projet2019

January 15, 2020

Projet Zillow-Prize 2019 Khalfaoui Ellyes Imen Rezoug Mohamed Benyassine

[51]:
```
import pandas as pd
import warnings
train = pd.read_csv('./properties_2016.csv', sep=',', quotechar='"')
warnings.filterwarnings("ignore") # supprime les warnings f
```

Dans la partie pré traitement, on a commencé d'abord par regarder notre jeu de données, voir quel type de probleme pourrait etre lié par rapport à ces données, comprendre ce que represente chaque caractéristique(attributs), d'abord par un appercu des premieres lignes du dataset, puis les statistiques déscriptives(mean,std..) pour chaque attribut, nous avons ensuite regarder le domaine des instances,puis pour des raisons de lisibilité, nous avons recoder les noms des attributs, après avoir lu le dictionnaire joint au dataset, nous avons ensuite regardé les valeurs nulles (Nan) et leurs proportion dans le dataset, nous avons pris la décision d'éxclure certains attributs du traitement si le taux est proche de 100% de valeurs nulles, nous avons remplacé les valeurs manquantes des autres attributs par la valeur la plus representé (mod), on a vu d'autres méthodes pour le traitement des valeurs manquantes, dans un travail d'une autre personne, ils ont utilisé K-nn pour trouver les valeurs manquantes..

Nous avons ensuite combiné deux fichier pour avoir notre dataset d'entrainement (ajout du fichier de logerror correspondant à l'ID de la proprieté), nous avons ensuite fait des visualisations afin de voir la pértinenace des attributs, et notamment en associant le logerror qui represente la donnée cible du problème avec d'autres attributs, comme par exmple l'évolution du logerror (valeur absolue) au fil des mois pour l'année 2016

Nous avons ensuite visualisé les corrélation entre les attributs (corrélation de Pearson) afin de détérminer les attributs pertinents

Nous avons également transformé les valeurs ordinales en valeurs quantitatives pour certains attributs avant de réaliser une normalisation des données (afin que les résultats de modèles statistiques ne soient pas biaisiés par la différence d'échelle des valeurs des différents attributs).

[52]:
```
train.head()
"""affichage des premieres lignes du dataset"""
```

[52]: 'affichage des premieres lignes du dataset'

[53]:
```
"""affichage des statistiques de bases sur les attributs du dataset"""
train.describe()
```

[53]:

	parcelid	airconditioningtypeid	architecturalstyletypeid	\
count	2.985217e+06	811519.000000	6061.000000	
mean	1.332586e+07	1.931166	7.202607	
std	7.909966e+06	3.148587	2.436290	
min	1.071172e+07	1.000000	2.000000	
25%	1.164371e+07	1.000000	7.000000	
50%	1.254509e+07	1.000000	7.000000	
75%	1.409712e+07	1.000000	7.000000	
max	1.696019e+08	13.000000	27.000000	

	basementsqft	bathroomcnt	bedroomcnt	buildingclasstypeid	\
count	1628.000000	2.973755e+06	2.973767e+06	12629.000000	
mean	646.883292	2.209143e+00	3.088949e+00	3.725948	
std	538.793473	1.077754e+00	1.275859e+00	0.501700	
min	20.000000	0.000000e+00	0.000000e+00	1.000000	
25%	272.000000	2.000000e+00	2.000000e+00	3.000000	
50%	534.000000	2.000000e+00	3.000000e+00	4.000000	
75%	847.250000	3.000000e+00	4.000000e+00	4.000000	
max	8516.000000	2.000000e+01	2.000000e+01	5.000000	

	buildingqualitytypeid	calculatedbathnbr	decktypeid	…	\
count	1.938488e+06	2.856305e+06	17096.0	…	
mean	5.784787e+00	2.299263e+00	66.0	…	
std	1.805352e+00	1.000736e+00	0.0	…	
min	1.000000e+00	1.000000e+00	66.0	…	
25%	4.000000e+00	2.000000e+00	66.0	…	
50%	7.000000e+00	2.000000e+00	66.0	…	
75%	7.000000e+00	3.000000e+00	66.0	…	
max	1.200000e+01	2.000000e+01	66.0	…	

	yardbuildingsqft26	yearbuilt	numberofstories	\
count	2647.000000	2.925289e+06	682069.000000	
mean	278.296562	1.964262e+03	1.401464	
std	369.731508	2.344132e+01	0.539076	
min	10.000000	1.801000e+03	1.000000	
25%	96.000000	1.950000e+03	1.000000	
50%	168.000000	1.963000e+03	1.000000	
75%	320.000000	1.981000e+03	2.000000	
max	6141.000000	2.015000e+03	41.000000	

	structuretaxvaluedollarcnt	taxvaluedollarcnt	assessmentyear	\
count	2.930235e+06	2.942667e+06	2.973778e+06	
mean	1.708836e+05	4.204790e+05	2.014999e+03	
std	4.020683e+05	7.263467e+05	3.683161e-02	
min	1.000000e+00	1.000000e+00	2.000000e+03	
25%	7.480000e+04	1.796750e+05	2.015000e+03	
50%	1.225900e+05	3.060860e+05	2.015000e+03	

```
75%                  1.968890e+05        4.880000e+05     2.015000e+03
max                  2.514860e+08        2.827860e+08     2.016000e+03

       landtaxvaluedollarcnt     taxamount  taxdelinquencyyear  \
count           2.917484e+06  2.953967e+06        56464.000000
mean            2.524780e+05  5.377607e+03           13.892409
std             4.450132e+05  9.183107e+03            2.581006
min             1.000000e+00  1.340000e+00            0.000000
25%             7.483600e+04  2.461070e+03           14.000000
50%             1.670420e+05  3.991780e+03           14.000000
75%             3.069180e+05  6.201005e+03           15.000000
max             9.024622e+07  3.458861e+06           99.000000

       censustractandblock
count         2.910091e+06
mean          6.048431e+13
std           3.249035e+11
min          -1.000000e+00
25%           6.037400e+13
50%           6.037572e+13
75%           6.059042e+13
max           4.830301e+14

[8 rows x 53 columns]
```

[54]:
```
train.shape
"""taille du dataset"""
```

[54]: 'taille du dataset'

[55]:
```
"""domaine des instances"""
for key in train.keys():
    print (key, ": [", min(train[key]), max(train[key]), "]")
    print ('U')
```

```
('parcelid', ': [', 10711725, 169601949, ']')
U
('airconditioningtypeid', ': [', nan, nan, ']')
U
('architecturalstyletypeid', ': [', nan, nan, ']')
U
('basementsqft', ': [', nan, nan, ']')
U
('bathroomcnt', ': [', 0.0, 20.0, ']')
U
('bedroomcnt', ': [', 0.0, 20.0, ']')
```

```
U
('buildingclasstypeid', ': [', nan, nan, ']')
U
('buildingqualitytypeid', ': [', nan, nan, ']')
U
('calculatedbathnbr', ': [', nan, nan, ']')
U
('decktypeid', ': [', nan, nan, ']')
U
('finishedfloor1squarefeet', ': [', nan, nan, ']')
U
('calculatedfinishedsquarefeet', ': [', nan, nan, ']')
U
('finishedsquarefeet12', ': [', nan, nan, ']')
U
('finishedsquarefeet13', ': [', nan, nan, ']')
U
('finishedsquarefeet15', ': [', nan, nan, ']')
U
('finishedsquarefeet50', ': [', nan, nan, ']')
U
('finishedsquarefeet6', ': [', nan, nan, ']')
U
('fips', ': [', 6037.0, 6111.0, ']')
U
('fireplacecnt', ': [', nan, nan, ']')
U
('fullbathcnt', ': [', nan, nan, ']')
U
('garagecarcnt', ': [', nan, nan, ']')
U
('garagetotalsqft', ': [', nan, nan, ']')
U
('hashottuborspa', ': [', nan, nan, ']')
U
('heatingorsystemtypeid', ': [', nan, nan, ']')
U
('latitude', ': [', 33324388.0, 34819650.0, ']')
U
('longitude', ': [', -119475780.0, -117554316.0, ']')
U
('lotsizesquarefeet', ': [', 100.0, 328263808.0, ']')
U
('poolcnt', ': [', nan, nan, ']')
U
('poolsizesum', ': [', nan, nan, ']')
U
('pooltypeid10', ': [', nan, nan, ']')
```

```
U
('pooltypeid2', ': [', nan, nan, ']')
U
('pooltypeid7', ': [', nan, nan, ']')
U
('propertycountylandusecode', ': [', nan, 'SFR', ']')
U
('propertylandusetypeid', ': [', 31.0, 275.0, ']')
U
('propertyzoningdesc', ': [', nan, 'ZONE LCC3', ']')
U
('rawcensustractandblock', ': [', 60371011.101, 61110091.003010996, ']')
U
('regionidcity', ': [', 3491.0, 396556.0, ']')
U
('regionidcounty', ': [', 1286.0, 3101.0, ']')
U
('regionidneighborhood', ': [', nan, nan, ']')
U
('regionidzip', ': [', 95982.0, 399675.0, ']')
U
('roomcnt', ': [', 0.0, 96.0, ']')
U
('storytypeid', ': [', nan, nan, ']')
U
('threequarterbathnbr', ': [', nan, nan, ']')
U
('typeconstructiontypeid', ': [', nan, nan, ']')
U
('unitcnt', ': [', nan, nan, ']')
U
('yardbuildingsqft17', ': [', nan, nan, ']')
U
('yardbuildingsqft26', ': [', nan, nan, ']')
U
('yearbuilt', ': [', nan, nan, ']')
U
('numberofstories', ': [', nan, nan, ']')
U
('fireplaceflag', ': [', nan, nan, ']')
U
('structuretaxvaluedollarcnt', ': [', nan, nan, ']')
U
('taxvaluedollarcnt', ': [', 1.0, 282786000.0, ']')
U
('assessmentyear', ': [', 2000.0, 2016.0, ']')
U
('landtaxvaluedollarcnt', ': [', 1.0, 90246219.0, ']')
```

```
U
('taxamount', ': [', nan, nan, ']')
U
('taxdelinquencyflag', ': [', nan, 'Y', ']')
U
('taxdelinquencyyear', ': [', nan, nan, ']')
U
('censustractandblock', ': [', nan, nan, ']')
U
```

[56]:
```python
"""renommer les attributs pour plus de clarté (inspiré d'un notebook fait avec␣
↪R)"""
train.rename(columns = {  "parcelid" : "id_parcel",
  "yearbuilt" : "build_year",
  "basementsqft" : "area_basement",
  "yardbuildingsqft17" : "area_patio",
  "yardbuildingsqft26" : "area_shed",
  "poolsizesum" : "area_pool",
  "lotsizesquarefeet" : "area_lot",
  "garagetotalsqft" : "area_garage",
  "finishedfloor1squarefeet" : "area_firstfloor_finished",
  "calculatedfinishedsquarefeet" : "area_total_calc",
  "finishedsquarefeet6" : "area_base",
  "finishedsquarefeet12" : "area_live_finished",
  "finishedsquarefeet13" : "area_liveperi_finished",
  "finishedsquarefeet15" : "area_total_finished",
  "finishedsquarefeet50" : "area_unknown",
  "unitcnt" : "num_unit",
  "numberofstories" : "num_story",
  "roomcnt" : "num_room",
  "bathroomcnt" : "num_bathroom",
  "bedroomcnt" : "num_bedroom",
  "calculatedbathnbr" : "num_bathroom_calc",
  "fullbathcnt" : "num_bath",
  "threequarterbathnbr" : "num_75_bath",
  "fireplacecnt" : "num_fireplace",
  "poolcnt" : "num_pool",
  "garagecarcnt" : "num_garage",
  "regionidcounty" : "region_county",
  "regionidcity" : "region_city",
  "regionidzip" : "region_zip",
  "regionidneighborhood" : "region_neighbor",
  "taxvaluedollarcnt" : "tax_total",
  "structuretaxvaluedollarcnt" : "tax_building",
  "landtaxvaluedollarcnt" : "tax_land",
  "taxamount" : "tax_property",
  "assessmentyear" : "tax_year",
```

```python
    "taxdelinquencyflag" : "tax_delinquency",
    "taxdelinquencyyear" : "tax_delinquency_year",
    "propertyzoningdesc" : "zoning_property",
    "propertylandusetypeid" : "zoning_landuse",
    "propertycountylandusecode" : "zoning_landuse_county",
    "fireplaceflag" : "flag_fireplace",
    "hashottuborspa" : "flag_tub",
    "buildingqualitytypeid" : "quality",
    "buildingclasstypeid" : "framing",
    "typeconstructiontypeid" : "material",
    "decktypeid" : "deck",
    "storytypeid" : "story",
    "heatingorsystemtypeid" : "heating",
    "airconditioningtypeid" : "aircon",
    "architecturalstyletypeid": "architectural_style"},
                                inplace = True)
```

[58]:
```python
print train.head()
```

```
   id_parcel  aircon  architectural_style  area_basement  num_bathroom  \
0   10754147     NaN                  NaN            NaN           0.0
1   10759547     NaN                  NaN            NaN           0.0
2   10843547     NaN                  NaN            NaN           0.0
3   10859147     NaN                  NaN            NaN           0.0
4   10879947     NaN                  NaN            NaN           0.0

   num_bedroom  framing  quality  num_bathroom_calc  deck  …  num_story  \
0          0.0      NaN      NaN                NaN   NaN  …        NaN
1          0.0      NaN      NaN                NaN   NaN  …        NaN
2          0.0      NaN      NaN                NaN   NaN  …        NaN
3          0.0      3.0      7.0                NaN   NaN  …        1.0
4          0.0      4.0      NaN                NaN   NaN  …        NaN

   flag_fireplace  tax_building  tax_total  tax_year  tax_land  tax_property  \
0             NaN           NaN        9.0    2015.0       9.0           NaN
1             NaN           NaN    27516.0    2015.0   27516.0           NaN
2             NaN      650756.0  1413387.0    2015.0  762631.0      20800.37
3             NaN      571346.0  1156834.0    2015.0  585488.0      14557.57
4             NaN      193796.0   433491.0    2015.0  239695.0       5725.17

   tax_delinquency  tax_delinquency_year  censustractandblock
0              NaN                   NaN                  NaN
1              NaN                   NaN                  NaN
2              NaN                   NaN                  NaN
3              NaN                   NaN                  NaN
4              NaN                   NaN                  NaN
```

[5 rows x 58 columns]

[59]:
```python
""" Affichage du nombre de  Valeurs manquantes pour chaque attributs"""
print('nombre de valeurs manquantes pour chaque attributs:\n')

print ('\n')
print( train.isnull().sum())
"""affichage du nombre total de valeur manquantes"""
print ('\n')
print('nombre total de valeurs manquantes: ',train.isnull().sum().sum())
```

```
nombre de valeurs manquantes pour chaque attributs:



id_parcel                          0
aircon                       2173698
architectural_style          2979156
area_basement                2983589
num_bathroom                   11462
num_bedroom                    11450
framing                      2972588
quality                      1046729
num_bathroom_calc             128912
deck                         2968121
area_firstfloor_finished     2782500
area_total_calc                55565
area_live_finished            276033
area_liveperi_finished       2977545
area_total_finished          2794419
area_unknown                 2782500
area_base                    2963216
fips                           11437
num_fireplace                2672580
num_bath                      128912
num_garage                   2101950
area_garage                  2101950
flag_tub                     2916203
heating                      1178816
latitude                       11437
longitude                      11437
area_lot                      276099
num_pool                     2467683
area_pool                    2957257
pooltypeid10                 2948278
pooltypeid2                  2953142
pooltypeid7                  2499758
zoning_landuse_county          12277
```

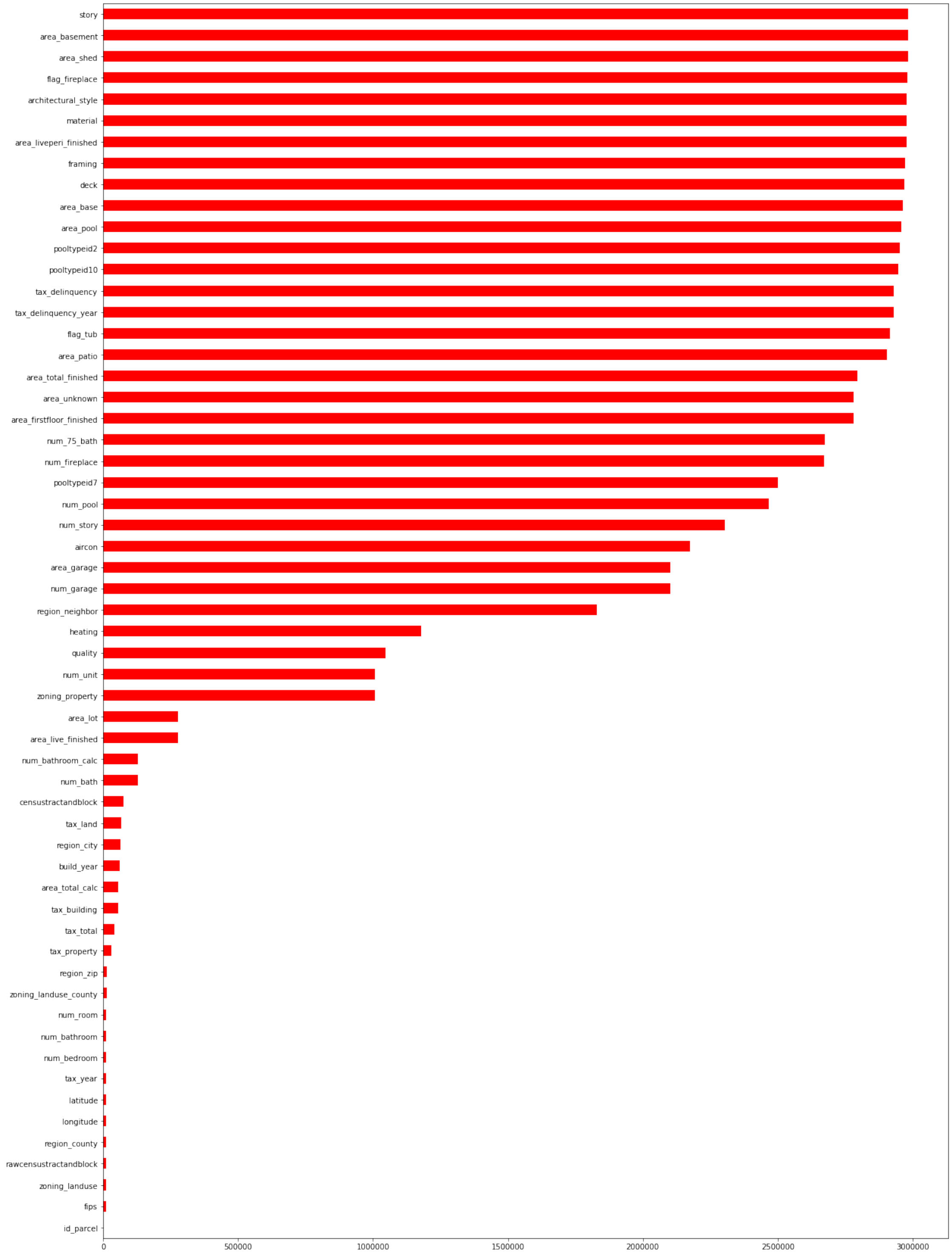
```
zoning_landuse                 11437
zoning_property              1006588
rawcensustractandblock         11437
region_city                    62845
region_county                  11437
region_neighbor              1828815
region_zip                     13980
num_room                       11475
story                        2983593
num_75_bath                  2673586
material                     2978470
num_unit                     1007727
area_patio                   2904862
area_shed                    2982570
build_year                     59928
num_story                    2303148
flag_fireplace               2980054
tax_building                   54982
tax_total                      42550
tax_year                       11439
tax_land                       67733
tax_property                   31250
tax_delinquency              2928755
tax_delinquency_year         2928753
censustractandblock            75126
dtype: int64
```

```
('nombre total de valeurs manquantes: ', 85129239)
```

On peut voir que le nombre de valeurs manquantes est énorme pour certains attributs, sachant que le nombre de valeurs possible pour chauqe attribut est : 2985217.

[60]:
```
missing_nb = train.isnull().sum()
```

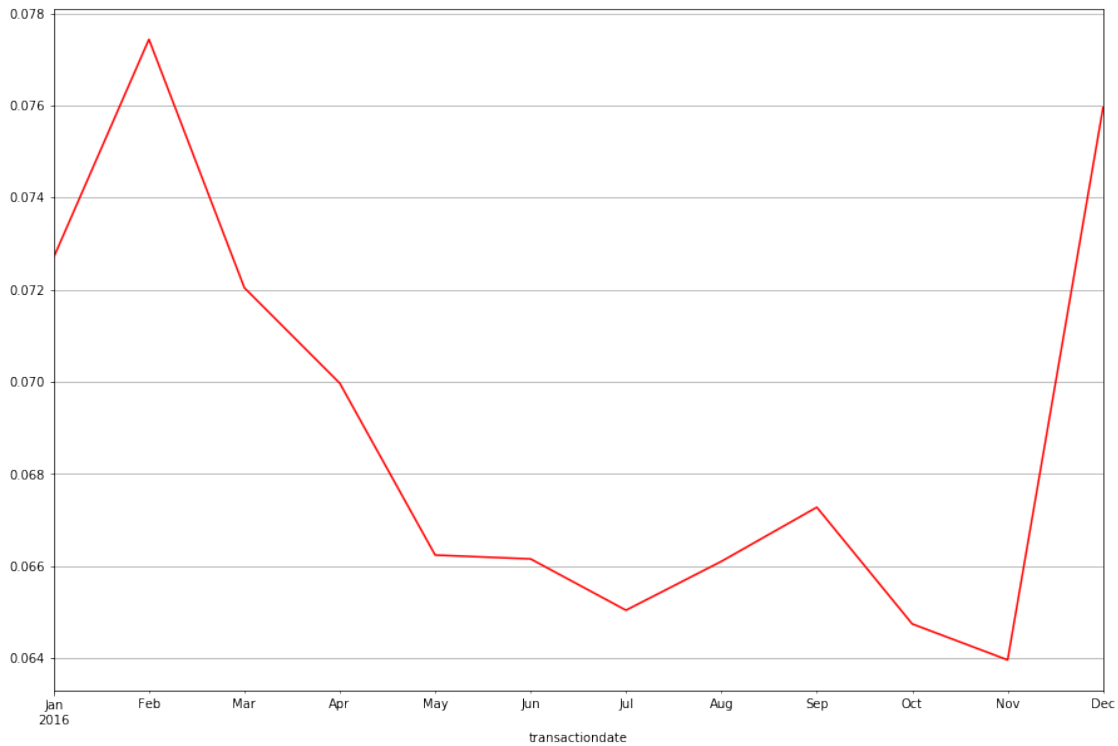
[61]:
```
""" affichage des valeurs manquantes pour tous les attributs en forme de␣
↪graphique pour mieux appércevoir la proportion"""
#print missing_nb.keys()

##missing_nb.plot.bar()
#ax = missing_nb.plot.bar(rot=0)
import matplotlib.pyplot as plt

ax = missing_nb.sort_values(ascending=True).plot.barh(x=missing_nb.keys(),␣
↪figsize=(20, 30), color= 'r')
```




```python
[62]: """importation du fichier qui contient les logerrors et date de transactions"""
logerror_2016 = pd.read_csv('./train_2016_v2.csv', sep=',', quotechar='"')
import pandas as pd
```

```python
print (logerror_2016.head())

logerror_2016['abslogerror'] =logerror_2016['logerror'].abs()


logerror_2016['transactiondate'] = pd.
 ↪to_datetime(logerror_2016['transactiondate'])
print type(logerror_2016['transactiondate'][0])
print logerror_2016.head()
a =logerror_2016.groupby(logerror_2016['transactiondate'].dt.
 ↪to_period("M"))['abslogerror'].mean()

print a.keys()
# un plot qui affiche la valeur absolue du logerror par mois
ax =a.plot.line(x=a.keys(), figsize=(15, 10), color= 'r', grid='true')
```

```
   parcelid  logerror transactiondate
0  11016594    0.0276      2016-01-01
1  14366692   -0.1684      2016-01-01
2  12098116   -0.0040      2016-01-01
3  12643413    0.0218      2016-01-02
4  14432541   -0.0050      2016-01-02
<class 'pandas._libs.tslibs.timestamps.Timestamp'>
   parcelid  logerror transactiondate  abslogerror
0  11016594    0.0276      2016-01-01       0.0276
1  14366692   -0.1684      2016-01-01       0.1684
2  12098116   -0.0040      2016-01-01       0.0040
3  12643413    0.0218      2016-01-02       0.0218
4  14432541   -0.0050      2016-01-02       0.0050
PeriodIndex(['2016-01', '2016-02', '2016-03', '2016-04', '2016-05', '2016-06',
             '2016-07', '2016-08', '2016-09', '2016-10', '2016-11', '2016-12'],
            dtype='period[M]', name=u'transactiondate', freq='M')
```



Si on affiche les valeurs absolues des logerror(prix prédit du zestimate - log du prix réel) on remarque que les valeurs absolue diminuent avec le temps

[63]:

```
""" inner join entre la table de donnée des caractéristiques des propriétés et␣
↪des log errors"""

merged = pd.merge(train,logerror_2016, left_on='id_parcel',␣
↪right_on='parcelid', how='inner').drop('parcelid', axis=1)
print (merged.shape)
print(merged.head())
```

```
(90275, 61)
   id_parcel  aircon  architectural_style  area_basement  num_bathroom  \
0   17073783     NaN                  NaN            NaN           2.5
1   17088994     NaN                  NaN            NaN           1.0
2   17100444     NaN                  NaN            NaN           2.0
3   17102429     NaN                  NaN            NaN           1.5
4   17109604     NaN                  NaN            NaN           2.5

   num_bedroom  framing  quality  num_bathroom_calc  deck  …  tax_total  \
0          3.0      NaN      NaN                2.5   NaN  …   191811.0
1          2.0      NaN      NaN                1.0   NaN  …   239679.0
2          3.0      NaN      NaN                2.0   NaN  …    47853.0
3          2.0      NaN      NaN                1.5   NaN  …    62914.0
```

```
4          4.0      NaN      NaN               2.5    NaN  …    554000.0

   tax_year  tax_land  tax_property  tax_delinquency  tax_delinquency_year  \
0    2015.0   76724.0       2015.06              NaN                   NaN
1    2015.0   95870.0       2581.30              NaN                   NaN
2    2015.0   14234.0        591.64              NaN                   NaN
3    2015.0   17305.0        682.78              NaN                   NaN
4    2015.0  277000.0       5886.92              NaN                   NaN

   censustractandblock  logerror transactiondate  abslogerror
0         6.111002e+13    0.0953      2016-01-27       0.0953
1         6.111002e+13    0.0198      2016-03-30       0.0198
2         6.111001e+13    0.0060      2016-05-27       0.0060
3         6.111001e+13   -0.0566      2016-06-07       0.0566
4         6.111001e+13    0.0573      2016-08-08       0.0573

[5 rows x 61 columns]
```

[64]:
```python
"""proprtion des valeurs manquantes par rapport à la taille des données (n␣
↪taille de l'ensmble de données S)"""
print( merged.isnull().sum()/merged.shape[0]*100)
```

```
id_parcel                     0.000000
aircon                       68.118527
architectural_style          99.710883
area_basement                99.952368
num_bathroom                  0.000000
num_bedroom                   0.000000
framing                      99.982276
quality                      36.456383
num_bathroom_calc             1.309333
deck                         99.271116
area_firstfloor_finished     92.405428
area_total_calc               0.732207
area_live_finished            5.183052
area_liveperi_finished       99.963445
area_total_finished          96.052063
area_unknown                 92.405428
area_base                    99.533647
fips                          0.000000
num_fireplace                89.358073
num_bath                      1.309333
num_garage                   66.837995
area_garage                  66.837995
flag_tub                     97.380227
heating                      37.878704
latitude                      0.000000
longitude                     0.000000
```

```
area_lot                    11.243423
num_pool                    80.170590
area_pool                   98.926613
pooltypeid10                98.713930
pooltypeid2                 98.666297
pooltypeid7                 81.504292
zoning_landuse_county        0.001108
zoning_landuse               0.000000
zoning_property             35.405151
rawcensustractandblock       0.000000
region_city                  1.997231
region_county                0.000000
region_neighbor             60.108557
region_zip                   0.038770
num_room                     0.000000
story                       99.952368
num_75_bath                 86.697314
material                    99.668790
num_unit                    35.360842
area_patio                  97.068956
area_shed                   99.894766
build_year                   0.837441
num_story                   77.214068
flag_fireplace              99.754085
tax_building                 0.420936
tax_total                    0.001108
tax_year                     0.000000
tax_land                     0.001108
tax_property                 0.006646
tax_delinquency             98.024924
tax_delinquency_year        98.024924
censustractandblock          0.670174
logerror                     0.000000
transactiondate              0.000000
abslogerror                  0.000000
dtype: float64
```

On remarque que pour certains attributs, la proportion de valeurs manquantes est proche de 100%, ces attributs sont donc à priori non nécéssaire pour faire notre étude sur cet ensmble

[65]:
```
merged.drop(['architectural_style', 'area_basement', 'framing',␣
 ↪'deck','area_base', 'area_liveperi_finished',
            'area_base', 'pooltypeid10', 'pooltypeid2', 'story', 'material',␣
 ↪'area_shed', 'flag_fireplace' ], axis='columns', inplace=True)
```

[66]:
```
""" traitement des valeurs manquantes, on a choisi le mode comme mesure de␣
 ↪remplacement"""
from scipy.stats import mode
```

```python
merged['num_bathroom_calc'].mode()

for key in merged.keys():
    merged[key].fillna(merged[key].mode().iloc[0], inplace=True)
```

[67]:
```python
"""domaine du logerror (cible)"""
print min(merged['logerror']), max(merged['logerror'])
```

```
-4.605 4.737
```

On s'appércoit que les valeurs des attributs ont des échélles différentes, on se doit de les normaliser pour qu'on ai pas des valeurs qui vont biaiser nos modèles, on doit aussi regarder les types des valeurs, si y'a des valeurs de type booléen ou de type nominale, on se doit des les normaliser aussi, car la plus part des modèles statistiques s'appliquent sur des données numériques à priori et en particulier dans notre cas, car notre probleme est à priori un probleme de régréssion.

[68]:
```python
"""affichage des types des """

for key in merged.keys():

    print (key, type(merged[key][0]))
```

```
('id_parcel', <type 'numpy.int64'>)
('aircon', <type 'numpy.float64'>)
('num_bathroom', <type 'numpy.float64'>)
('num_bedroom', <type 'numpy.float64'>)
('quality', <type 'numpy.float64'>)
('num_bathroom_calc', <type 'numpy.float64'>)
('area_firstfloor_finished', <type 'numpy.float64'>)
('area_total_calc', <type 'numpy.float64'>)
('area_live_finished', <type 'numpy.float64'>)
('area_total_finished', <type 'numpy.float64'>)
('area_unknown', <type 'numpy.float64'>)
('fips', <type 'numpy.float64'>)
('num_fireplace', <type 'numpy.float64'>)
('num_bath', <type 'numpy.float64'>)
('num_garage', <type 'numpy.float64'>)
('area_garage', <type 'numpy.float64'>)
('flag_tub', <type 'numpy.bool_'>)
('heating', <type 'numpy.float64'>)
('latitude', <type 'numpy.float64'>)
('longitude', <type 'numpy.float64'>)
('area_lot', <type 'numpy.float64'>)
('num_pool', <type 'numpy.float64'>)
('area_pool', <type 'numpy.float64'>)
('pooltypeid7', <type 'numpy.float64'>)
```

```
('zoning_landuse_county', <type 'str'>)
('zoning_landuse', <type 'numpy.float64'>)
('zoning_property', <type 'str'>)
('rawcensustractandblock', <type 'numpy.float64'>)
('region_city', <type 'numpy.float64'>)
('region_county', <type 'numpy.float64'>)
('region_neighbor', <type 'numpy.float64'>)
('region_zip', <type 'numpy.float64'>)
('num_room', <type 'numpy.float64'>)
('num_75_bath', <type 'numpy.float64'>)
('num_unit', <type 'numpy.float64'>)
('area_patio', <type 'numpy.float64'>)
('build_year', <type 'numpy.float64'>)
('num_story', <type 'numpy.float64'>)
('tax_building', <type 'numpy.float64'>)
('tax_total', <type 'numpy.float64'>)
('tax_year', <type 'numpy.float64'>)
('tax_land', <type 'numpy.float64'>)
('tax_property', <type 'numpy.float64'>)
('tax_delinquency', <type 'str'>)
('tax_delinquency_year', <type 'numpy.float64'>)
('censustractandblock', <type 'numpy.float64'>)
('logerror', <type 'numpy.float64'>)
('transactiondate', <class 'pandas._libs.tslibs.timestamps.Timestamp'>)
('abslogerror', <type 'numpy.float64'>)
```

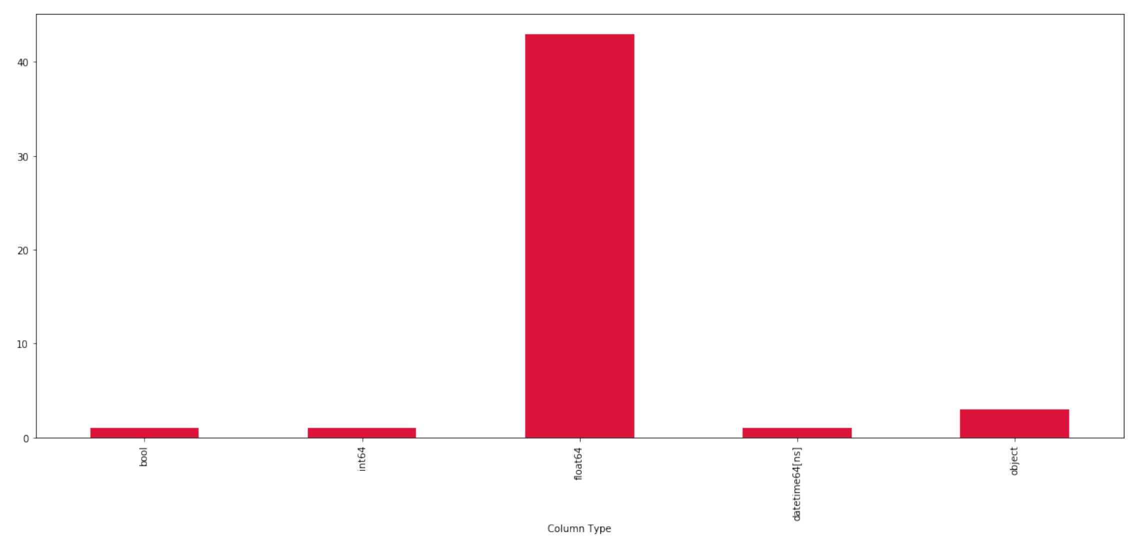
[69]:

```
"""affichage tabulaire des types d'attributs"""
pd.options.display.max_rows = 65

dtype_df = merged.dtypes.reset_index()
dtype_df.columns = ["Count", "Column Type"]
dtype_df
def background_color(val):
    if val == object:
        color = 'yellow'
    elif val == int:
        color = 'pink'
    elif val == float:
        color = 'crimson'
    else: color = 'orange'
    return 'background-color: {}'.format(color)
s = dtype_df.style.applymap(background_color)
s
```

[69]: <pandas.io.formats.style.Styler at 0x7f65a03109d0>

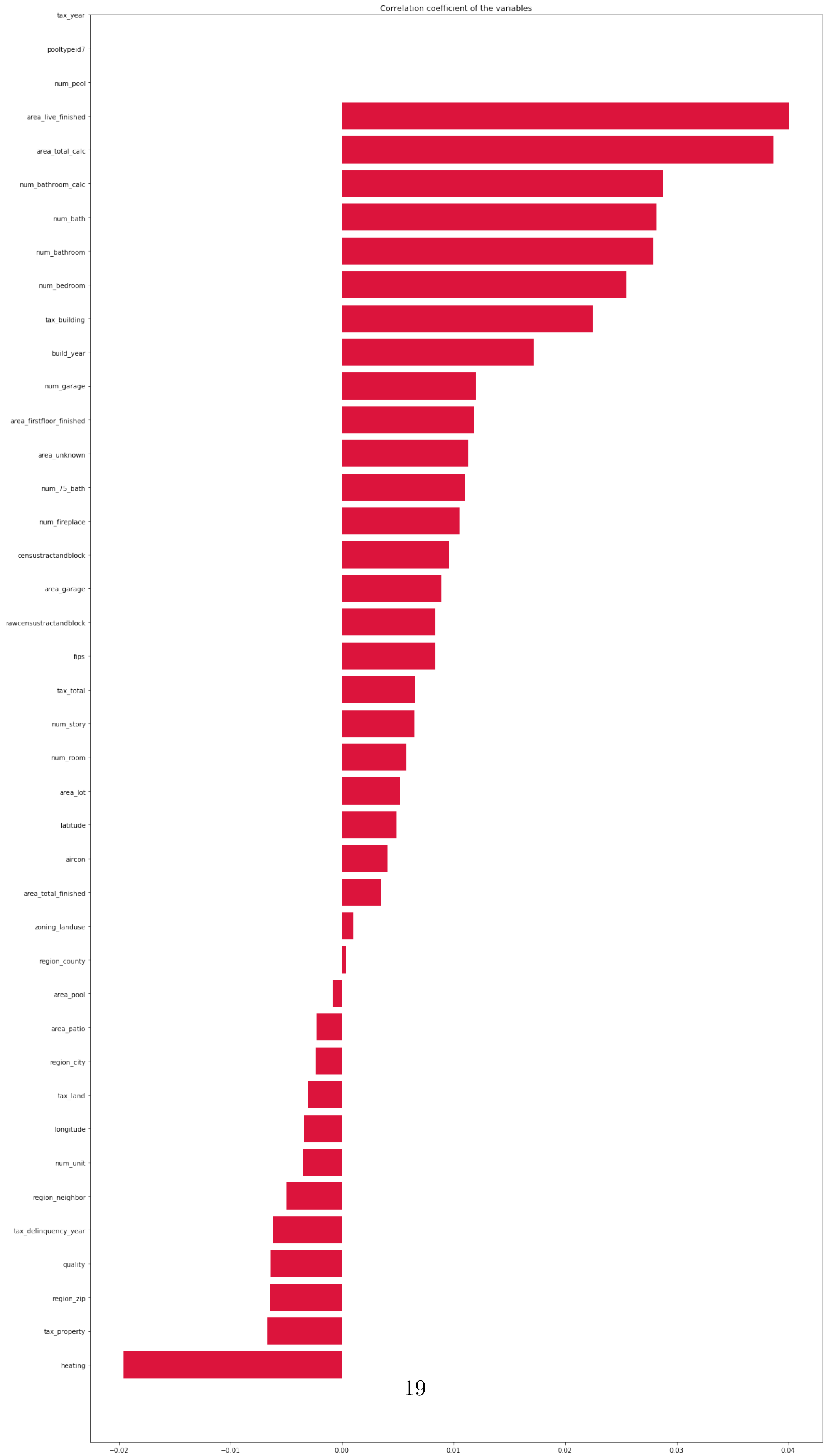
[70]:
```
"""affichage graphique des types d'attributs"""


d = dtype_df.groupby("Column Type").aggregate('count').reset_index()

print a
d =dtype_df.groupby(dtype_df["Column Type"])['Count'].count()


print d.keys()
ax =d.plot.bar(x=d.keys(),figsize=(20, 8), color= 'crimson', grid='false')
```

```
transactiondate
2016-01    0.072695
2016-02    0.077434
2016-03    0.072044
2016-04    0.069972
2016-05    0.066241
2016-06    0.066158
2016-07    0.065044
2016-08    0.066104
2016-09    0.067279
2016-10    0.064746
2016-11    0.063965
2016-12    0.075952
Freq: M, Name: abslogerror, dtype: float64
Index([bool, int64, float64, datetime64[ns], object], dtype='object',
name=u'Column Type')
```



[71]:
```
del merged['abslogerror']
# suprresion de la colonne de abslogerror qui a servit d'étudier la valeur en␣
↪fonction des mois de transaction
```

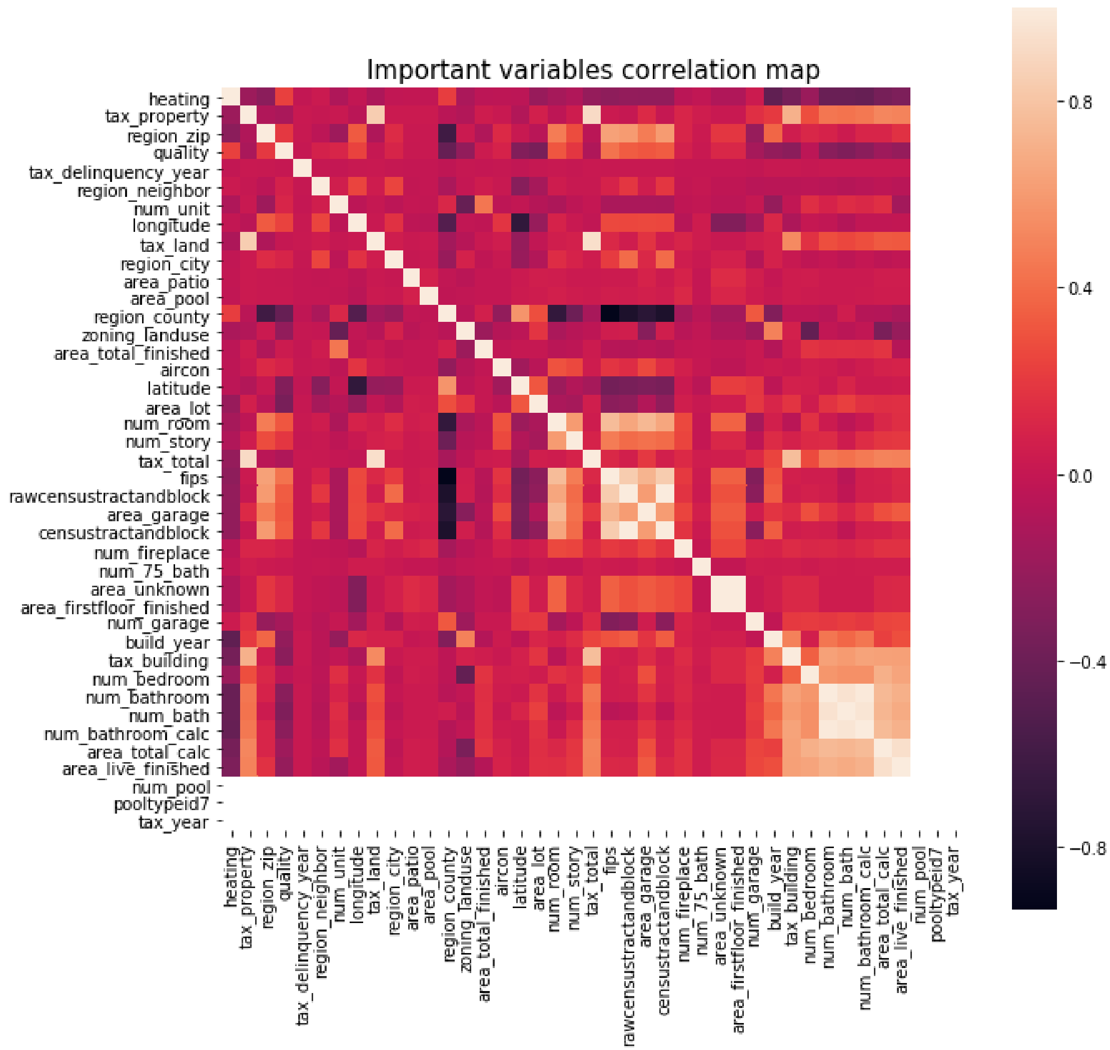
[72]:
```
""" etudes de corrélation des variables inspiré de ce notebook https://www.
↪kaggle.com/sudalairajkumar/simple-exploration-notebook-zillow-prize"""
import numpy as np
x_cols = [col for col in merged.columns if col not in ['logerror']  if␣
↪merged[col].dtype=='float64']

labels = []
values = []
for col in x_cols:
    labels.append(col)
    values.append(np.corrcoef(merged[col].values, merged.logerror.values)[0,1])
    """numpy.corrcoef(x, y=None, rowvar=True, bias=<no value>, ddof=<no␣
↪value>)[source]
Return Pearson product-moment correlation coefficients"""
corr_df = pd.DataFrame({'col_labels':labels, 'corr_values':values})
corr_df = corr_df.sort_values(by='corr_values')

ind = np.arange(len(labels))
width = 0.9
fig, ax = plt.subplots(figsize=(20,40))
rects = ax.barh(ind, np.array(corr_df.corr_values.values), color='crimson')
ax.set_yticks(ind)
ax.set_yticklabels(corr_df.col_labels.values, rotation='horizontal')
ax.set_xlabel("Correlation coefficient")
ax.set_title("Correlation coefficient of the variables")
plt.show()
```


Correlation coefficient of the variables
tax_year
pooltypeid7
num_pool
area_live_finished
area_total_calc
num_bathroom_calc
num_bath
num_bathroom
num_bedroom
tax_building
build_year
num_garage
area_firstfloor_finished
area_unknown
num_75_bath
num_fireplace
censustractandblock
area_garage
rawcensustractandblock
fips
tax_total
num_story
num_room
area_lot
latitude
aircon
area_total_finished
zoning_landuse
region_county
area_pool
area_patio
region_city
tax_land
longitude
num_unit
region_neighbor
tax_delinquency_year
quality
region_zip
tax_property
heating
−0.02
−0.01
0.00
0.01
0.02
0.03
0.04
Correlation coefficient

[73]:

```
"""affichage d'une heatmap de correlation entre toutes les caractéristiques"""
corr_df_sel = corr_df.ix[(corr_df['corr_values']!=0)]
corr_df_sel
import seaborn as sns
cols_to_use = corr_df_sel.col_labels.tolist()
print len(cols_to_use)

temp_df = merged[cols_to_use]
corrmat = temp_df.corr(method='spearman')
f, ax = plt.subplots(figsize=(10, 10))

# Draw the heatmap using seaborn
sns.heatmap(corrmat, vmax=1., square=True)
plt.title("Important variables correlation map", fontsize=15)
plt.show()
```

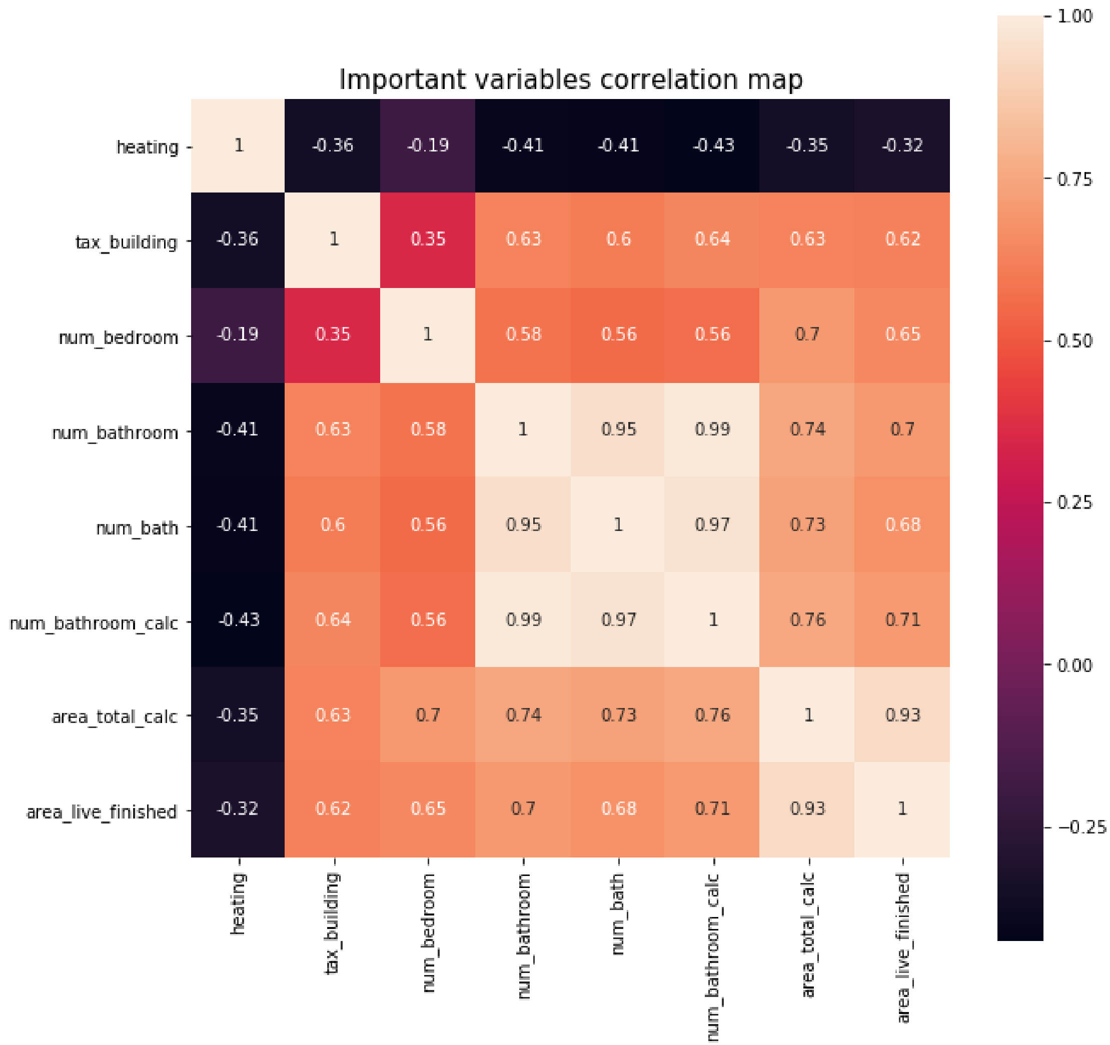
41




```
[74]: """affichage d'une heatmap de correlation entre les caractéristiques les␣
       ↪corrélations les plus elevés"""
      corr_df_sel = corr_df.ix[(corr_df['corr_values']>0.02) |␣
       ↪(corr_df['corr_values'] < -0.01)]
      corr_df_sel

      cols_to_use = corr_df_sel.col_labels.tolist()
      print len(cols_to_use)

      temp_df = merged[cols_to_use]
      corrmat = temp_df.corr(method='spearman')
      f, ax = plt.subplots(figsize=(10, 10))

      # Draw the heatmap using seaborn
```

```
sns.heatmap(corrmat, vmax=1., annot=True,square=True)
plt.title("Important variables correlation map", fontsize=15)
plt.show()
```

8



On a exploré les possibilités de transformation des valeurs ordinales, il y'a quelques attributs dans le dataset qui sont de ce type, mais après avoir compris leur signification et avoir testé quelques méthodes de transformations, en commencant par un OrdinalEncoder de la libraire sckit learn, qui conciste à attribuer des valeurs numériques pour chaque label, ce qui n'a pas forcément beacoup de sens point de vue statistque (avis subjectif) car attribuer une valeur numérique plus grande pour un Code de zonne par exemple par rapport un à un autre compte va biaiser le modèle, autre méthode OneHotEncoder qui contrairement au à une méthode de qui va transformer les attributs en variables Dummy, et attribuer par exemple un 1 si une instance contient le code et 0 sinon, elle va encoder les valeurs ordinales en valeurs binaires uniques, après avoir transformé les attribut au préalable en valeur numérique avec une une fonction LabelEncoder, on a eu des dataframes de 70 colonnes suplémentaires pour representer ces Dummy variables et au vu de la signification de ces attributs et en général pour les travaux dans les différents notebook réalisés sur ces données et au vu des résultats de régréssion obtenus, on a eu de meilleurs résultats en excluant ces attributs

[75]:
```
"""choix du OneHotENcoder pour la codification de nos variables ordinales """
print(merged['zoning_landuse_county'][1000])
print type(merged['zoning_landuse_county'][12])
"""valeurs booléennes remplacées par des valeurs comprises entre 0 et 1"""
#print ((merged['flag_tub'][0]))
#merged['flag_tub'] = merged['flag_tub'].replace(True, 1)
#merged['flag_tub'] = merged['flag_tub'].replace(False, 0)

#merged['zoning_landuse_county'] = pd.
 ↪to_numeric(merged['zoning_landuse_county'])
#print (merged['zoning_landuse_county'][26])

#pd.to_numeric(merged['zoning_landuse_county'])
""" attribut avec valeurs nominales ici des codes"""
print (merged['zoning_landuse_county'].unique())
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder


X = [['1128', '1129', '1110' ,'1111','0100', '0101', '010D', '010C' ,'010E'␣
 ↪,'0200',
 '0700', '0400', '0300', '122', '34' ,'01DC' ,'1', '012C' ,'01HC' ,'010V'␣
 ↪,'1117',
 '0104', '020G', '0109', '96', '1321', '1222', '1116' ,'010M', '1210', '010G',
 '0103' ,'38' ,'010H', '73', '1112', '0108', '135', '010F' ,'1410' ,'012D'␣
 ↪,'0201',
 '6050', '070D', '1200' ,'0401', '1720', '020M', '105', '012E', '0102', '1310',
 '010', '040V', '030G' ,'0110', '1421' ,'1432', '1011' ,'0111' '0130' ,'1333',
 '01DD', '0' ,'0210' ,'0131' ,'040A' ,'1722', '0105' ,'1420' ,'0114']]
enc = OneHotEncoder(handle_unknown='ignore')
enc.fit_transform(X)
C = enc.transform(X).toarray()
from sklearn.preprocessing import OrdinalEncoder
print type(merged['zoning_landuse_county'])
ordinalencoder = OrdinalEncoder()
ordinalencoder.fit_transform(merged[['zoning_landuse_county']])
```

```
0100
<type 'str'>
['1128' '1129' '1111' '1110' '010C' '0100' '0101' '010D' '010E' '0200'
 '0700' '0400' '0300' '122' '34' '01DC' '1' '012C' '01HC' '100V' '1117'
 '0104' '020G' '0109' '96' '1321' '010V' '1222' '1116' '010M' '1210'
 '010G' '0103' '38' '010H' '73' '1112' '0108' '135' '010F' '1014' '1410'
```

```
 '012D' '0201' '6050' '070D' '1200' '0401' '1720' '020M' '105' '012E'
 '1012' '1011' '1310' '010' '040V' '030G' '0110' '0102' '1421' '1432'
 '0303' '0111' '0130' '1333' '01DD' '0' '0210' '0131' '8800' '040A' '200'
 '0301' '1722' '1420' '0114']
<class 'pandas.core.series.Series'>
```

[75]:
```
array([[54.],
       [55.],
       [50.],
       …,
       [ 9.],
       [43.],
       [43.]])
```

[76]:
```python
"""codification en valeurs uniques binaires, solution potontiellement meilleure␣
↪que l'encoding ordinal ou Dummy"""
from sklearn.preprocessing import LabelEncoder
lenc = LabelEncoder()
merged['zoning_landuse_county'] = lenc.
↪fit_transform(merged['zoning_landuse_county'])

lenc.classes_

from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(categorical_features=[0], sparse=False)
ohe_results = ohe.fit_transform(merged[['zoning_landuse_county']])
df_ohe_results = pd.DataFrame(ohe_results, columns=lenc.classes_)
df_ohe_results.head()
```

[76]:
```
     0  010  0100  0101  0102  0103  0104  0108  0109  010C  …  1432  1720  \
0  0.0  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  …   0.0   0.0
1  0.0  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  …   0.0   0.0
2  0.0  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  …   0.0   0.0
3  0.0  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  …   0.0   0.0
4  0.0  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  …   0.0   0.0

   1722  200   34   38  6050   73  8800   96
0   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0
1   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0
2   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0
3   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0
4   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0

[5 rows x 77 columns]
```

[78]:
```python
# création d'un dataframe avec les valeurs ordinales transformés
```

```
df = pd.concat([merged.reset_index(drop=True), df_ohe_results.
 ↪reset_index(drop=True)], axis=1)
df.head()
```

[78]:

```
   id_parcel  aircon  num_bathroom  num_bedroom  quality  num_bathroom_calc  \
0   17073783     1.0           2.5          3.0      7.0                2.5
1   17088994     1.0           1.0          2.0      7.0                1.0
2   17100444     1.0           2.0          3.0      7.0                2.0
3   17102429     1.0           1.5          2.0      7.0                1.5
4   17109604     1.0           2.5          4.0      7.0                2.5

   area_firstfloor_finished  area_total_calc  area_live_finished  \
0                     548.0           1264.0              1264.0
1                     777.0            777.0               777.0
2                    1101.0           1101.0              1101.0
3                    1554.0           1554.0              1554.0
4                    1305.0           2415.0              2415.0

   area_total_finished  …  1432  1720  1722  200   34   38  6050   73  8800  \
0               1680.0  …   0.0   0.0   0.0  0.0  0.0  0.0   0.0  0.0   0.0
1               1680.0  …   0.0   0.0   0.0  0.0  0.0  0.0   0.0  0.0   0.0
2               1680.0  …   0.0   0.0   0.0  0.0  0.0  0.0   0.0  0.0   0.0
3               1680.0  …   0.0   0.0   0.0  0.0  0.0  0.0   0.0  0.0   0.0
4               1680.0  …   0.0   0.0   0.0  0.0  0.0  0.0   0.0  0.0   0.0

    96
0  0.0
1  0.0
2  0.0
3  0.0
4  0.0

[5 rows x 125 columns]
```

Préprocessing: Avant de réaliser des modèles sur nos données, on va d'abord les analyser, les regarder dans un premier temps, voir les features(attributs), les échelles de ses features, si y'a des valeurs manquantes.

Normalisation et Standardisation:

Le Feature Scaling permet de préparer les données quand elles ont des échelles différentes. Il permettra d'avoir de meilleurs modèles prédictifs.

Parmi les techniques du feature scaling, on retrouve la Standardisation et la Normalisation.

La normalisation:

afin que les valeurs des attributs soient inclus dans l'intervalle 0,1

La standardisation:

Standardize features by removing the mean and scaling to unit variance

The standard score of a sample x is calculated as:

z = (x - u) / s

where u is the mean of the training samples or zero if with_mean=False, and s is the standard deviation of the training samples or one if with_std=False. (source sckitlearn)

[79]:
```
"""standardization"""
object_cols = [col for col in merged.columns if merged[col].dtype!='float64']
merged.drop(object_cols, axis='columns', inplace=True)
print object_cols
merged.head()
print "statistiques avant standardisation"
print(merged.describe())
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_data = scaler.fit_transform(merged)



scaled_features_df = pd.DataFrame(scaled_data, index=merged.index,␣
↪columns=merged.columns)

print "statistiques après standardisation"
print scaled_features_df.describe()


from matplotlib import pyplot

sns.distplot(scaled_features_df['area_total_calc'], color = 'green', rug =␣
↪True, kde_kws = {'color': 'red', 'lw': 1})
#yplot.xlabel('Value')
#yplot.ylabel('Frequency')
pyplot.title('area_total_calc')
```

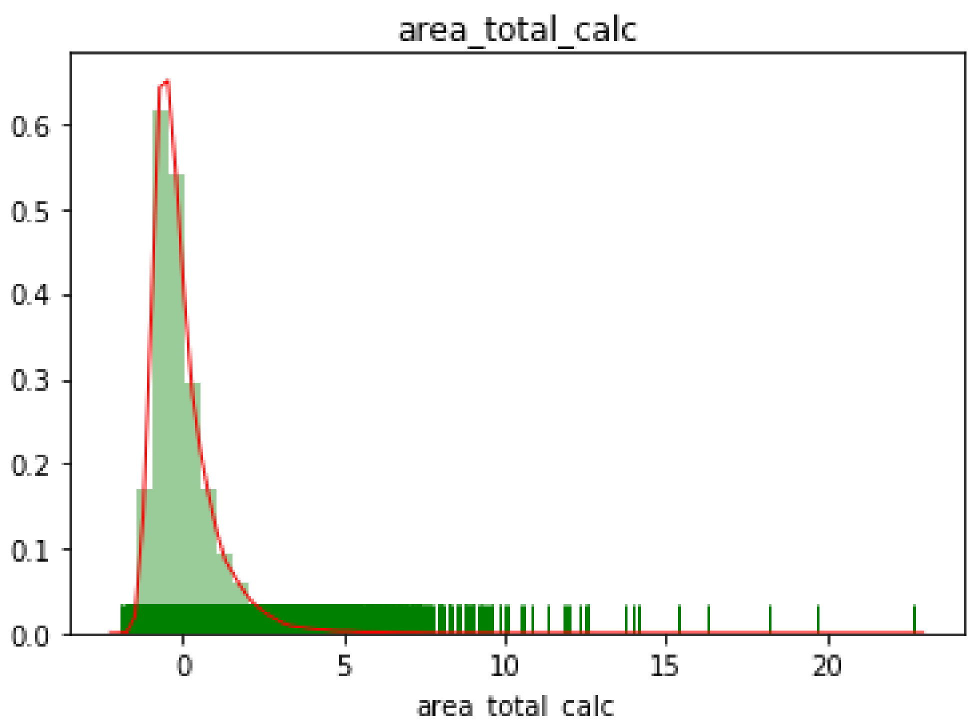
```
['id_parcel', 'flag_tub', 'zoning_landuse_county', 'zoning_property',
'tax_delinquency', 'transactiondate']
statistiques avant standardisation
             aircon  num_bathroom   num_bedroom       quality  \
count  90275.000000  90275.000000  90275.000000  90275.000000
mean       1.260271      2.279474      3.031869      6.088408
std        1.721860      1.004271      1.156436      1.664972
min        1.000000      0.000000      0.000000      1.000000
25%        1.000000      2.000000      2.000000      4.000000
50%        1.000000      2.000000      3.000000      7.000000
75%        1.000000      3.000000      4.000000      7.000000
max       13.000000     20.000000     16.000000     12.000000
```

```
       num_bathroom_calc  area_firstfloor_finished  area_total_calc  \
count       90275.000000              90275.000000     90275.000000
mean            2.305168                857.325206      1768.989078
std             0.970398                228.266810       926.048336
min             1.000000                 44.000000         2.000000
25%             2.000000                817.000000      1187.000000
50%             2.000000                817.000000      1535.000000
75%             3.000000                817.000000      2089.000000
max            20.000000               7625.000000     22741.000000

       area_live_finished  area_total_finished  area_unknown  …  \
count        90275.000000         90275.000000  90275.000000  …
mean          1717.183340          1707.639114    857.900316  …
std            894.258742           252.235211    234.135522  …
min              2.000000           560.000000     44.000000  …
25%           1190.000000          1680.000000    817.000000  …
50%           1476.000000          1680.000000    817.000000  …
75%           2013.000000          1680.000000    817.000000  …
max          20013.000000         22741.000000   8352.000000  …

         build_year     num_story  tax_building     tax_total  tax_year  \
count  90275.000000  90275.000000  9.027500e+04  9.027500e+04   90275.0
mean    1968.419540      1.100426  1.797563e+05  4.576714e+05    2015.0
std       23.695875      0.318951  2.087537e+05  5.548814e+05       0.0
min     1885.000000      1.000000  1.000000e+02  2.200000e+01    2015.0
25%     1953.000000      1.000000  8.149000e+04  1.990235e+05    2015.0
50%     1969.000000      1.000000  1.315070e+05  3.428720e+05    2015.0
75%     1987.000000      1.000000  2.100425e+05  5.405890e+05    2015.0
max     2015.000000      4.000000  9.948100e+06  2.775000e+07    2015.0

           tax_land   tax_property  tax_delinquency_year  censustractandblock  \
count  9.027500e+04   90275.000000          90275.000000         9.027500e+04
mean   2.783325e+05    5983.680847             13.988203         6.049076e+13
std    4.004942e+05    6838.745460              0.390536         2.041793e+11
min    2.200000e+01      49.080000              6.000000         6.037101e+13
25%    8.222750e+04    2872.470000             14.000000         6.037400e+13
50%    1.929600e+05    4542.440000             14.000000         6.037620e+13
75%    3.454150e+05    6900.600000             14.000000         6.059042e+13
max    2.450000e+07  321936.090000             99.000000         6.111009e+13

           logerror
count  90275.000000
mean       0.011457
std        0.161079
min       -4.605000
25%       -0.025300
50%        0.006000
```

```
75%        0.039200
max        4.737000

[8 rows x 42 columns]
statistiques après standardisation
             aircon  num_bathroom   num_bedroom       quality  \
count  9.027500e+04  9.027500e+04  9.027500e+04  9.027500e+04
mean  -4.565104e-17  1.760713e-16  5.399416e-17  2.110180e-16
std    1.000006e+00  1.000006e+00  1.000006e+00  1.000006e+00
min   -1.511579e-01 -2.269792e+00 -2.621751e+00 -3.056169e+00
25%   -1.511579e-01 -2.782868e-01 -8.922893e-01 -1.254327e+00
50%   -1.511579e-01 -2.782868e-01 -2.755836e-02  5.475152e-01
75%   -1.511579e-01  7.174659e-01  8.371726e-01  5.475152e-01
max    6.818087e+00  1.764526e+01  1.121394e+01  3.550585e+00

       num_bathroom_calc  area_firstfloor_finished  area_total_calc  \
count       9.027500e+04              9.027500e+04     9.027500e+04
mean       -4.250269e-17              5.694574e-17     8.819309e-17
std         1.000006e+00              1.000006e+00     1.000006e+00
min        -1.344990e+00             -3.563066e+00    -1.908107e+00
25%        -3.144786e-01             -1.766592e-01    -6.284686e-01
50%        -3.144786e-01             -1.766592e-01    -2.526762e-01
75%         7.160326e-01             -1.766592e-01     3.455680e-01
max         1.823472e+01              2.964825e+01     2.264690e+01

       area_live_finished  area_total_finished  area_unknown  …  \
count        9.027500e+04         9.027500e+04  9.027500e+04  …
mean        -2.707579e-17         1.213688e-16  2.046426e-16  …
std          1.000006e+00         1.000006e+00  1.000006e+00  …
min         -1.918005e+00        -4.549902e+00 -3.476212e+00  …
25%         -5.895232e-01        -1.095774e-01 -1.746875e-01  …
50%         -2.697035e-01        -1.095774e-01 -1.746875e-01  …
75%          3.307972e-01        -1.095774e-01 -1.746875e-01  …
max          2.045931e+01         8.338835e+01  3.200771e+01  …

         build_year     num_story  tax_building     tax_total  tax_year  \
count  9.027500e+04  9.027500e+04  9.027500e+04  9.027500e+04   90275.0
mean   9.970817e-16 -1.748907e-16  1.550561e-17 -4.718586e-17       0.0
std    1.000006e+00  1.000006e+00  1.000006e+00  1.000006e+00       0.0
min   -3.520444e+00 -3.148669e-01 -8.606183e-01 -8.247743e-01       0.0
25%   -6.507304e-01 -3.148669e-01 -4.707308e-01 -4.661345e-01       0.0
50%    2.449637e-02 -3.148669e-01 -2.311313e-01 -2.068912e-01       0.0
75%    7.841265e-01 -3.148669e-01  1.450821e-01  1.494338e-01       0.0
max    1.965773e+00  9.091027e+00  4.679389e+01  4.918615e+01       0.0

           tax_land  tax_property  tax_delinquency_year  censustractandblock  \
count  9.027500e+04  9.027500e+04          9.027500e+04         9.027500e+04
mean  -6.513144e-18 -1.930232e-16         -6.757732e-16         4.736650e-14
```

```
std     1.000006e+00  1.000006e+00          1.000006e+00          1.000006e+00
min    -6.949215e-01 -8.677957e-01         -2.045458e+01         -5.864722e-01
25%    -4.896602e-01 -4.549413e-01          3.020811e-02         -5.718235e-01
50%    -2.131690e-01 -2.107475e-01          3.020811e-02         -5.610437e-01
75%     1.675003e-01  1.340778e-01          3.020811e-02          4.881393e-01
max     6.047979e+01  4.620060e+01          2.176811e+02          3.033308e+00

           logerror
count  9.027500e+04
mean  -7.516680e-18
std    1.000006e+00
min   -2.865977e+01
25%   -2.281952e-01
50%   -3.387937e-02
75%    1.722320e-01
max    2.933699e+01

[8 rows x 42 columns]
```

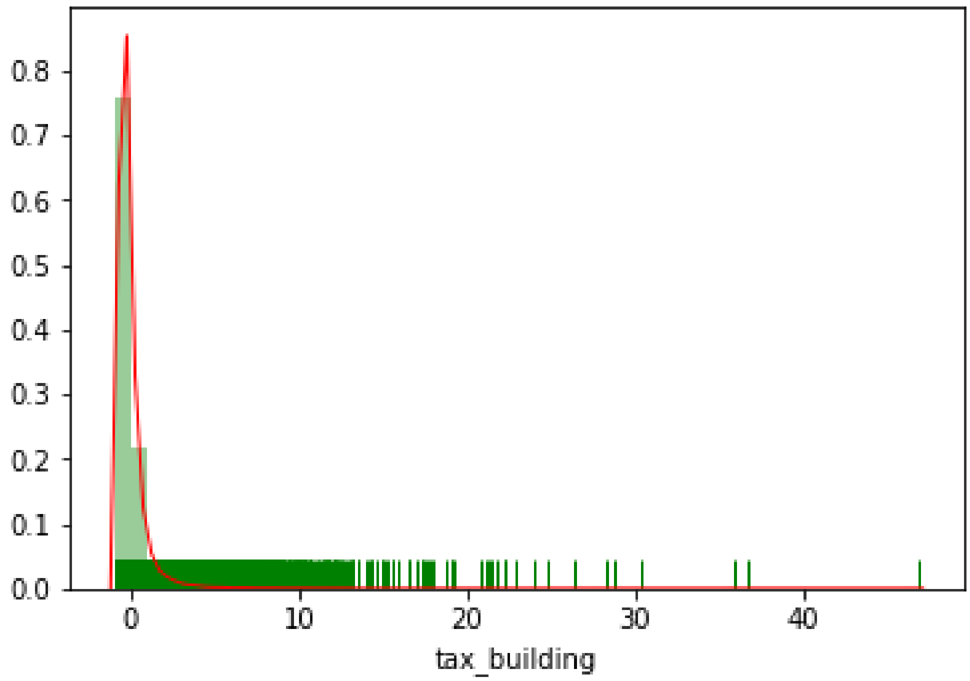
[79]: Text(0.5,1,'area_total_calc')



Distributions de quelques attributs après standardisation des données

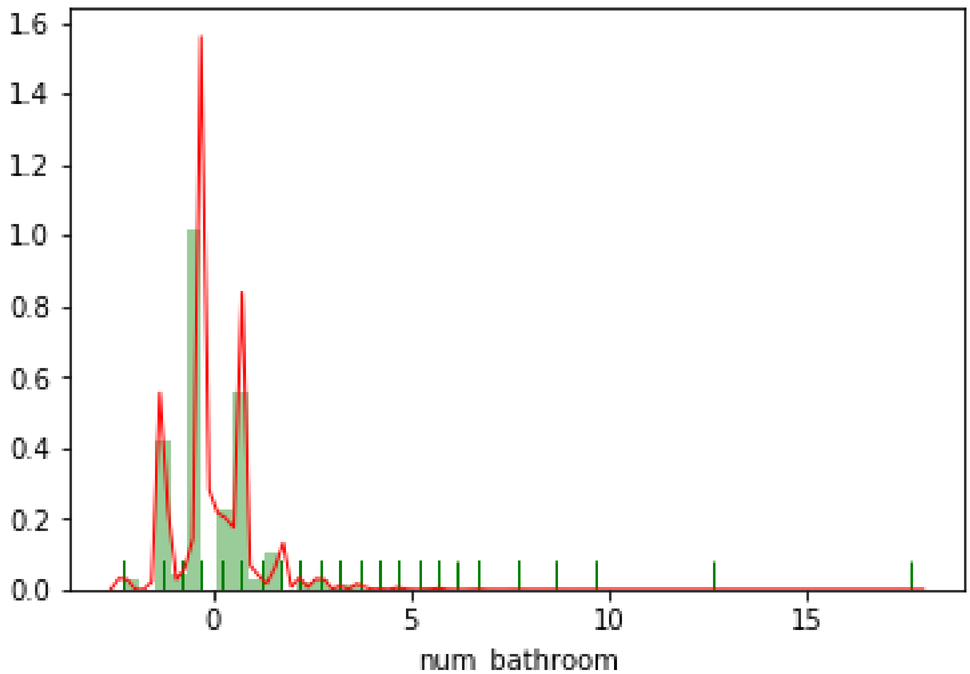
[80]:
```
sns.distplot(scaled_features_df['tax_building'], color = 'green', rug = True,␣
↪kde_kws = {'color': 'red', 'lw': 1})
```

[80]: <matplotlib.axes._subplots.AxesSubplot at 0x7f659cbdab50>



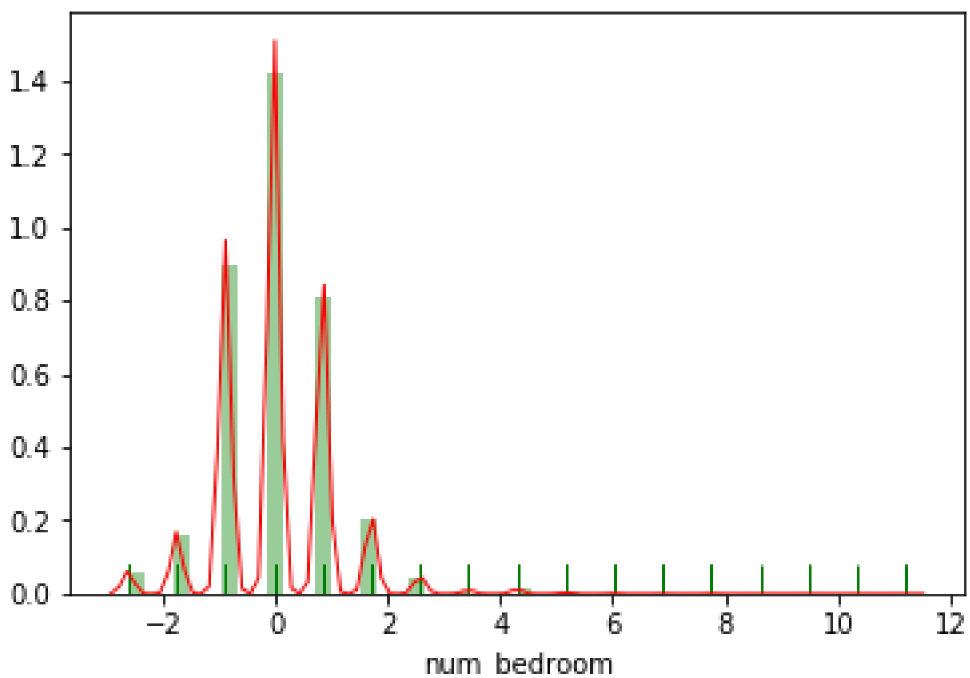
[81]:
```
sns.distplot(scaled_features_df['num_bathroom'], color = 'green', rug = True,␣
↪kde_kws = {'color': 'red', 'lw': 1})
```

[81]: <matplotlib.axes._subplots.AxesSubplot at 0x7f654852b9d0>



[82]:
```
sns.distplot(scaled_features_df['num_bedroom'], color = 'green', rug = True,␣
↪kde_kws = {'color': 'red', 'lw': 1})
```

[82]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6513c91cd0>



[83]:
```python
"""utilisation de PCA pour améliorer les performances des modèles"""
cle = 'logerror'
columns=[]
#columns = [ key  if cle != key else pass for key  in merged.keys()]
columns = [key for key  in merged.keys() if key not in cle]

from sklearn.model_selection import train_test_split
#répartition des données  en ensmble de test et ensmble d'entrainement
train_data, test_data, train_lbl, test_lbl = train_test_split(merged[columns],␣
 ↪merged['logerror'],
                                                              test_size=1/7.0,
                                                              random_state=0)



from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# entrainnement sur le training set.
scaler.fit(train_data)
# transformation du jeu de données
train_scaled = scaler.transform(train_data)
test_scaled = scaler.transform(test_data)

from sklearn.decomposition import PCA
# Make an instance of the Model
pca = PCA(.95)

#pca = PCA(n_components=2)
""". scikit-learn va choisir le nombre minimum de
composantes principales afin que 95% de la variance est retenue"""

pca.fit(train_scaled)

print 'nombre de composantes principales choisis afin d\'expliquer 95% de la␣
 ↪variance:'
print (pca.n_components_)
print "\n"*2

train_pca = pca.transform(train_scaled)
test_pca = pca.transform(test_scaled)


print ('variance expliquée:')
print pca.explained_variance_,
```

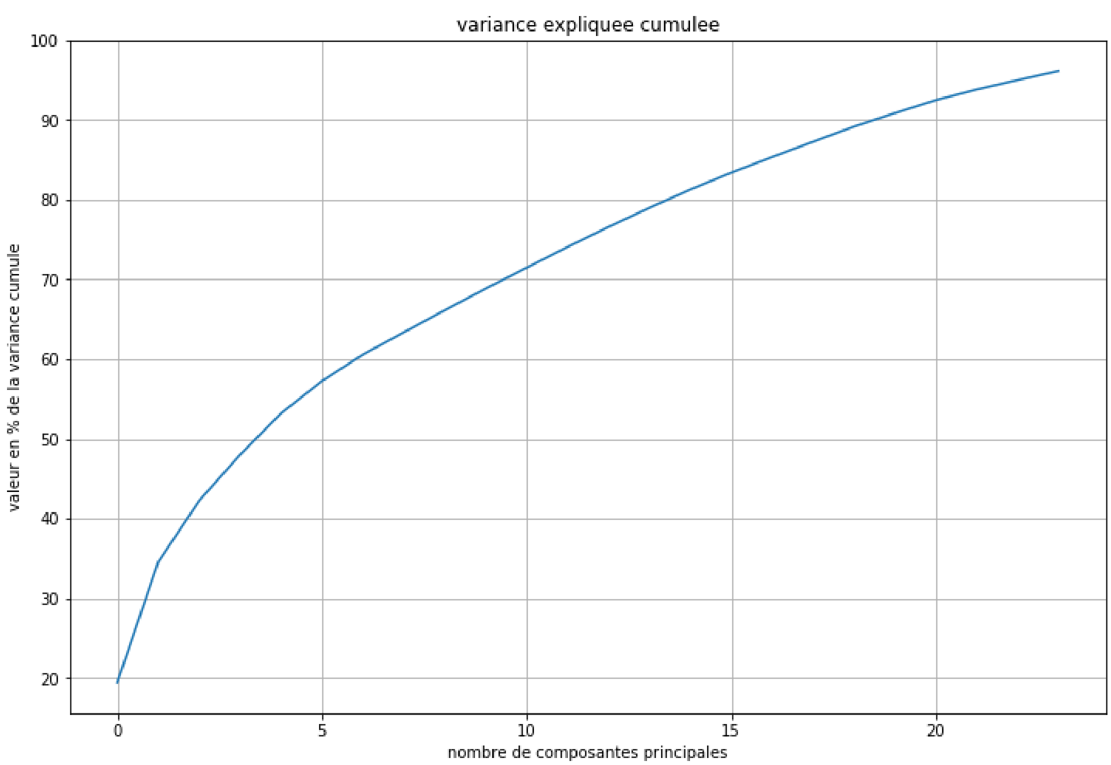
```
print "\n" * 2
print ('pourcentage de la variance expliquée')
print pca.explained_variance_ratio_.cumsum()
print "\n" * 2
print  "la variance expliquée cumulée pour chaque composante principale"
variances_expliquee = np.cumsum(np.round(pca.explained_variance_ratio_,␣
 ↪decimals=4)*100)
#print type(variances_expliquee.tolist())
print np.cumsum(np.round(pca.explained_variance_ratio_, decimals=4)*100)
print "\n" * 2

plt.figure(figsize = (12, 8))
plt.title("variance expliquee cumulee")
plt.xlabel("nombre de composantes principales")
plt.ylabel("valeur en % de la variance cumule")
ax = plt.axes()
ax.plot(variances_expliquee)
fig.figsize=(25, 10)
plt.grid()






from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
k = 10
X = train_pca # Matrice colonne plutôt que vecteur ligne
y = train_lbl

from sklearn.linear_model import LinearRegression
reg_lin = LinearRegression(fit_intercept=True)

reg_lin.fit(train_pca[:100], train_lbl[:100])


y_pred = reg_lin.predict(train_pca[:100])
y_pred = np.around(y_pred, decimals=3)
y_true = np.around(train_lbl[:100].to_numpy(), decimals = 3)
#explained_variance_score( y_true, y_pred)

from sklearn.metrics import mean_squared_error
#mean_squared_error(y_true, y_pred)
```

```
nombre de composantes principales choisis afin d'expliquer 95% de la variance:
24
```

```
variance expliquée:
[7.39556707 5.73241246 2.92370914 2.18931493 1.97391133 1.537239
 1.26282073 1.05876163 1.05185188 1.02926814 0.99467017 0.98928167
 0.95288114 0.91293999 0.87061686 0.80693245 0.74368644 0.729543
 0.72212961 0.64032133 0.61484552 0.50893111 0.44855487 0.43269835]
```

```
pourcentage de la variance expliquée
[0.19461767 0.34546868 0.4224074  0.48002021 0.53196457 0.57241771
 0.6056494  0.63351119 0.66119114 0.6882768  0.71445199 0.74048538
 0.76556088 0.7895853  0.81249598 0.83373077 0.85330121 0.87249946
 0.89150263 0.90835297 0.92453291 0.93792566 0.94972958 0.96111623]
```

```
la variance expliquée cumulée pour chaque composante principale
[19.46 34.55 42.24 48.   53.19 57.24 60.56 63.35 66.12 68.83 71.45 74.05
 76.56 78.96 81.25 83.37 85.33 87.25 89.15 90.84 92.46 93.8  94.98 96.12]
```



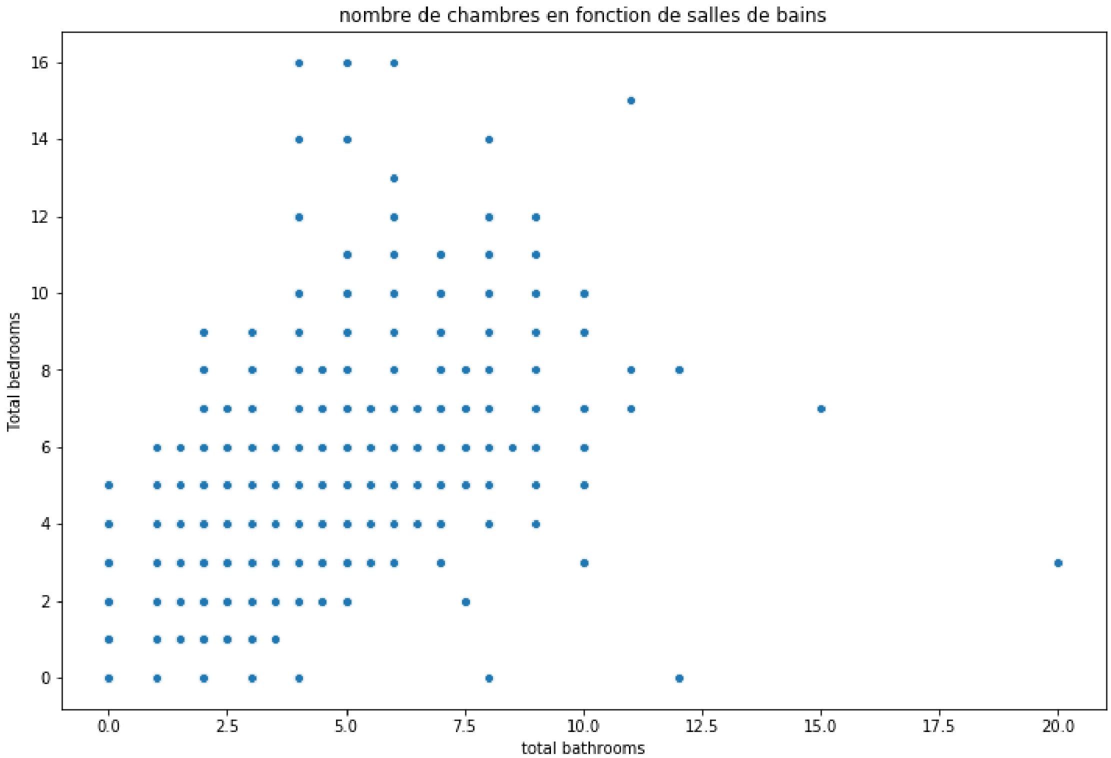
[ ]:
```
"""Extraction des attributs jugés non pertinents pour nos analyses"""
merged.drop(['architectural_style', 'area_basement', 'framing',␣
↪'deck','area_base', 'area_liveperi_finished',
            'area_base', 'pooltypeid10', 'pooltypeid2', 'story', 'material',␣
↪'area_shed', 'flag_fireplace' ], axis='columns', inplace=True)
```

[84]:
```
print merged.keys()
```

```
Index([u'aircon', u'num_bathroom', u'num_bedroom', u'quality',
       u'num_bathroom_calc', u'area_firstfloor_finished', u'area_total_calc',
       u'area_live_finished', u'area_total_finished', u'area_unknown', u'fips',
       u'num_fireplace', u'num_bath', u'num_garage', u'area_garage',
       u'heating', u'latitude', u'longitude', u'area_lot', u'num_pool',
       u'area_pool', u'pooltypeid7', u'zoning_landuse',
       u'rawcensustractandblock', u'region_city', u'region_county',
       u'region_neighbor', u'region_zip', u'num_room', u'num_75_bath',
       u'num_unit', u'area_patio', u'build_year', u'num_story',
       u'tax_building', u'tax_total', u'tax_year', u'tax_land',
       u'tax_property', u'tax_delinquency_year', u'censustractandblock',
       u'logerror'],
      dtype='object')
```

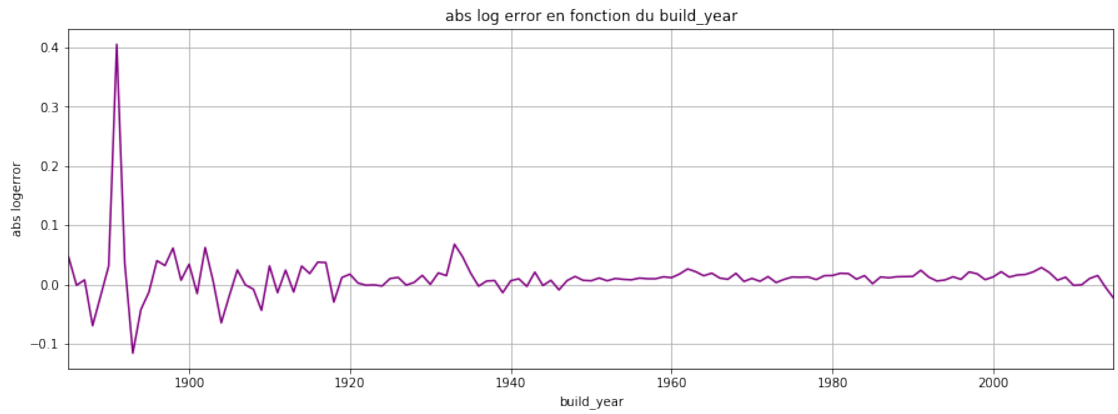
[85]:
```
"""etudes de l'importance des variables"""
""" liens entre les attributs"""
#variable yearbuild
import seaborn as sns

plt.figure(figsize = (12, 8))
sns.scatterplot(data = merged,
               x = 'num_bathroom',
               y = 'num_bedroom')
plt.title("nombre de chambres en fonction de salles de bains")
plt.xlabel("total bathrooms")
plt.ylabel("Total bedrooms")
```

[85]:
```
Text(0,0.5,'Total bedrooms')
```



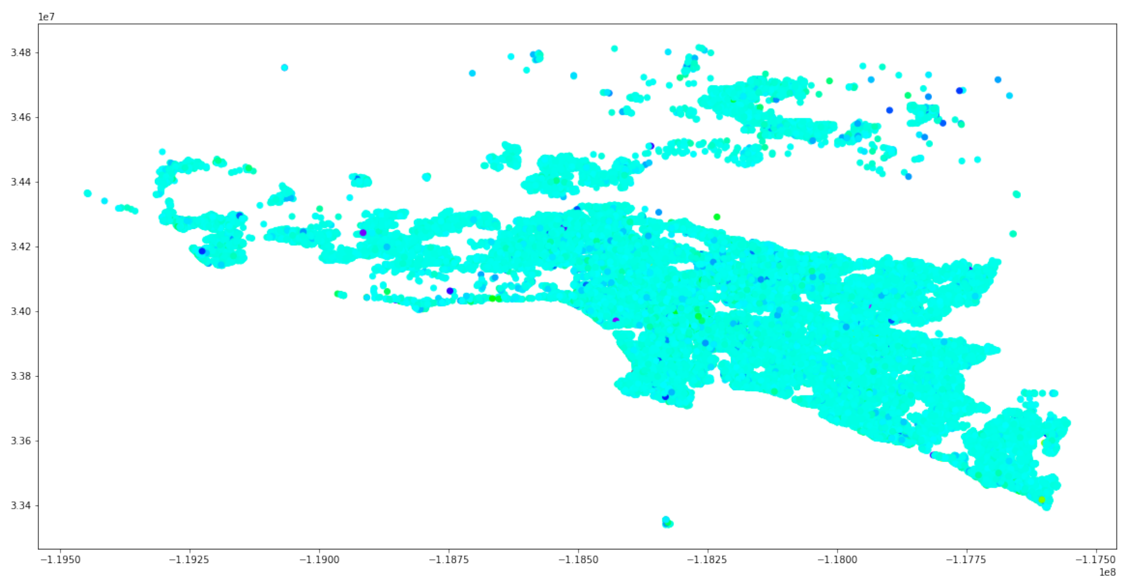
[86]:
```
"""etudes de l'importance des variables"""
""" liens entre certaines caractèristiques et le logerror"""
v =merged.groupby(merged['build_year'])['logerror'].mean()
plt.figure(figsize = (12, 8))

ax =v.plot.line(x=v.keys(), figsize=(15, 5), color= 'purple', grid='true')
plt.title("abs log error en fonction du build_year")
plt.xlabel("build_year")
plt.ylabel("abs logerror")

#print merged['build_year'][100]
```

[86]: Text(0,0.5,'abs logerror')



[87]:
```
#plot de lattitude et longitude en coloration, on a le logerror

plt.figure(figsize = (20, 10))
plt.scatter(x=merged['longitude'], y=merged['latitude'], c=merged['logerror'],␣
 ↪cmap='hsv')
```

[87]: <matplotlib.collections.PathCollection at 0x7f651167be90>



[88]:
```
x_cols = [col for col in merged.columns if col not in ['logerror'] if␣
 ↪merged[col].dtype=='float64']
print x_cols
df1 = merged.ix[:, x_cols]
print df1
```

```
['aircon', 'num_bathroom', 'num_bedroom', 'quality', 'num_bathroom_calc',
'area_firstfloor_finished', 'area_total_calc', 'area_live_finished',
'area_total_finished', 'area_unknown', 'fips', 'num_fireplace', 'num_bath',
'num_garage', 'area_garage', 'heating', 'latitude', 'longitude', 'area_lot',
'num_pool', 'area_pool', 'pooltypeid7', 'zoning_landuse',
'rawcensustractandblock', 'region_city', 'region_county', 'region_neighbor',
'region_zip', 'num_room', 'num_75_bath', 'num_unit', 'area_patio', 'build_year',
'num_story', 'tax_building', 'tax_total', 'tax_year', 'tax_land',
'tax_property', 'tax_delinquency_year', 'censustractandblock']
       aircon  num_bathroom  num_bedroom  quality  num_bathroom_calc  \
0         1.0           2.5          3.0      7.0                2.5
1         1.0           1.0          2.0      7.0                1.0
2         1.0           2.0          3.0      7.0                2.0
3         1.0           1.5          2.0      7.0                1.5
4         1.0           2.5          4.0      7.0                2.5
5         1.0           2.5          4.0      7.0                2.5
6         1.0           2.0          3.0      7.0                2.0
7         1.0           2.5          5.0      7.0                2.5
8         1.0           2.0          3.0      7.0                2.0
9         1.0           1.0          3.0      7.0                1.0
10        1.0           2.0          4.0      7.0                2.0
11        1.0           2.5          4.0      7.0                2.5
12        1.0           2.0          5.0      7.0                2.0
13        1.0           3.0          3.0      7.0                3.0
14        1.0           2.5          5.0      7.0                2.5
15        1.0           2.0          4.0      7.0                2.0
16        1.0           2.5          5.0      7.0                2.5
17        1.0           1.0          2.0      7.0                1.0
18        1.0           2.5          2.0      7.0                2.5
19        1.0           2.0          2.0      7.0                2.0
20        1.0           1.0          2.0      7.0                1.0
21        1.0           3.0          5.0      7.0                3.0
22        1.0           2.5          4.0      7.0                2.5
23        1.0           2.5          3.0      7.0                2.5
24        1.0           2.0          3.0      7.0                2.0
25        1.0           3.0          5.0      7.0                3.0
26        1.0           3.0          3.0      4.0                3.0
27        1.0           2.0          3.0      7.0                2.0
28        1.0           2.0          4.0      7.0                2.0
29        1.0           3.0          4.0      4.0                3.0
30        1.0           7.0          6.0     10.0                7.0
31        1.0           3.0          2.0      4.0                3.0
…       …            …            …        …                 …
90243     1.0           4.5          4.0      7.0                4.5
90244     1.0           2.0          3.0      4.0                2.0
90245     1.0           4.0          4.0      1.0                4.0
90246     1.0           3.0          2.0     10.0                3.0
90247     1.0           2.0          1.0      4.0                2.0
```

90248	1.0	2.0	2.0	1.0	2.0
90249	1.0	3.5	3.0	7.0	3.5
90250	1.0	2.0	3.0	7.0	2.0
90251	1.0	2.0	2.0	7.0	2.0
90252	1.0	2.5	1.0	7.0	2.5
90253	1.0	3.0	4.0	7.0	3.0
90254	1.0	3.0	2.0	1.0	3.0
90255	1.0	1.0	1.0	7.0	1.0
90256	1.0	4.0	5.0	7.0	4.0
90257	1.0	2.0	2.0	4.0	2.0
90258	1.0	2.0	2.0	4.0	2.0
90259	1.0	1.0	1.0	4.0	1.0
90260	1.0	3.0	4.0	4.0	3.0
90261	1.0	3.0	3.0	4.0	3.0
90262	1.0	2.0	0.0	7.0	2.0
90263	1.0	3.0	2.0	4.0	3.0
90264	1.0	1.0	1.0	1.0	1.0
90265	1.0	2.0	2.0	10.0	2.0
90266	1.0	3.0	4.0	7.0	3.0
90267	1.0	2.5	2.0	7.0	2.5
90268	1.0	2.0	2.0	7.0	2.0
90269	1.0	3.0	4.0	7.0	3.0
90270	1.0	3.0	4.0	4.0	3.0
90271	1.0	2.0	3.0	7.0	2.0
90272	1.0	2.0	2.0	4.0	2.0
90273	1.0	2.5	3.0	7.0	2.5
90274	1.0	2.5	3.0	7.0	2.5

	area_firstfloor_finished	area_total_calc	area_live_finished	\
0	548.0	1264.0	1264.0	
1	777.0	777.0	777.0	
2	1101.0	1101.0	1101.0	
3	1554.0	1554.0	1554.0	
4	1305.0	2415.0	2415.0	
5	1303.0	2882.0	2882.0	
6	1772.0	1772.0	1772.0	
7	1240.0	2632.0	2632.0	
8	1292.0	1292.0	1292.0	
9	804.0	1385.0	1385.0	
10	1260.0	1260.0	1260.0	
11	1448.0	2735.0	2735.0	
12	2085.0	2085.0	2085.0	
13	906.0	1508.0	1508.0	
14	977.0	1958.0	1958.0	
15	1120.0	1687.0	1687.0	
16	1236.0	2232.0	2232.0	
17	435.0	834.0	834.0	
18	691.0	1361.0	1361.0	

19	917.0	917.0	917.0
20	817.0	907.0	907.0
21	2524.0	2524.0	2524.0
22	2400.0	2400.0	2400.0
23	1137.0	2113.0	2113.0
24	2297.0	2297.0	2297.0
25	1996.0	1996.0	1996.0
26	817.0	2445.0	2445.0
27	817.0	1160.0	1160.0
28	817.0	1570.0	1570.0
29	817.0	2863.0	2863.0
30	817.0	6610.0	6610.0
31	817.0	1394.0	1394.0
…	…	…	…
90243	817.0	4365.0	4365.0
90244	817.0	1565.0	1565.0
90245	817.0	3568.0	3568.0
90246	817.0	1656.0	1656.0
90247	817.0	1450.0	1450.0
90248	817.0	1240.0	1240.0
90249	817.0	2972.0	2972.0
90250	817.0	1456.0	1456.0
90251	817.0	1252.0	1252.0
90252	817.0	1680.0	1680.0
90253	1587.0	3096.0	3096.0
90254	817.0	1110.0	1110.0
90255	817.0	728.0	728.0
90256	817.0	3308.0	3308.0
90257	817.0	1440.0	1440.0
90258	817.0	1550.0	1550.0
90259	817.0	860.0	860.0
90260	817.0	2781.0	2781.0
90261	817.0	1432.0	1432.0
90262	817.0	2140.0	2140.0
90263	817.0	1060.0	1060.0
90264	817.0	918.0	918.0
90265	817.0	1492.0	1492.0
90266	440.0	1771.0	1771.0
90267	817.0	1638.0	1638.0
90268	817.0	1308.0	1308.0
90269	817.0	1713.0	1713.0
90270	817.0	2068.0	2068.0
90271	817.0	1352.0	1352.0
90272	817.0	860.0	860.0
90273	817.0	2268.0	2268.0
90274	817.0	1812.0	1812.0

```
       area_total_finished  area_unknown  …  area_patio  build_year  \
```

0	1680.0	548.0	…	128.0	1986.0
1	1680.0	777.0	…	198.0	1990.0
2	1680.0	1101.0	…	240.0	1956.0
3	1680.0	1554.0	…	240.0	1965.0
4	1680.0	1305.0	…	240.0	1984.0
5	1680.0	1303.0	…	240.0	1980.0
6	1680.0	1772.0	…	1045.0	1978.0
7	1680.0	1240.0	…	180.0	1971.0
8	1680.0	1292.0	…	304.0	1979.0
9	1680.0	804.0	…	240.0	1950.0
10	1680.0	1260.0	…	240.0	1969.0
11	1680.0	1448.0	…	700.0	1984.0
12	1680.0	2085.0	…	240.0	1962.0
13	1680.0	906.0	…	240.0	1981.0
14	1680.0	977.0	…	243.0	1964.0
15	1680.0	1120.0	…	240.0	1961.0
16	1680.0	1236.0	…	280.0	1965.0
17	1680.0	435.0	…	240.0	1976.0
18	1680.0	691.0	…	240.0	1980.0
19	1680.0	917.0	…	154.0	1985.0
20	1680.0	817.0	…	240.0	1985.0
21	1680.0	2524.0	…	240.0	1963.0
22	1680.0	2400.0	…	698.0	1969.0
23	1680.0	1137.0	…	275.0	1993.0
24	1680.0	2297.0	…	84.0	1988.0
25	1680.0	1996.0	…	204.0	1970.0
26	1680.0	817.0	…	240.0	1982.0
27	1680.0	817.0	…	240.0	1960.0
28	1680.0	817.0	…	240.0	1959.0
29	1680.0	817.0	…	240.0	1963.0
30	1680.0	817.0	…	240.0	1997.0
31	1680.0	817.0	…	240.0	1998.0
…	…	…	…	…	…
90243	1680.0	817.0	…	240.0	2005.0
90244	1680.0	817.0	…	240.0	2004.0
90245	1680.0	817.0	…	240.0	2004.0
90246	1680.0	817.0	…	240.0	2003.0
90247	1680.0	817.0	…	240.0	2005.0
90248	1680.0	817.0	…	240.0	2006.0
90249	1680.0	817.0	…	240.0	2005.0
90250	1680.0	817.0	…	240.0	1988.0
90251	1680.0	817.0	…	240.0	1988.0
90252	1680.0	817.0	…	240.0	2004.0
90253	1680.0	1587.0	…	240.0	2006.0
90254	1680.0	817.0	…	240.0	2006.0
90255	1680.0	817.0	…	240.0	2005.0
90256	1680.0	817.0	…	240.0	2007.0
90257	1680.0	817.0	…	240.0	2007.0

90258	1680.0	817.0	…	240.0	2013.0
90259	1680.0	817.0	…	240.0	2007.0
90260	1680.0	817.0	…	240.0	2006.0
90261	1680.0	817.0	…	240.0	2005.0
90262	1680.0	817.0	…	240.0	1928.0
90263	1680.0	817.0	…	240.0	2008.0
90264	1680.0	817.0	…	240.0	2004.0
90265	1680.0	817.0	…	240.0	2006.0
90266	1680.0	440.0	…	240.0	2007.0
90267	1680.0	817.0	…	240.0	2007.0
90268	1680.0	817.0	…	240.0	2007.0
90269	1680.0	817.0	…	240.0	2007.0
90270	1680.0	817.0	…	240.0	2008.0
90271	1680.0	817.0	…	240.0	1956.0
90272	1680.0	817.0	…	240.0	2011.0
90273	1680.0	817.0	…	240.0	2012.0
90274	1680.0	817.0	…	240.0	2013.0

	num_story	tax_building	tax_total	tax_year	tax_land	tax_property	\
0	2.0	115087.0	191811.0	2015.0	76724.0	2015.06	
1	1.0	143809.0	239679.0	2015.0	95870.0	2581.30	
2	1.0	33619.0	47853.0	2015.0	14234.0	591.64	
3	1.0	45609.0	62914.0	2015.0	17305.0	682.78	
4	2.0	277000.0	554000.0	2015.0	277000.0	5886.92	
5	2.0	222070.0	289609.0	2015.0	67539.0	3110.44	
6	1.0	185000.0	526000.0	2015.0	341000.0	5632.20	
7	2.0	342611.0	571086.0	2015.0	228475.0	6109.94	
8	1.0	231297.0	462594.0	2015.0	231297.0	5026.40	
9	1.0	134251.0	268502.0	2015.0	134251.0	3217.06	
10	1.0	42257.0	61453.0	2015.0	19196.0	702.40	
11	2.0	239850.0	399742.0	2015.0	159892.0	4595.36	
12	1.0	230000.0	657000.0	2015.0	427000.0	6991.06	
13	2.0	142797.0	407991.0	2015.0	265194.0	4267.96	
14	2.0	136437.0	201400.0	2015.0	64963.0	2512.42	
15	1.0	170705.0	311415.0	2015.0	140710.0	3763.38	
16	2.0	225950.0	451900.0	2015.0	225950.0	5185.66	
17	2.0	94858.0	158096.0	2015.0	63238.0	1991.34	
18	2.0	150363.0	300726.0	2015.0	150363.0	3216.88	
19	1.0	126000.0	256000.0	2015.0	130000.0	2674.64	
20	1.0	122951.0	204919.0	2015.0	81968.0	2212.54	
21	1.0	424187.0	706979.0	2015.0	282792.0	7428.58	
22	1.0	265054.0	544564.0	2015.0	279510.0	5744.52	
23	2.0	376126.0	626875.0	2015.0	250749.0	7249.80	
24	1.0	324000.0	926000.0	2015.0	602000.0	9867.14	
25	1.0	397792.0	662986.0	2015.0	265194.0	8644.28	
26	1.0	436551.0	581388.0	2015.0	144837.0	7170.22	
27	1.0	105045.0	437584.0	2015.0	332539.0	5421.96	
28	1.0	115379.0	397138.0	2015.0	281759.0	5097.78	

29	1.0	358711.0	593502.0	2015.0	234791.0	7475.21
30	1.0	1333515.0	2148058.0	2015.0	814543.0	24878.86
31	1.0	203426.0	460937.0	2015.0	257511.0	5550.36
…	…	…	…	…	…	…
90243	1.0	1005655.0	1349000.0	2015.0	343345.0	23257.66
90244	1.0	135300.0	212000.0	2015.0	76700.0	3356.49
90245	1.0	506900.0	836600.0	2015.0	329700.0	12204.90
90246	1.0	467787.0	752472.0	2015.0	284685.0	9145.99
90247	1.0	220496.0	389351.0	2015.0	168855.0	4406.77
90248	1.0	352000.0	503000.0	2015.0	151000.0	6107.51
90249	1.0	420279.0	1306000.0	2015.0	885721.0	14773.94
90250	1.0	174840.0	332997.0	2015.0	158157.0	4367.26
90251	1.0	183644.0	351000.0	2015.0	167356.0	4565.96
90252	1.0	259927.0	535000.0	2015.0	275073.0	6315.96
90253	2.0	421000.0	803000.0	2015.0	382000.0	15003.80
90254	1.0	222000.0	518000.0	2015.0	296000.0	7890.69
90255	1.0	152749.0	223397.0	2015.0	70648.0	3859.80
90256	1.0	506199.0	802944.0	2015.0	296745.0	9079.58
90257	1.0	841483.0	1019979.0	2015.0	178496.0	12915.44
90258	1.0	374842.0	645136.0	2015.0	270294.0	9785.82
90259	1.0	475900.0	571000.0	2015.0	95100.0	7247.35
90260	1.0	209020.0	339656.0	2015.0	130636.0	10735.74
90261	1.0	238063.0	509989.0	2015.0	271926.0	6119.54
90262	1.0	270404.0	338004.0	2015.0	67600.0	4406.28
90263	1.0	168854.0	297946.0	2015.0	129092.0	3694.31
90264	1.0	309943.0	419839.0	2015.0	109896.0	5181.65
90265	1.0	441137.0	710668.0	2015.0	269531.0	8814.80
90266	3.0	174290.0	348580.0	2015.0	174290.0	3740.30
90267	1.0	206240.0	522229.0	2015.0	315989.0	5259.36
90268	1.0	223410.0	490808.0	2015.0	267398.0	5005.48
90269	1.0	276843.0	433819.0	2015.0	156976.0	5179.82
90270	1.0	388582.0	596082.0	2015.0	207500.0	7335.81
90271	1.0	86209.0	178408.0	2015.0	92199.0	2441.74
90272	1.0	129000.0	420000.0	2015.0	291000.0	5070.41
90273	1.0	389474.0	1215816.0	2015.0	826342.0	12508.30
90274	1.0	237048.0	471286.0	2015.0	234238.0	5470.12

	tax_delinquency_year	censustractandblock
0	14.0	6.111002e+13
1	14.0	6.111002e+13
2	14.0	6.111001e+13
3	14.0	6.111001e+13
4	14.0	6.111001e+13
5	14.0	6.111005e+13
6	14.0	6.111006e+13
7	14.0	6.111006e+13
8	14.0	6.111005e+13
9	14.0	6.111004e+13

10	14.0	6.111005e+13
11	14.0	6.111008e+13
12	14.0	6.111007e+13
13	14.0	6.111007e+13
14	14.0	6.111008e+13
15	14.0	6.111008e+13
16	14.0	6.111008e+13
17	14.0	6.111008e+13
18	14.0	6.111007e+13
19	14.0	6.111007e+13
20	14.0	6.111007e+13
21	14.0	6.111007e+13
22	14.0	6.111007e+13
23	14.0	6.111007e+13
24	14.0	6.111007e+13
25	14.0	6.111007e+13
26	14.0	6.037135e+13
27	14.0	6.037135e+13
28	14.0	6.037135e+13
29	14.0	6.037137e+13
30	14.0	6.037800e+13
31	14.0	6.037800e+13
…	…	…
90243	14.0	6.059032e+13
90244	14.0	6.037901e+13
90245	14.0	6.037920e+13
90246	14.0	6.037268e+13
90247	14.0	6.037311e+13
90248	14.0	6.037206e+13
90249	14.0	6.059099e+13
90250	14.0	6.059022e+13
90251	14.0	6.059022e+13
90252	14.0	6.059110e+13
90253	14.0	6.111004e+13
90254	14.0	6.037602e+13
90255	14.0	6.059086e+13
90256	14.0	6.059001e+13
90257	14.0	6.037208e+13
90258	14.0	6.037920e+13
90259	14.0	6.037208e+13
90260	14.0	6.037920e+13
90261	14.0	6.037481e+13
90262	14.0	6.037191e+13
90263	14.0	6.037199e+13
90264	14.0	6.037464e+13
90265	14.0	6.037274e+13
90266	14.0	6.111001e+13
90267	14.0	6.059063e+13

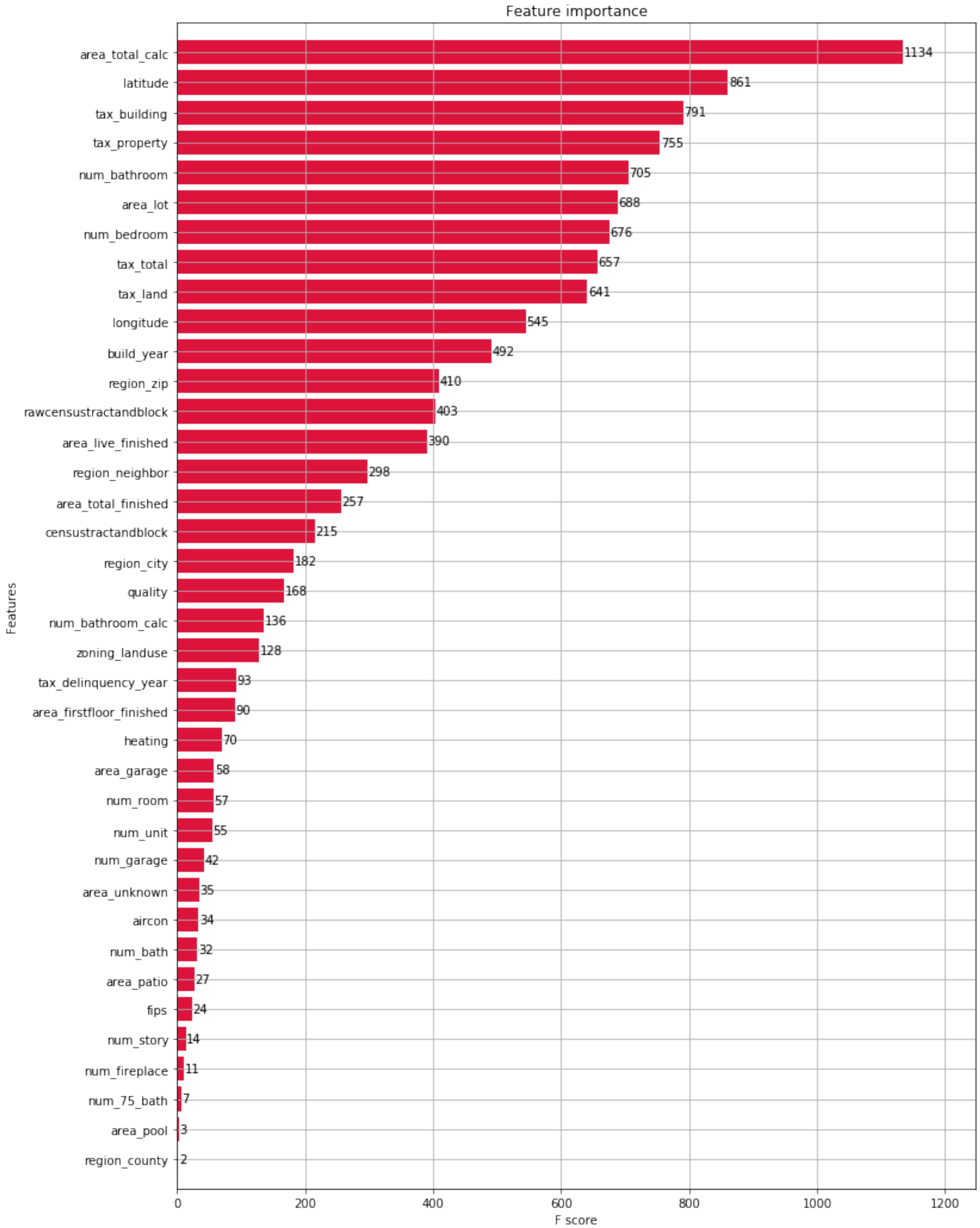
```
90268                  14.0          6.059063e+13
90269                  14.0          6.059075e+13
90270                  14.0          6.037201e+13
90271                  14.0          6.037407e+13
90272                  14.0          6.037191e+13
90273                  14.0          6.037920e+13
90274                  14.0          6.037920e+13

[90275 rows x 41 columns]
```

L'algorithme xgboost, pour xtreme gradient boosting est une implémentation d'algorithme d'arbres de boosting du gradient

définition: Le Boosting de Gradient est un algorithme d'apprentissage supervisé dont le principe et de combiner les résultats d'un ensemble de modèles plus simple et plus faibles afin de fournir une meilleur prédiction.

L'algorithme va donc combiner plusieurs modèles et obtenir un seul résultats il est très utilisés dans les compétitions de ML, et notamment le plus utilisés dans les notebooks de ce challenge, et il permet de fournir plusieurs hyperparametre

[89]:
```
"""importance des caractèristiques, méthode xgboost sans tenir compte des␣
↪valeur non numérique (exclusion des autres types)"""
import xgboost as xgb
#paramétrage qui permet le non overfiting en choisiant la longeur des arbres et␣
↪le taux d'apprentissage a chaque ittération
# pour éviter l'overfiting
xgb_params = {
    'eta': 0.05,
    'max_depth': 12,
    'subsample': 0.7,
    'colsample_bytree': 0.7,
    'objective': 'reg:linear',
    'silent': 1,
    'seed' : 0
}
dtrain = xgb.DMatrix(df1, merged['logerror'], feature_names=df1.columns.values)
model = xgb.train(dict(xgb_params, silent=0), dtrain, num_boost_round=50)

# classement des attributs #
fig, ax = plt.subplots(figsize=(12,18))
xgb.plot_importance(model, max_num_features=50, height=0.8, ax=ax,␣
↪color='crimson')
plt.show()
```



[42]:
```
feature_names=df1.columns.values
print columns
print len(columns)
```

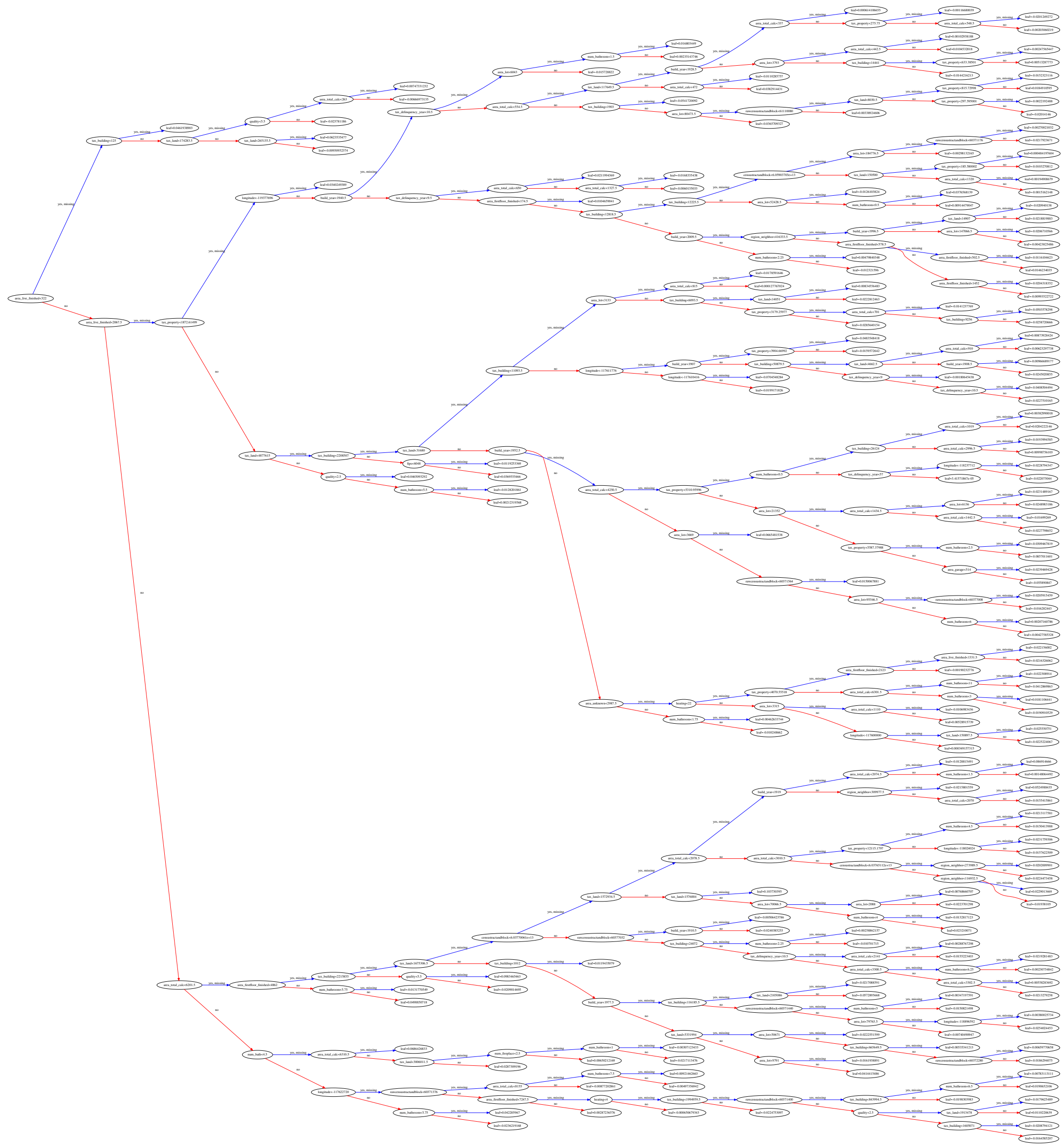
```
['aircon', 'num_bathroom', 'num_bedroom', 'quality', 'num_bathroom_calc',
```

```
'area_firstfloor_finished', 'area_total_calc', 'area_live_finished',
'area_total_finished', 'area_unknown', 'fips', 'num_fireplace', 'num_bath',
'num_garage', 'area_garage', 'heating', 'latitude', 'longitude', 'area_lot',
'num_pool', 'area_pool', 'pooltypeid7', 'zoning_landuse',
'rawcensustractandblock', 'region_city', 'region_county', 'region_neighbor',
'region_zip', 'num_room', 'num_75_bath', 'num_unit', 'area_patio', 'build_year',
'num_story', 'tax_building', 'tax_total', 'tax_year', 'tax_land',
'tax_property', 'tax_delinquency_year', 'censustractandblock']
41
```

on a obtenu le classement de l'importance des caractèristiques des propriétés selon cet ordre avec la méthode xgboost

[90]:
```
"""affichage de l'arbre de décision réalisé par xgboost"""
xgb.to_graphviz(model, num_trees=2,rankdir='LR')
```

[90]:



Dans la partie qui suit on a testé nos différentes formes des données en régréssion (les attributs pertinentes, les composantes principales, les attributs ordinales encodés)

[91]:
```
# entrainement d'un modèle xgboost régressor sur les composantes principales
↪obtenus en fonction de 95% de variance expliquée
best_xgb_model = xgb.XGBRegressor(colsample_bytree=0.4,
                 gamma=0,
                 learning_rate=0.07,
                 max_depth=3,
                 min_child_weight=1.5,
```

```
                 n_estimators=10000,
                 reg_alpha=0.75,
                 reg_lambda=0.45,
                 subsample=0.6,
                 seed=42)
best_xgb_model.fit(train_pca[:1000],train_lbl[:1000])
```

[91]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=0.4, gamma=0, importance_type='gain',
       learning_rate=0.07, max_delta_step=0, max_depth=3,
       min_child_weight=1.5, missing=None, n_estimators=10000, n_jobs=1,
       nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0.75, reg_lambda=0.45, scale_pos_weight=1, seed=42,
       silent=True, subsample=0.6)
```

[92]:
```
y_pred = np.expm1(best_xgb_model.predict(test_pca[:20]))
```

[93]:
```
from sklearn.metrics import explained_variance_score
y_pred = np.expm1(best_xgb_model.predict(train_pca[:1000]))
y_pred = np.around(y_pred, decimals=2)
y_true = np.around(train_lbl[:1000].to_numpy(), decimals = 2)
explained_variance_score(y_pred, y_true)
"""meilleure performance obtenu en faisant la prédiction sur un sous␣
 ↪échantillion"""
```

[93]:
```
'meilleure performance obtenu en faisant la pr\xc3\xa9diction sur un sous
\xc3\xa9chantillion'
```

[94]:
```
best_xgb_model = xgb.XGBRegressor(colsample_bytree=0.4,
                 gamma=0,
                 learning_rate=0.07,
                 max_depth=6,
                 min_child_weight=1.5,
                 n_estimators=10000,
                 reg_alpha=0.75,
                 reg_lambda=0.45,
                 subsample=0.6,
                 seed=42)
best_xgb_model.fit(train_pca[:1000],train_lbl[:1000])

from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import explained_variance_score
y_pred = np.expm1(best_xgb_model.predict(train_pca[:1000]))
y_pred = np.around(y_pred, decimals=3)
y_true = np.around(train_lbl[:1000].to_numpy(), decimals = 3)
explained_variance_score( y_true, y_pred)
```

```
mean_squared_error(y_true, y_pred)
mse = mean_squared_error(y_true, y_pred)
mae = mean_absolute_error(y_true, y_pred)
erreur_general = 1-explained_variance_score(y_true, y_pred)
erreur_general
scores_acp ={'erreur_general': erreur_general,'mse':mse, 'mae':mae }
scores_acp



#print("Erreur de train normalisée (1-explained␣
↪var)",1-explained_variance_score(y_true, y_pred))
```

[94]:
```
{'erreur_general': 2.5868646881251354,
 'mae': 0.022608000078124923,
 'mse': 0.057859991662471855}
```

[132]:
```
X_test = test["x"].values[:,np.newaxis]
y_test = test["y"]


from sklearn.metrics import mean_squared_error
```

[118]:
```
explained_variance_score(y_pred,a)
```

[118]:
```
0.5089846204709485
```

[120]:
```
explained_variance_score(y_pred, a)
```

[120]:
```
0.5089846204709485
```

[138]:
```
print df.head()
```

```
   id_parcel  aircon  architectural_style  area_basement  num_bathroom  \
0   17073783     1.0                  7.0         1528.0           2.5
1   17088994     1.0                  7.0         1528.0           1.0
2   17100444     1.0                  7.0         1528.0           2.0
3   17102429     1.0                  7.0         1528.0           1.5
4   17109604     1.0                  7.0         1528.0           2.5

   num_bedroom  framing  quality  num_bathroom_calc  deck  …  1432  1720  \
0          3.0      4.0      7.0                2.5  66.0  …   0.0   0.0
1          2.0      4.0      7.0                1.0  66.0  …   0.0   0.0
2          3.0      4.0      7.0                2.0  66.0  …   0.0   0.0
```

```
3          2.0      4.0      7.0                 1.5   66.0  …   0.0    0.0
4          4.0      4.0      7.0                 2.5   66.0  …   0.0    0.0

   1722  200   34   38  6050   73  8800   96
0   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0
1   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0
2   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0
3   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0
4   0.0  0.0  0.0  0.0   0.0  0.0   0.0  0.0

[5 rows x 137 columns]
```

[84]:

```python
scaled_features_df.head()
scaled_features_df.keys()
#print scaled_features_df['logerror']
#print merged['logerror']
#scaled_features_df['logerror'] = merged['logerror']
print scaled_features_df.head()

merged.head()
df.head
```

```
     aircon  num_bathroom  num_bedroom   quality  num_bathroom_calc  \
0 -0.151158      0.219590    -0.027558  0.547515           0.200777
1 -0.151158     -1.274040    -0.892289  0.547515          -1.344990
2 -0.151158     -0.278287    -0.027558  0.547515          -0.314479
3 -0.151158     -0.776163    -0.892289  0.547515          -0.829734
4 -0.151158      0.219590     0.837173  0.547515           0.200777

   area_firstfloor_finished  area_total_calc  area_live_finished  \
0                 -1.355111        -0.545319           -0.506773
1                 -0.351894        -1.071213           -1.051361
2                  1.067506        -0.721337           -0.689048
3                  3.052037        -0.232159           -0.182480
4                  1.961202         0.697603            0.780334

   area_total_finished  area_unknown  …  build_year  num_story  \
0            -0.109577     -1.323601  …    0.741925   2.820431
1            -0.109577     -0.345530  …    0.910731  -0.314867
2            -0.109577      1.038292  …   -0.524125  -0.314867
3            -0.109577      2.973080  …   -0.144310  -0.314867
4            -0.109577      1.909587  …    0.657521   2.820431

   tax_building  tax_total  tax_year  tax_land  tax_property  \
0     -0.309789  -0.479133       0.0 -0.503402     -0.580317
1     -0.172200  -0.392865       0.0 -0.455596     -0.497518
2     -0.700050  -0.738573       0.0 -0.659435     -0.788459
3     -0.642614  -0.711431       0.0 -0.651767     -0.775132
```

```
4      0.465833   0.173603       0.0 -0.003327     -0.014149

   tax_delinquency_year  censustractandblock  logerror
0              0.030208             3.032970    0.0953
1              0.030208             3.032936    0.0198
2              0.030208             3.032896    0.0060
3              0.030208             3.032901   -0.0566
4              0.030208             3.032931    0.0573

[5 rows x 42 columns]
```

[84]:
```
<bound method DataFrame.head of        id_parcel  aircon  num_bathroom
num_bedroom  quality  \
0       17073783     1.0           2.5          3.0      7.0
1       17088994     1.0           1.0          2.0      7.0
2       17100444     1.0           2.0          3.0      7.0
3       17102429     1.0           1.5          2.0      7.0
4       17109604     1.0           2.5          4.0      7.0
5       17125829     1.0           2.5          4.0      7.0
6       17132911     1.0           2.0          3.0      7.0
7       17134926     1.0           2.5          5.0      7.0
8       17139988     1.0           2.0          3.0      7.0
9       17167359     1.0           1.0          3.0      7.0
10      17179760     1.0           2.0          4.0      7.0
11      17198685     1.0           2.5          4.0      7.0
12      17212207     1.0           2.0          5.0      7.0
13      17213421     1.0           3.0          3.0      7.0
14      17250387     1.0           2.5          5.0      7.0
15      17254534     1.0           2.0          4.0      7.0
16      17260270     1.0           2.5          5.0      7.0
17      17261131     1.0           1.0          2.0      7.0
18      17275640     1.0           2.5          2.0      7.0
19      17275763     1.0           2.0          2.0      7.0
20      17276736     1.0           1.0          2.0      7.0
21      17283162     1.0           3.0          5.0      7.0
22      17283891     1.0           2.5          4.0      7.0
23      17290104     1.0           2.5          3.0      7.0
24      17291231     1.0           2.0          3.0      7.0
25      17296734     1.0           3.0          5.0      7.0
26      10726315     1.0           3.0          3.0      4.0
27      10727091     1.0           2.0          3.0      7.0
28      10730788     1.0           2.0          4.0      7.0
29      10735394     1.0           3.0          4.0      4.0
30      10737937     1.0           7.0          6.0     10.0
31      10743512     1.0           3.0          2.0      4.0
…           …      …           …          …      …
90243   14457704     1.0           4.5          4.0      7.0
```

90244	11272499	1.0	2.0	3.0	4.0
90245	11348706	1.0	4.0	4.0	1.0
90246	11607868	1.0	3.0	2.0	10.0
90247	10944382	1.0	2.0	1.0	4.0
90248	11793964	1.0	2.0	2.0	1.0
90249	13863024	1.0	3.5	3.0	7.0
90250	14700275	1.0	2.0	3.0	7.0
90251	14700375	1.0	2.0	2.0	7.0
90252	14748051	1.0	2.5	1.0	7.0
90253	17153910	1.0	3.0	4.0	7.0
90254	11485157	1.0	3.0	2.0	1.0
90255	14602791	1.0	1.0	1.0	7.0
90256	13853998	1.0	4.0	5.0	7.0
90257	11780879	1.0	2.0	2.0	4.0
90258	11483546	1.0	2.0	2.0	4.0
90259	11780710	1.0	1.0	1.0	4.0
90260	11125730	1.0	3.0	4.0	4.0
90261	11907577	1.0	3.0	3.0	4.0
90262	12011124	1.0	2.0	0.0	7.0
90263	11812411	1.0	3.0	2.0	4.0
90264	12111197	1.0	1.0	1.0	1.0
90265	11538534	1.0	2.0	2.0	10.0
90266	17095942	1.0	3.0	4.0	7.0
90267	14722093	1.0	2.5	2.0	7.0
90268	14722150	1.0	2.0	2.0	7.0
90269	14600359	1.0	3.0	4.0	7.0
90270	11876798	1.0	3.0	4.0	4.0
90271	12808516	1.0	2.0	3.0	7.0
90272	12010248	1.0	2.0	2.0	4.0
90273	14310905	1.0	2.5	3.0	7.0
90274	14636609	1.0	2.5	3.0	7.0

	num_bathroom_calc	area_firstfloor_finished	area_total_calc	\
0	2.5	548.0	1264.0	
1	1.0	777.0	777.0	
2	2.0	1101.0	1101.0	
3	1.5	1554.0	1554.0	
4	2.5	1305.0	2415.0	
5	2.5	1303.0	2882.0	
6	2.0	1772.0	1772.0	
7	2.5	1240.0	2632.0	
8	2.0	1292.0	1292.0	
9	1.0	804.0	1385.0	
10	2.0	1260.0	1260.0	
11	2.5	1448.0	2735.0	
12	2.0	2085.0	2085.0	
13	3.0	906.0	1508.0	

14	2.5	977.0	1958.0
15	2.0	1120.0	1687.0
16	2.5	1236.0	2232.0
17	1.0	435.0	834.0
18	2.5	691.0	1361.0
19	2.0	917.0	917.0
20	1.0	817.0	907.0
21	3.0	2524.0	2524.0
22	2.5	2400.0	2400.0
23	2.5	1137.0	2113.0
24	2.0	2297.0	2297.0
25	3.0	1996.0	1996.0
26	3.0	817.0	2445.0
27	2.0	817.0	1160.0
28	2.0	817.0	1570.0
29	3.0	817.0	2863.0
30	7.0	817.0	6610.0
31	3.0	817.0	1394.0
…	…	…	…
90243	4.5	817.0	4365.0
90244	2.0	817.0	1565.0
90245	4.0	817.0	3568.0
90246	3.0	817.0	1656.0
90247	2.0	817.0	1450.0
90248	2.0	817.0	1240.0
90249	3.5	817.0	2972.0
90250	2.0	817.0	1456.0
90251	2.0	817.0	1252.0
90252	2.5	817.0	1680.0
90253	3.0	1587.0	3096.0
90254	3.0	817.0	1110.0
90255	1.0	817.0	728.0
90256	4.0	817.0	3308.0
90257	2.0	817.0	1440.0
90258	2.0	817.0	1550.0
90259	1.0	817.0	860.0
90260	3.0	817.0	2781.0
90261	3.0	817.0	1432.0
90262	2.0	817.0	2140.0
90263	3.0	817.0	1060.0
90264	1.0	817.0	918.0
90265	2.0	817.0	1492.0
90266	3.0	440.0	1771.0
90267	2.5	817.0	1638.0
90268	2.0	817.0	1308.0
90269	3.0	817.0	1713.0
90270	3.0	817.0	2068.0

```
90271               2.0                     817.0          1352.0
90272               2.0                     817.0           860.0
90273               2.5                     817.0          2268.0
90274               2.5                     817.0          1812.0

       area_live_finished  area_total_finished  …  1432  1720  1722  200  \
0                  1264.0               1680.0  …   0.0   0.0   0.0  0.0
1                   777.0               1680.0  …   0.0   0.0   0.0  0.0
2                  1101.0               1680.0  …   0.0   0.0   0.0  0.0
3                  1554.0               1680.0  …   0.0   0.0   0.0  0.0
4                  2415.0               1680.0  …   0.0   0.0   0.0  0.0
5                  2882.0               1680.0  …   0.0   0.0   0.0  0.0
6                  1772.0               1680.0  …   0.0   0.0   0.0  0.0
7                  2632.0               1680.0  …   0.0   0.0   0.0  0.0
8                  1292.0               1680.0  …   0.0   0.0   0.0  0.0
9                  1385.0               1680.0  …   0.0   0.0   0.0  0.0
10                 1260.0               1680.0  …   0.0   0.0   0.0  0.0
11                 2735.0               1680.0  …   0.0   0.0   0.0  0.0
12                 2085.0               1680.0  …   0.0   0.0   0.0  0.0
13                 1508.0               1680.0  …   0.0   0.0   0.0  0.0
14                 1958.0               1680.0  …   0.0   0.0   0.0  0.0
15                 1687.0               1680.0  …   0.0   0.0   0.0  0.0
16                 2232.0               1680.0  …   0.0   0.0   0.0  0.0
17                  834.0               1680.0  …   0.0   0.0   0.0  0.0
18                 1361.0               1680.0  …   0.0   0.0   0.0  0.0
19                  917.0               1680.0  …   0.0   0.0   0.0  0.0
20                  907.0               1680.0  …   0.0   0.0   0.0  0.0
21                 2524.0               1680.0  …   0.0   0.0   0.0  0.0
22                 2400.0               1680.0  …   0.0   0.0   0.0  0.0
23                 2113.0               1680.0  …   0.0   0.0   0.0  0.0
24                 2297.0               1680.0  …   0.0   0.0   0.0  0.0
25                 1996.0               1680.0  …   0.0   0.0   0.0  0.0
26                 2445.0               1680.0  …   0.0   0.0   0.0  0.0
27                 1160.0               1680.0  …   0.0   0.0   0.0  0.0
28                 1570.0               1680.0  …   0.0   0.0   0.0  0.0
29                 2863.0               1680.0  …   0.0   0.0   0.0  0.0
30                 6610.0               1680.0  …   0.0   0.0   0.0  0.0
31                 1394.0               1680.0  …   0.0   0.0   0.0  0.0
…                     …                  …  …   …   …   …   …
90243              4365.0               1680.0  …   0.0   0.0   0.0  0.0
90244              1565.0               1680.0  …   0.0   0.0   0.0  0.0
90245              3568.0               1680.0  …   0.0   0.0   0.0  0.0
90246              1656.0               1680.0  …   0.0   0.0   0.0  0.0
90247              1450.0               1680.0  …   0.0   0.0   0.0  0.0
90248              1240.0               1680.0  …   0.0   0.0   0.0  0.0
90249              2972.0               1680.0  …   0.0   0.0   0.0  0.0
90250              1456.0               1680.0  …   0.0   0.0   0.0  0.0
```

90251	1252.0	1680.0	…	0.0	0.0	0.0	0.0
90252	1680.0	1680.0	…	0.0	0.0	0.0	0.0
90253	3096.0	1680.0	…	0.0	0.0	0.0	0.0
90254	1110.0	1680.0	…	0.0	0.0	0.0	0.0
90255	728.0	1680.0	…	0.0	0.0	0.0	0.0
90256	3308.0	1680.0	…	0.0	0.0	0.0	0.0
90257	1440.0	1680.0	…	0.0	0.0	0.0	0.0
90258	1550.0	1680.0	…	0.0	0.0	0.0	0.0
90259	860.0	1680.0	…	0.0	0.0	0.0	0.0
90260	2781.0	1680.0	…	0.0	0.0	0.0	0.0
90261	1432.0	1680.0	…	0.0	0.0	0.0	0.0
90262	2140.0	1680.0	…	0.0	0.0	0.0	0.0
90263	1060.0	1680.0	…	0.0	0.0	0.0	0.0
90264	918.0	1680.0	…	0.0	0.0	0.0	0.0
90265	1492.0	1680.0	…	0.0	0.0	0.0	0.0
90266	1771.0	1680.0	…	0.0	0.0	0.0	0.0
90267	1638.0	1680.0	…	0.0	0.0	0.0	0.0
90268	1308.0	1680.0	…	0.0	0.0	0.0	0.0
90269	1713.0	1680.0	…	0.0	0.0	0.0	0.0
90270	2068.0	1680.0	…	0.0	0.0	0.0	0.0
90271	1352.0	1680.0	…	0.0	0.0	0.0	0.0
90272	860.0	1680.0	…	0.0	0.0	0.0	0.0
90273	2268.0	1680.0	…	0.0	0.0	0.0	0.0
90274	1812.0	1680.0	…	0.0	0.0	0.0	0.0

	34	38	6050	73	8800	96
0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0
15	0.0	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0

```
21     0.0  0.0   0.0  0.0   0.0  0.0
22     0.0  0.0   0.0  0.0   0.0  0.0
23     0.0  0.0   0.0  0.0   0.0  0.0
24     0.0  0.0   0.0  0.0   0.0  0.0
25     0.0  0.0   0.0  0.0   0.0  0.0
26     0.0  0.0   0.0  0.0   0.0  0.0
27     0.0  0.0   0.0  0.0   0.0  0.0
28     0.0  0.0   0.0  0.0   0.0  0.0
29     0.0  0.0   0.0  0.0   0.0  0.0
30     0.0  0.0   0.0  0.0   0.0  0.0
31     0.0  0.0   0.0  0.0   0.0  0.0
…    …  …    …  …    …  …
90243  0.0  0.0   0.0  0.0   0.0  0.0
90244  0.0  0.0   0.0  0.0   0.0  0.0
90245  0.0  0.0   0.0  0.0   0.0  0.0
90246  0.0  0.0   0.0  0.0   0.0  0.0
90247  0.0  0.0   0.0  0.0   0.0  0.0
90248  0.0  0.0   0.0  0.0   0.0  0.0
90249  0.0  0.0   0.0  0.0   0.0  0.0
90250  0.0  0.0   0.0  0.0   0.0  0.0
90251  0.0  0.0   0.0  0.0   0.0  0.0
90252  0.0  0.0   0.0  0.0   0.0  0.0
90253  0.0  0.0   0.0  0.0   0.0  0.0
90254  0.0  0.0   0.0  0.0   0.0  0.0
90255  0.0  0.0   0.0  0.0   0.0  0.0
90256  0.0  0.0   0.0  0.0   0.0  0.0
90257  0.0  0.0   0.0  0.0   0.0  0.0
90258  0.0  0.0   0.0  0.0   0.0  0.0
90259  0.0  0.0   0.0  0.0   0.0  0.0
90260  0.0  0.0   0.0  0.0   0.0  0.0
90261  0.0  0.0   0.0  0.0   0.0  0.0
90262  0.0  0.0   0.0  0.0   0.0  0.0
90263  0.0  0.0   0.0  0.0   0.0  0.0
90264  0.0  0.0   0.0  0.0   0.0  0.0
90265  0.0  0.0   0.0  0.0   0.0  0.0
90266  0.0  0.0   0.0  0.0   0.0  0.0
90267  0.0  0.0   0.0  0.0   0.0  0.0
90268  0.0  0.0   0.0  0.0   0.0  0.0
90269  0.0  0.0   0.0  0.0   0.0  0.0
90270  0.0  0.0   0.0  0.0   0.0  0.0
90271  0.0  0.0   0.0  0.0   0.0  0.0
90272  0.0  0.0   0.0  0.0   0.0  0.0
90273  0.0  0.0   0.0  0.0   0.0  0.0
90274  0.0  0.0   0.0  0.0   0.0  0.0

[90275 rows x 125 columns]>
```

```
[86]: #print  x_cols
liste_modeles_scores = []
```

```
['aircon', 'num_bathroom', 'num_bedroom', 'quality', 'num_bathroom_calc',
'area_firstfloor_finished', 'area_total_calc', 'area_live_finished',
'area_total_finished', 'area_unknown', 'fips', 'num_fireplace', 'num_bath',
'num_garage', 'area_garage', 'heating', 'latitude', 'longitude', 'area_lot',
'num_pool', 'area_pool', 'pooltypeid7', 'zoning_landuse',
'rawcensustractandblock', 'region_city', 'region_county', 'region_neighbor',
'region_zip', 'num_room', 'num_75_bath', 'num_unit', 'area_patio', 'build_year',
'num_story', 'tax_building', 'tax_total', 'tax_year', 'tax_land',
'tax_property', 'tax_delinquency_year', 'censustractandblock']
```

```
[153]: """Régréssion sur des données standardisés numériques seulement"""
scores_stdrnum ={}
# test_size: what proportion of original data is used for test set
xtrain, xtest, ytrain, ytest = train_test_split(scaled_features_df[x_cols],
                                                                            ␣
 ↪merged['logerror'],
                                                                             test_size=1/7.0,
                                                                             random_state=0)







from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import explained_variance_score
#X = train_pca # Matrice colonne plutôt que vecteur ligne
#y = train_lbl

from sklearn.linear_model import LinearRegression
reg_lin = LinearRegression(fit_intercept=True)

reg_lin.fit(xtrain, ytrain)


y_pred = reg_lin.predict(xtrain)

y_pred = np.around(y_pred, decimals=3)

y_true = np.around(ytrain.to_numpy(), decimals = 3).tolist()
#print ytest
mse = mean_absolute_error( y_true, y_pred)
print y_pred[:10]
```

```
print y_true[:10]
mse = mean_squared_error(y_true, y_pred)
mae = mean_absolute_error(y_true, y_pred)
erreur_general = 1-explained_variance_score(y_true, y_pred)
erreur_general
scores_stdrnum ={'erreur_general': erreur_general,'mse':mse, 'mae':mae }
scores_stdrnum
```

```
[ 0.     0.001  0.013  0.009 -0.003  0.025  0.001  0.005  0.017  0.019]
[-0.02, 0.0, -0.008, -0.081, -0.032, -0.026, -0.011, -0.041, -0.098, 0.247]
```

[153]:
```
{'erreur_general': 0.9934614190313762,
 'mae': 0.06873285688438573,
 'mse': 0.02596930681847553}
```

[154]:
```
"""Régréssion sur des données standardisés numériques seulement"""
scores_stdrnum ={}

# test_size: what proportion of original data is used for test set
xtrain, xtest, ytrain, ytest = train_test_split(merged[x_cols],
                                                                      ␣
 ↪merged['logerror'],
                                                                        test_size=1/7.0,
                                                                        random_state=0)






from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import explained_variance_score
#X = train_pca # Matrice colonne plutôt que vecteur ligne
#y = train_lbl

from sklearn.linear_model import LinearRegression
reg_lin = LinearRegression(fit_intercept=True)

reg_lin.fit(xtrain, ytrain)


y_pred = reg_lin.predict(xtrain)

y_pred = np.around(y_pred, decimals=3)

y_true = np.around(ytrain.to_numpy(), decimals = 3).tolist()
```

```
#print ytest
mse = mean_absolute_error( y_true, y_pred)
print y_pred[:10]
print y_true[:10]
mse = mean_squared_error(y_true, y_pred)
mae = mean_absolute_error(y_true, y_pred)
erreur_general = 1-explained_variance_score(y_true, y_pred)
erreur_general
scores_stdrnum ={'erreur_general': erreur_general,'mse':mse, 'mae':mae }
scores_stdrnum
```

```
[ 0.      0.001  0.013  0.009 -0.003  0.025  0.001  0.005  0.017  0.019]
[-0.02, 0.0, -0.008, -0.081, -0.032, -0.026, -0.011, -0.041, -0.098, 0.247]
```

[154]:

```
{'erreur_general': 0.9934606968321642,
 'mae': 0.06873281811367572,
 'mse': 0.025969287937139755}
```

[159]:

```
"""Régréssion sur les ACP """
scores_stdrnum ={}
from sklearn.model_selection import train_test_split
# test_size: what proportion of original data is used for test set
xtrain, xtest, ytrain, ytest = train_test_split(scaled_features_df[x_cols],
                                                                         ␣
 ↪merged['logerror'],
                                                                          test_size=1/7.0,
                                                                          random_state=0)






from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import explained_variance_score

from sklearn.linear_model import LinearRegression
reg_lin = LinearRegression(fit_intercept=True)

reg_lin.fit(train_pca, train_lbl)


y_pred = reg_lin.predict(train_pca)

y_pred = np.around(y_pred, decimals=3)
```

```
y_true = np.around(train_lbl.to_numpy(), decimals = 3).tolist()
#print ytest
mse = mean_absolute_error( y_true, y_pred)
print y_pred[:10]
print y_true[:10]
mse = mean_squared_error(y_true, y_pred)
mae = mean_absolute_error(y_true, y_pred)
erreur_general = 1-explained_variance_score(y_true, y_pred)
erreur_general
scores_stdrnum ={'erreur_general': erreur_general,'mse':mse, 'mae':mae }
scores_stdrnum
```

```
[0.002 0.013 0.01  0.005 0.005 0.024 0.009 0.009 0.011 0.015]
[-0.02, 0.0, -0.008, -0.081, -0.032, -0.026, -0.011, -0.041, -0.098, 0.247]
```

[159]:
```
{'erreur_general': 0.996866216973792,
 'mae': 0.06883642637442167,
 'mse': 0.026058309170565272}
```

[166]:
```
"""Régréssion sur les caractéristiques importantes (jugées par xgboost)"""
important_features=['area_total_calc','latitude', 'tax_building',␣
 ↪'tax_property', 'num_bathroom']
scores_stdrnum ={}
from sklearn.model_selection import train_test_split
# test_size: what proportion of original data is used for test set
xtrain, xtest, ytrain, ytest = train_test_split(merged[important_features],
                                                                          ␣
 ↪merged['logerror'],
                                                                            test_size=1/7.0,
                                                                            random_state=0)






from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import explained_variance_score


from sklearn.linear_model import LinearRegression
reg_lin = LinearRegression(fit_intercept=True)

reg_lin.fit(xtrain, ytrain)
```

```python
y_pred = reg_lin.predict(xtrain)

y_pred = np.around(y_pred, decimals=3)

y_true = np.around(ytrain.to_numpy(), decimals = 3).tolist()
#print ytest
mse = mean_absolute_error( y_true, y_pred)
print y_pred[:10]
print y_true[:10]
mse = mean_squared_error(y_true, y_pred)
mae = mean_absolute_error(y_true, y_pred)
erreur_general = 1-explained_variance_score(y_true, y_pred)
erreur_general
scores_stdrnum ={'erreur_general': erreur_general,'mse':mse, 'mae':mae }
scores_stdrnum
```

```
[0.002 0.018 0.008 0.011 0.001 0.023 0.005 0.009 0.015 0.01 ]
[-0.02, 0.0, -0.008, -0.081, -0.032, -0.026, -0.011, -0.041, -0.098, 0.247]
```

[166]:
```
{'erreur_general': 0.9961742288855999,
 'mae': 0.06866617126314974,
 'mse': 0.026040220489027887}
```

Malgrès nos études divérses en essayant des méthodes afin d'amliorer la qualité de nos données afin d'améliorer la prédiction, néanmoins la difficulté de ce dataset au vu de sa taille et la variété des natures et des échélles de valeurs, les performances de prédiction en généralisation ont été compliqués, la méilleur performance a été un score de 0.5 en variance explained score sur un sous échaintillion, nous avons essayé de nous inspirer des notebooks réalisés sur ce projet, mais on a trouvé aucun qui a étuidé des performances, on c'est donc focalisé sur l'étude des données et on a apris de nouveaux algorithmes et différentes étapes du pré-processing

[ ]: