0, 'Nombre de ventes par année en fonction de la natur
e de la vente')

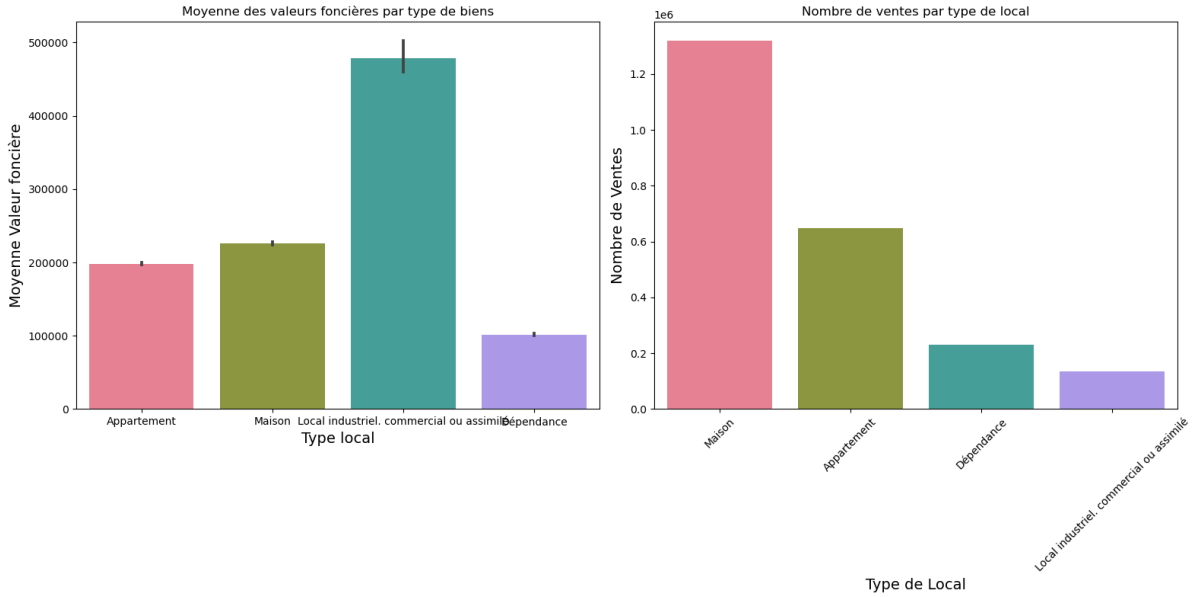


On remarque que la grande majorité des ventes de 2018 à 2021 soit de nature "Vente". Cela se confirme d'après le graphique ci-dessous: 96.85% des ventes, toutes années confondues de 2018 à 2021, sont des "ventes". Lors du pré-traitement des données, nous conserverons uniquement les ventes de type "Vente" et nous excluerons le reste.

Out[103]: <Axes: title={'center': 'Diagrammme de distribution de vente en fo
nction de la nature de la vente'}, ylabel='Nature mutation'>

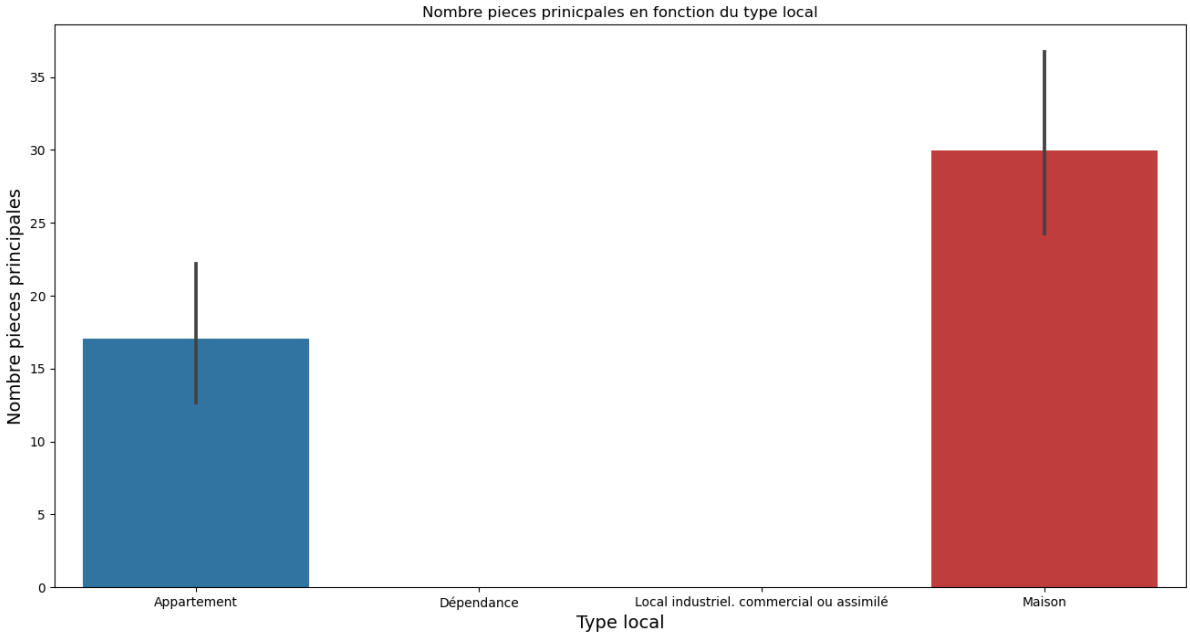


Nous nous intéressons maintenant au type de local vendu: ceux-ci sont renseignés dans la variable "Type local" et sont de 4 modalités "Appartement"; "Maison"; "Local industriel.commercial ou assimilé" ; "Dépendance":

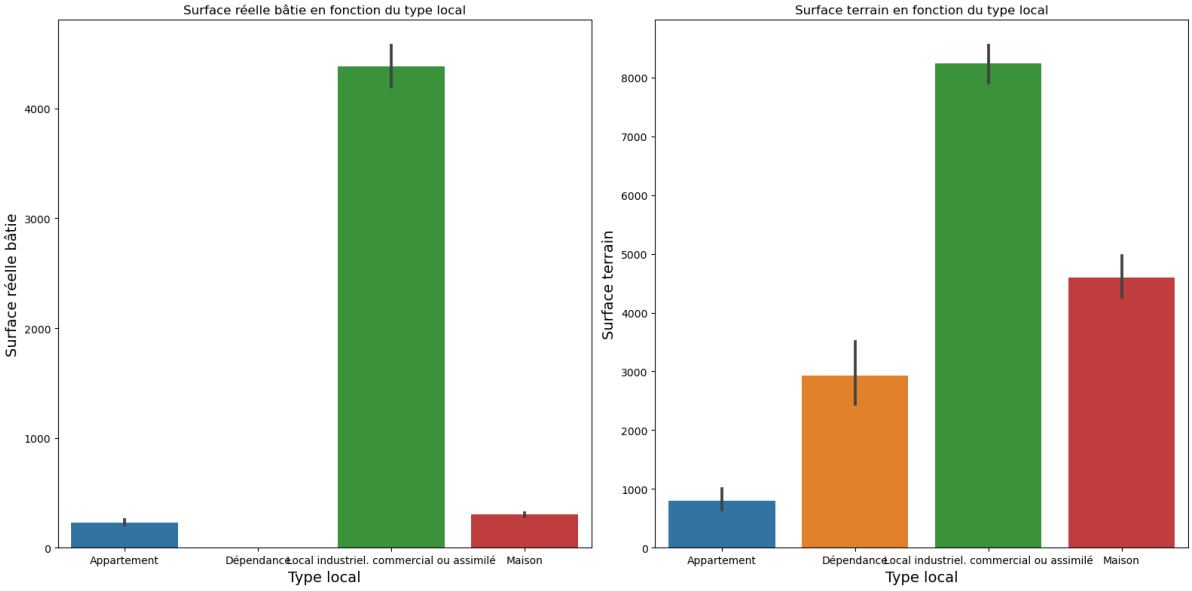


D'après le graphique de droite, on remarque que la majorité des ventes concernent les Maisons et Appartements, puis plus faiblement les Dépendances et les locaux.
Cependant le graphique de gauche montre que bien que les Locaux représentent le type de bien le moins vendus, ils représentent le type de bien vendus le plus cher.
Donc les locaux industriels ou commerciaux auront tendance à avoir des valeurs foncières élevés, ce qui seraient la cause de l'hétérogénéité des la distribution des "valeurs foncieres" montrés précédemment.

Nous nous intéressons à présent aux relations entre la variable "Type local" et les variables "Nombre pieces principales"; "Surface réelle bati" et "Surface terrain":



D'après ce graphe, on constate que le nombre de pièces principales ne concernent pas les dépendances (indiqué dans le dictionnaire es donées) ni les locaux. Il va falloir le prendre en compte dans nos modèles de classification du type local.



Afin d'évaluer la surface des biens vendus en fonction des différents types de locaux, deux variables sont à notre disopsition: "Surface réelle bati" et "Surface terrain".
Ces graphiques renseignent le nombre de "surface terrain" et "surface reelle bati" renseignée pour chaque type de local.
D'après le graphique de gauche, nous constatons la "surface réelle bati" n'est pas renseignée pour les dépédance (indiqué dans le dictionnaire des données).
La surface des dépendances sont renseignées dans "surface terrain"uniquement.
La surface des types "locaux" peut être à la fois reseigné dans "surface réelle bati" et "surface terrain".
Il faudra prendre en compte ces observations lors de la construction de nos modèles.

Pré-traitement des données

Le dataframe initial va subir un ensemble d'étapes de pré-traitement:
- Suppression des variables contenant plus de 65% de valeurs manquantes
- Suppresion de doublons de ventes apparaissant le même jour à la même adresse
- Suppression des ventes dont la "Nature mutation" ne sont pas des "Ventes"
- Suppression des ventes dont la valeur foncière est < ou égale à 1
- Suppression des ventes dont les type locaux ont des données manquantes
- Suppression des ventes dont le code département ont des données manquantes
- Suppression de variables qui ne nous intéressent pas pour la construction des modèles
- Convertion de "Date mutation" en datetime
- Ajout de la colonne "total" comme expliquer précédemment

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2273401 entries, 2 to 3676921
Data columns (total 15 columns):
 #   Column                      Dtype
---  ------                      -----
 0   Date mutation               datetime64[ns]
 1   Nature mutation             object
 2   Valeur fonciere             float64
 3   Code postal                 object
 4   Code departement            object
 5   Code commune                object
 6   Nombre de lots              int64
 7   Code type local             object
 8   Type local                  object
 9   Surface reelle bati         float64
 10  Nombre pieces principales   float64
 11  Surface terrain             float64
 12  code_departement            object
 13  departement                 object
 14  total                       float64
dtypes: datetime64[ns](1), float64(5), int64(1), object(8)
memory usage: 277.5+ MB
```

Out[126]:

	Date mutation	Nature mutation	Valeur fonciere	Code postal	Code departement	Code commune	Nombre de lots	Code type local	Type lo
2	2018-04-01	Vente	67000.0	1000.0	01	53	1	2.0	Apparten
3	2018-11-01	Vente	76200.0	1000.0	01	53	2	2.0	Apparten
5	2018-12-01	Vente	130000.0	1160.0	01	430	0	1.0	Mai
6	2018-04-01	Vente	164370.0	1290.0	01	123	0	1.0	Mai
7	2018-01-15	Vente	97000.0	1750.0	01	370	1	2.0	Apparten

Ce daset est le dataset initial obtenu après toutes les étapes de pré-traitement.
On va maintenant faire 2 copies de ce dataset en:
- "df_classif" que l'on va utliser pour réaliser le modèle de classification.
- "df_reg" que l'on va utiliser pour construire les modèles de régression.

Construction du modèle de classification

```
On décide de ne garder que les variables qui nous intéressent pour
construire le modèle de classification.
Puis on enlèvent les outliers de ces variables.
On remplace les données manquantes des variables quantitatives par
leur médiane.
Et on visualise les corrélations de ces variables avec le "code
type local".
```

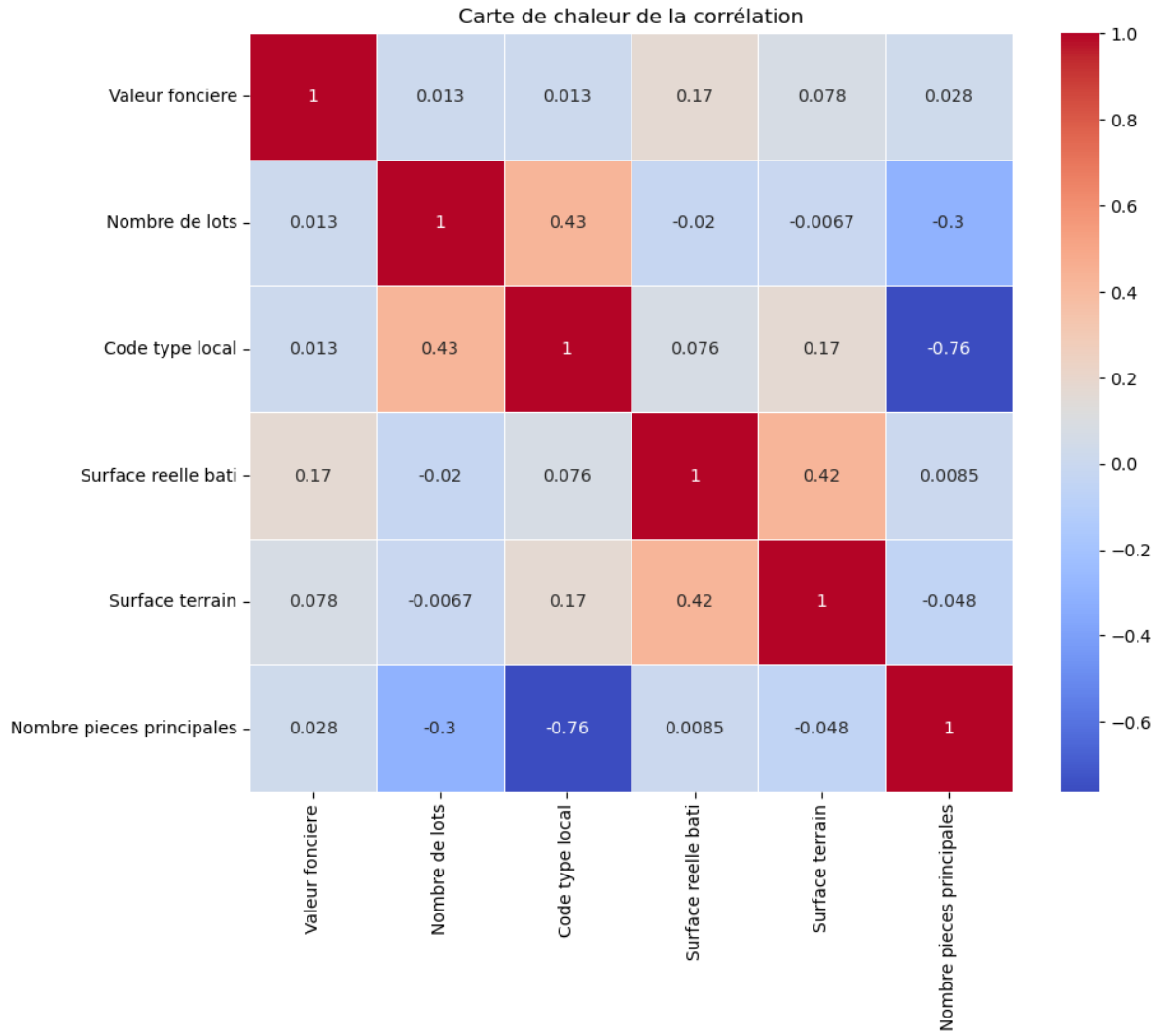
Out[191]:

	Valeur fonciere	Nombre de lots	Code type local	Surface reelle bati	Surface terrain	Nombre pieces principales	Type local
2	67000.0	1	2.0	45.0	NaN	1.0	Appartement
3	76200.0	2	2.0	68.0	NaN	3.0	Appartement
5	130000.0	0	1.0	80.0	55.0	3.0	Maison
6	164370.0	0	1.0	88.0	419.0	4.0	Maison
7	97000.0	1	2.0	90.0	NaN	4.0	Appartement

```
/var/folders/sj/kc5dv8dd4_18gtz7tjx528p40000gn/T/ipykernel_1956/20
39808800.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pa
ndas-docs/stable/user_guide/indexing.html#returning-a-view-versus-
a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/in
dexing.html#returning-a-view-versus-a-copy)
  df_classif1['Code type local']= df_classif1['Code type local'].a
stype(float)
```

```
/var/folders/sj/kc5dv8dd4_18gtz7tjx528p40000gn/T/ipykernel_1956/3740178429.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  matrice_correlation=df_classif1.corr()
```



Bien que les coefficients de corrélations de soient pas satisafaisants, nous garderons les variables "Nombre pieces principales" et "Nombre de lots" comme variables explicatives de notre modèle de régression pour prédire le type local.

Out[200]:

	Nombre de lots	Nombre pieces principales
count	2.273401e+06	2.271068e+06
mean	6.264803e-01	3.080503e+00
std	1.088475e+00	1.929847e+00
min	0.000000e+00	0.000000e+00
25%	0.000000e+00	2.000000e+00
50%	0.000000e+00	3.000000e+00
75%	1.000000e+00	4.000000e+00
max	3.300000e+02	1.090000e+02

Ici nos deux variables expicatives sont à la même échelle, il n'y a pas besoin de normaliser

```
[[184147      0     16  14175]
 [   188  55946  13300     10]
 [    70  11046  24109    262]
 [  3380      1    673 374698]]
                                        precision    recall  f1-
score   support

                              Appartement       0.98      0.93
0.95    198338
                               Dépendance       0.84      0.81
0.82     69444
Local industriel. commercial ou assimilé       0.63      0.68
0.66     35487
                                  Maison       0.96      0.99
0.98    378752

                                accuracy
0.94    682021
                               macro avg       0.85      0.85
0.85    682021
                            weighted avg       0.94      0.94
0.94    682021
```

Nous avons testé plusieurs modèles de classification: modèle des KNN, modèle Random forest et le modèles de l'arbre Decision Tree Classifier. Nous avons conservé ce dernier car il montrait de meilleures métriques que les deux premiers.
Puis, nous avons réaliser un GridSearch CV sur ce modèle de Decision Tree Classifier et nous avons conservé les meilleurs paramètres.
Le modèle de Decision Tree Classifier paramétré avec ces meilleurs paramètres montrent une accuracy de 0.94.
Cela signifie que la proportion de prédictions correctes est de 94% parmi l'ensemble de l'échantillon test. Cette accuracy étant proche de 1 montre que le modèle arrive à classé correctement la majorité des échantillons de test.
On remarque également que ce modèle à tendance à moins bien prédire correctement le type local "local" que les autres types: d'après le rappel, il arrive à prédire correctement le type local "Local" que dans 68% des cas, ce qui reste correct.

Construction du modèle de régression

À partir de ce dataset initial qui a subit les étapes de pré-traitement évoquées précédemment:
On a construit une nouvelle variable "Surface", qui prend en valeur soit "Surface réelle bati" si elle est mentionnée (>0), soit "Surface terrain" si >0 et si "Surface réelle bati" n'est pas mentionnée. Si une vente a une valeur à la fois dans "Surface réelle bati" et dans "Surface terrain", alors la valeur prendra celle de "Surface réelle bati". Si une vente n'a aucune valeur sur sa surface (ni réelle ni bati) alors la valeur retournée est "NA".

Out[56]:

	Date mutation	Nature mutation	Valeur fonciere	Code postal	Code departement	Code commune	Nombre de lots	Code type local	Type lo
2	2018-04-01	Vente	67000.0	1000.0	01	53	1	2.0	Apparten
3	2018-11-01	Vente	76200.0	1000.0	01	53	2	2.0	Apparten
5	2018-12-01	Vente	130000.0	1160.0	01	430	0	1.0	Mai
6	2018-04-01	Vente	164370.0	1290.0	01	123	0	1.0	Mai
7	2018-01-15	Vente	97000.0	1750.0	01	370	1	2.0	Apparten

On crée maintenant une nouvelle colonne "metre_carre": on a calculé un prix au m^2 pour chaque vente à partir de sa "Valeur foncière" / sa "Surface", puis on a fait la moyenne de ce prix au m^2 pour chaque vente partageant le même code département: on se retrouve donc avec une variable "metre_carre" dont la valeur est la même pour chaque vente partageant le même code département.

Out[60]:

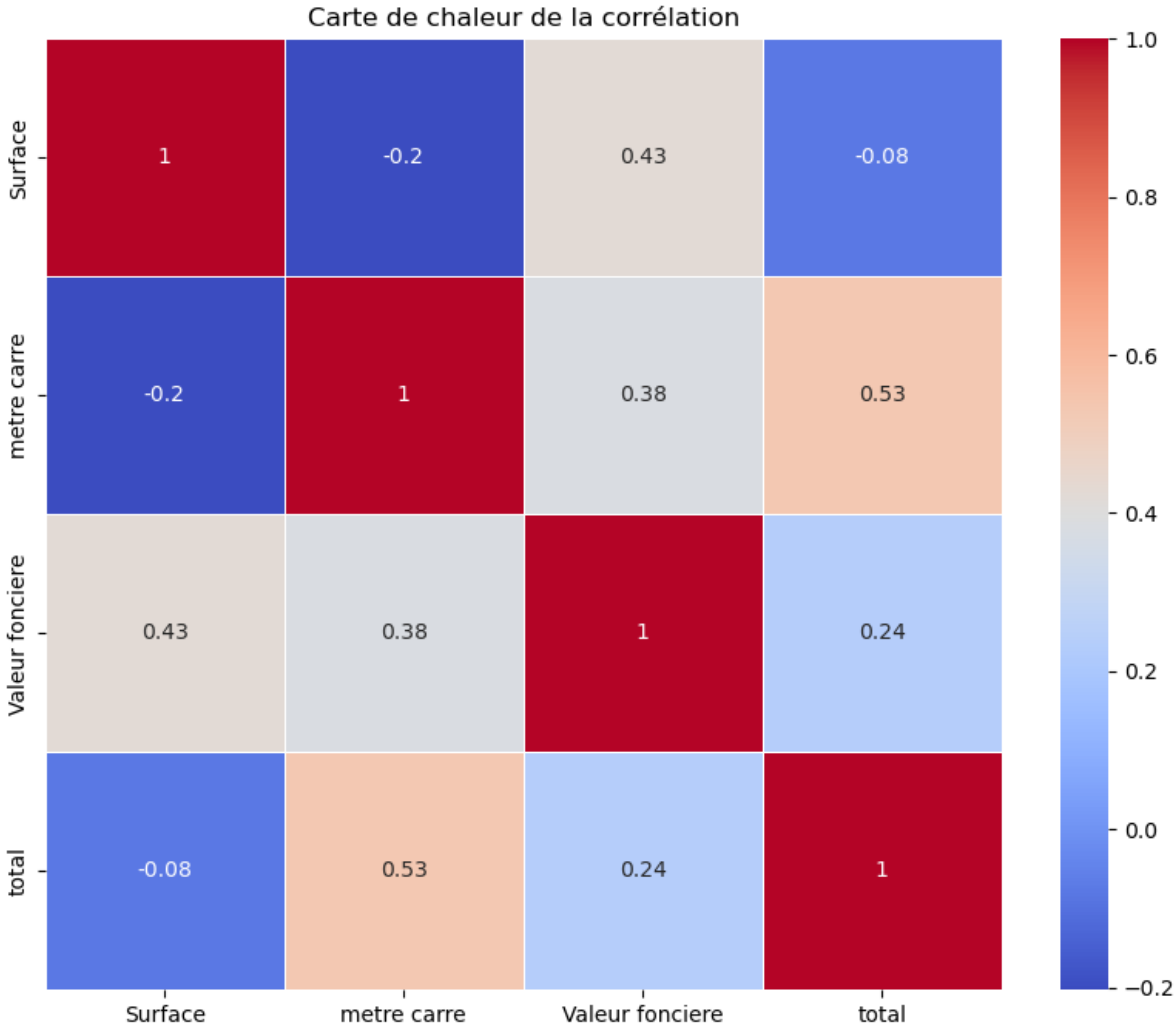
	Date mutation	Nature mutation	Valeur fonciere	Code postal	Code departement	Code commune	Nombre de lots	Code type local	Type lo
0	2018-04-01	Vente	67000.0	1000.0	01	53	1	2.0	Apparten
1	2018-11-01	Vente	76200.0	1000.0	01	53	2	2.0	Apparten
2	2018-12-01	Vente	130000.0	1160.0	01	430	0	1.0	Mai
3	2018-04-01	Vente	164370.0	1290.0	01	123	0	1.0	Mai
4	2018-01-15	Vente	97000.0	1750.0	01	370	1	2.0	Apparten

Out[70]: (2273401, 4)

Out[73]:
```
Surface            0
metre carre        0
Valeur fonciere    0
total              0
dtype: int64
```

Out[76]: (1773562, 4)

On a enlevé les outliers variables d'intérêts 'metre carre','Valeur fonciere', 'total', 'Surface', et remplacer les données manquantes par leur médiane.



Nos variables d'intérêts montrent des valeurs de corrélation satisfaisantes avec notres variables cible "Valeur foncière".

Out[86]:

StandardScaler

StandardScaler()

Out[87]:

	Surface	metre carre	total
0	0.175759	-1.239564	-1.315762
1	-0.427993	-0.266037	-0.510497
2	1.685140	-1.392390	-1.185678
3	-0.397806	-0.266037	-0.510497
4	-0.427993	-0.097255	2.625118

Nous normalisons les variables X train, puis nous appliquons la même normalisation sur les données X test (scaler.transform)

Out[92]:

```
DecisionTreeRegressor(max_depth=9, min_samples_leaf=5, min_samples_split=200,
                      random_state=0)
```

73369.48688053069 0.4526700299322568

Avec le modèle DecisionTreeRegressor nous obtenons un RMSE= 73369.48688053069 ; et un r^2 = 0.4526700299322568

Out[94]: array(['Surface', 'metre carre', 'total'], dtype=object)

Out[95]: array([0.46718728, 0.51190884, 0.02090389])

Les variables ayant le plus d'importance dans notre modèles sont "Surface" et "metre carre".

Nous essayons le modèle RandomForestRegressor

Out[96]:

```
RandomForestRegressor(max_depth=9, min_samples_leaf=5, min_samples_split=400,
                      n_estimators=150, random_state=0)
```

73211.79744894287 0.4550202004623365

Avec le modèle DecisionTreeRegressor nous obtenons un RMSE= 73211.79744894287 ; et un r^2 = 0.4550202004623365, proche du modèle précédent. Le modèle RandomForestRegressor paraît être le meilleur modèle.
Les variables ayant le plus d'importance dans notre modèles sont "Surface" et "metre carre".

Out[98]: array(['Surface', 'metre carre', 'total'], dtype=object)

Out[99]: array([0.467056 , 0.51046854, 0.02247546])