In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [2]:
```
    #librairies de data viz
import matplotlib.pyplot as plt
import seaborn as sns
# librairies des modèle linéaire
from sklearn import linear_model
#librairie des modèle d'ensemble
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
# librairies des métriques
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error, r2_score
# librairies pour le k-fold cross validation
from sklearn.model_selection import cross_val_score

from sklearn.model_selection import train_test_split
```

In [4]:
```
df = pd.read_csv("C:/Users/99210/Downloads/dataset.csv")
```

In [5]:
```
df.head()
```

Out[5]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built
0	02-05-2014 00:00	313000.0	3	1.50	1340	7912	1.5	0	0	3	1340	0	1955
1	02-05-2014 00:00	2384000.0	5	2.50	3650	9050	2.0	0	4	5	3370	280	1921
2	02-05-2014 00:00	342000.0	3	2.00	1930	11947	1.0	0	0	4	1930	0	1966
3	02-05-2014 00:00	420000.0	3	2.25	2000	8030	1.0	0	0	4	1000	1000	1963
4	02-05-2014 00:00	550000.0	4	2.50	1940	10500	1.0	0	0	4	1140	800	1976

In [6]:
```
print('Nbr de lignes et nbr de colonnes : ',df.shape)
print(df.dtypes)
```

```
Nbr de lignes et nbr de colonnes :  (4600, 18)
date              object
price            float64
bedrooms           int64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
street            object
city              object
statezip          object
country           object
dtype: object
```

In [7]:
```
df[['floors', 'bathrooms', 'bedrooms','price']] = df[['floors', 'bathrooms', 'bedrooms','price']].astype('int
df.dtypes
```

Out[7]:
```
date             object
price             int32
bedrooms          int32
bathrooms         int32
sqft_living       int64
sqft_lot          int64
floors            int32
waterfront        int64
view              int64
condition         int64
sqft_above        int64
sqft_basement     int64
yr_built          int64
yr_renovated      int64
street           object
city             object
statezip         object
country          object
dtype: object
```

In [8]:
```
df.bathrooms.dtype
```

Out[8]:
```
dtype('int32')
```

In [9]:
```
df.isnull().sum()
```

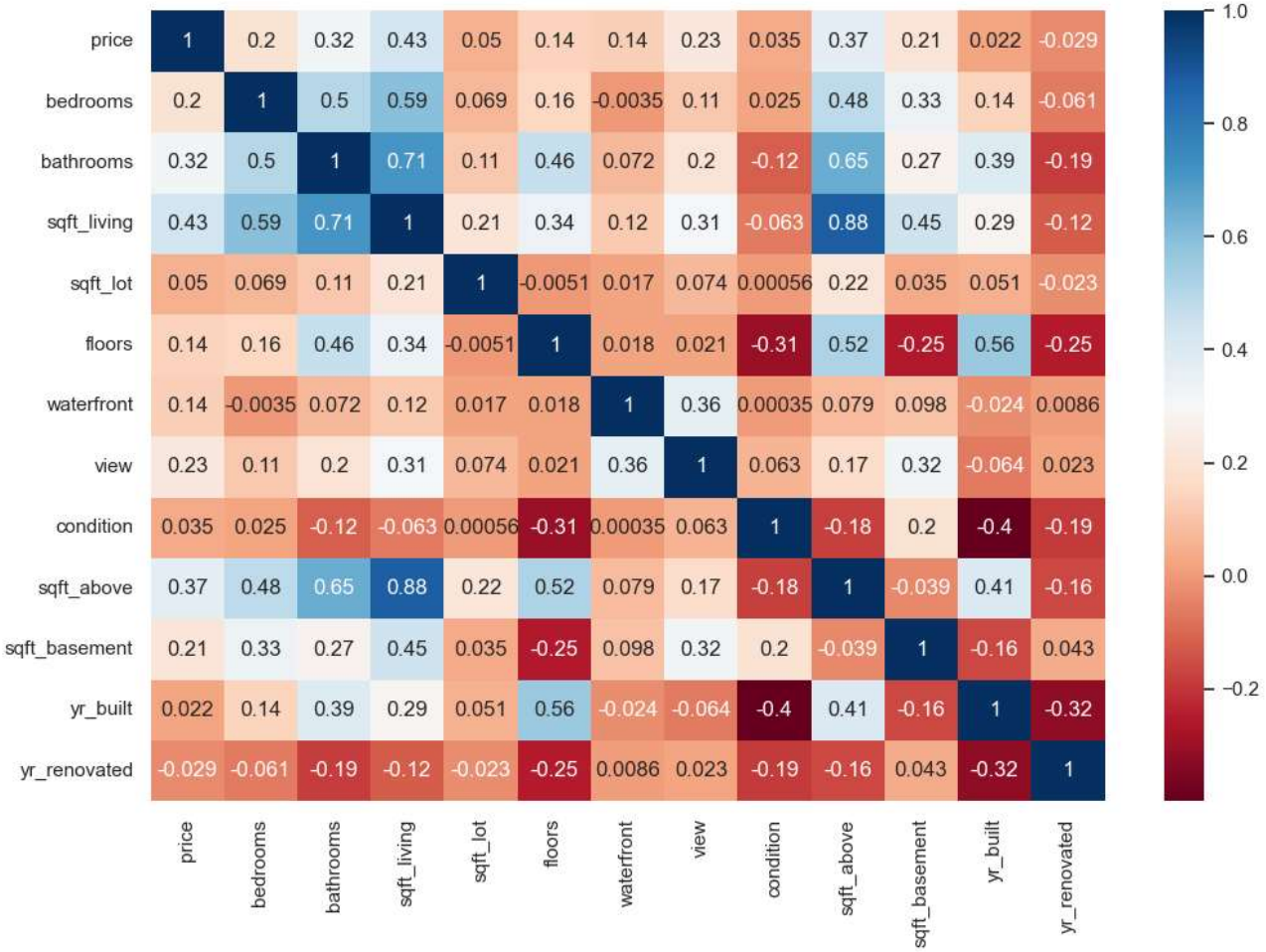
Out[9]:
```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [10]:
```
df.describe()
```

Out[10]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.000000	4600.000000	4600.000000	4600.(
mean	5.519630e+05	3.400870	1.788913	2139.346957	1.485252e+04	1.459130	0.007174	0.240652	3.451739	1827.2
std	5.638347e+05	0.908848	0.752185	963.206916	3.588444e+04	0.552194	0.084404	0.778405	0.677230	862.1
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.000000	0.000000	1.000000	370.(
25%	3.228750e+05	3.000000	1.000000	1460.000000	5.000750e+03	1.000000	0.000000	0.000000	3.000000	1190.(
50%	4.609430e+05	3.000000	2.000000	1980.000000	7.683000e+03	1.000000	0.000000	0.000000	3.000000	1590.(
75%	6.549625e+05	4.000000	2.000000	2620.000000	1.100125e+04	2.000000	0.000000	0.000000	4.000000	2300.(
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.000000	1.000000	4.000000	5.000000	9410.(

In [11]:
```
sns.set(rc={'figure.figsize':(12,8)})
sns.heatmap(df.corr(), annot=True,cmap='RdBu')
plt.show()
```



In [13]:
```
df['price'].replace(0,np.nan, inplace = True)
df.fillna(df['price'].mean(),inplace=True)
```

In [14]:
```
df.drop(['sqft_above','sqft_lot','yr_built','yr_renovated','condition'], axis=1,inplace=True)
```

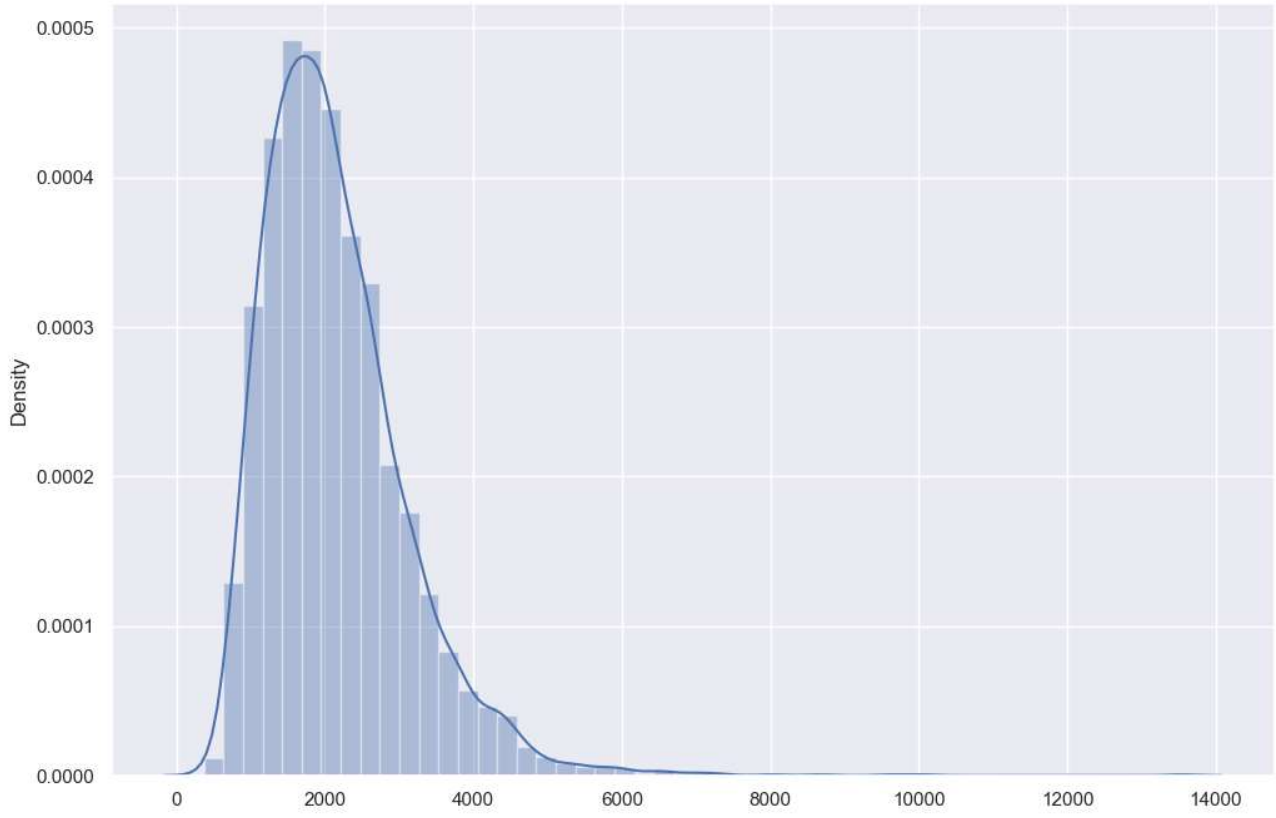
In [15]:
```
df.columns
```

Out[15]:
```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'floors',
       'waterfront', 'view', 'sqft_basement', 'street', 'city', 'statezip',
       'country'],
      dtype='object')
```

In [16]:
```
sns.distplot(x = df.sqft_living)
```

```
C:\Users\99210\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

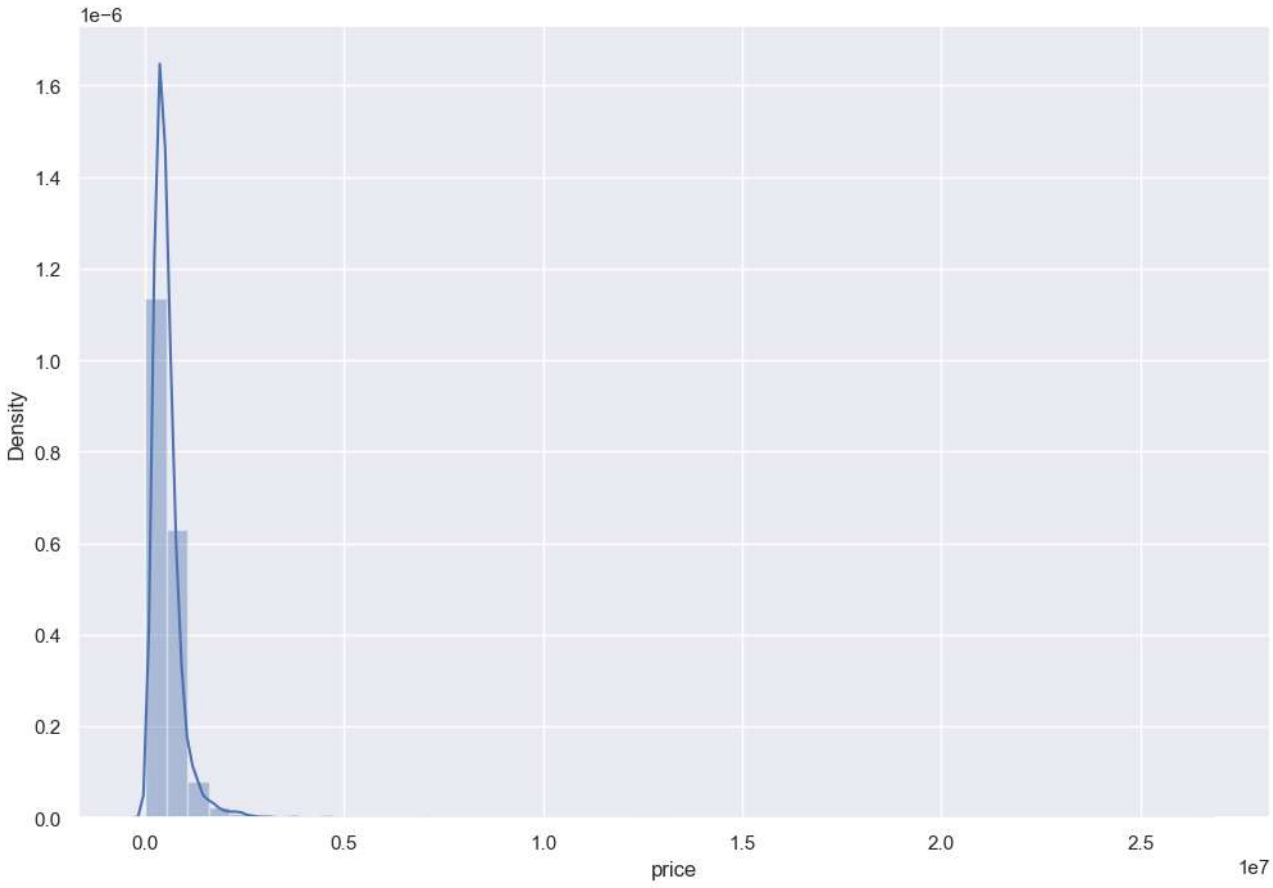
Out[16]: <AxesSubplot:ylabel='Density'>



In [17]:
```
sns.distplot(df.price)
```

```
C:\Users\99210\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[17]: <AxesSubplot:xlabel='price', ylabel='Density'>

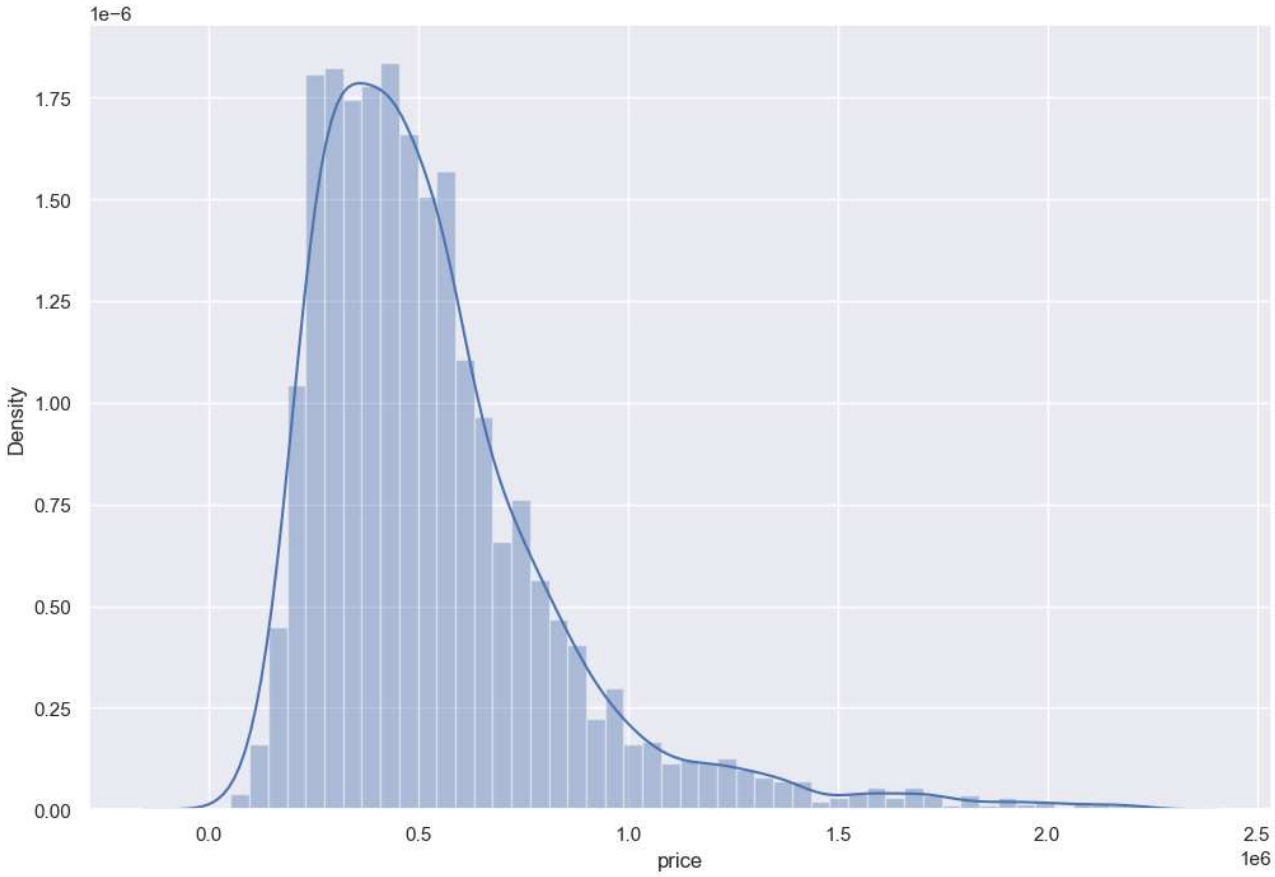


In [18]:
```
from scipy import stats
df['price'] = df['price'].replace([df['price'][np.abs(stats.zscore(df['price'])) > 3]],np.median(df['price']))
```

In [19]:
```
sns.distplot(df.price)
```

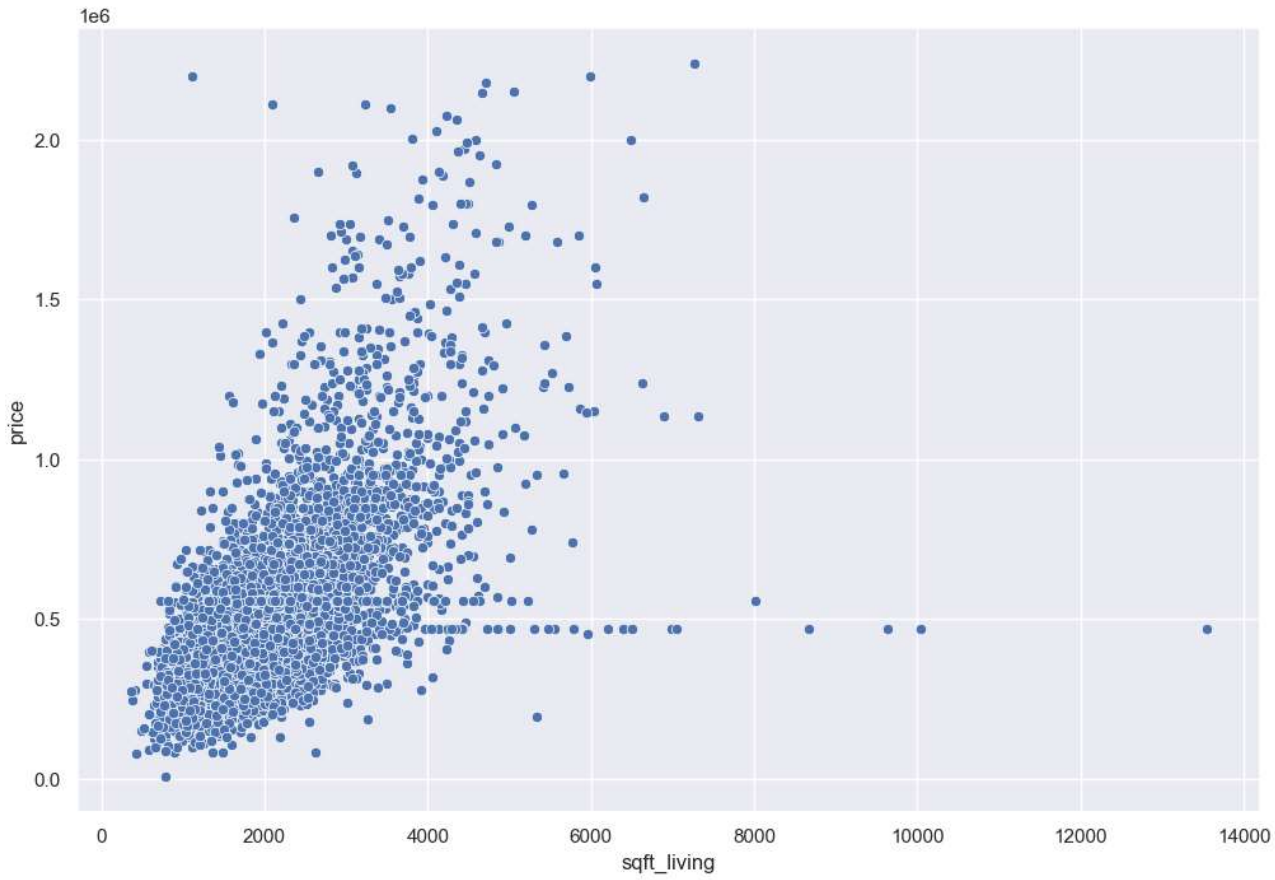
```
C:\Users\99210\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[19]:
```
<AxesSubplot:xlabel='price', ylabel='Density'>
```



In [20]:
```
sns.scatterplot(data = df, x='sqft_living',y='price')
```

Out[20]: <AxesSubplot:xlabel='sqft_living', ylabel='price'>



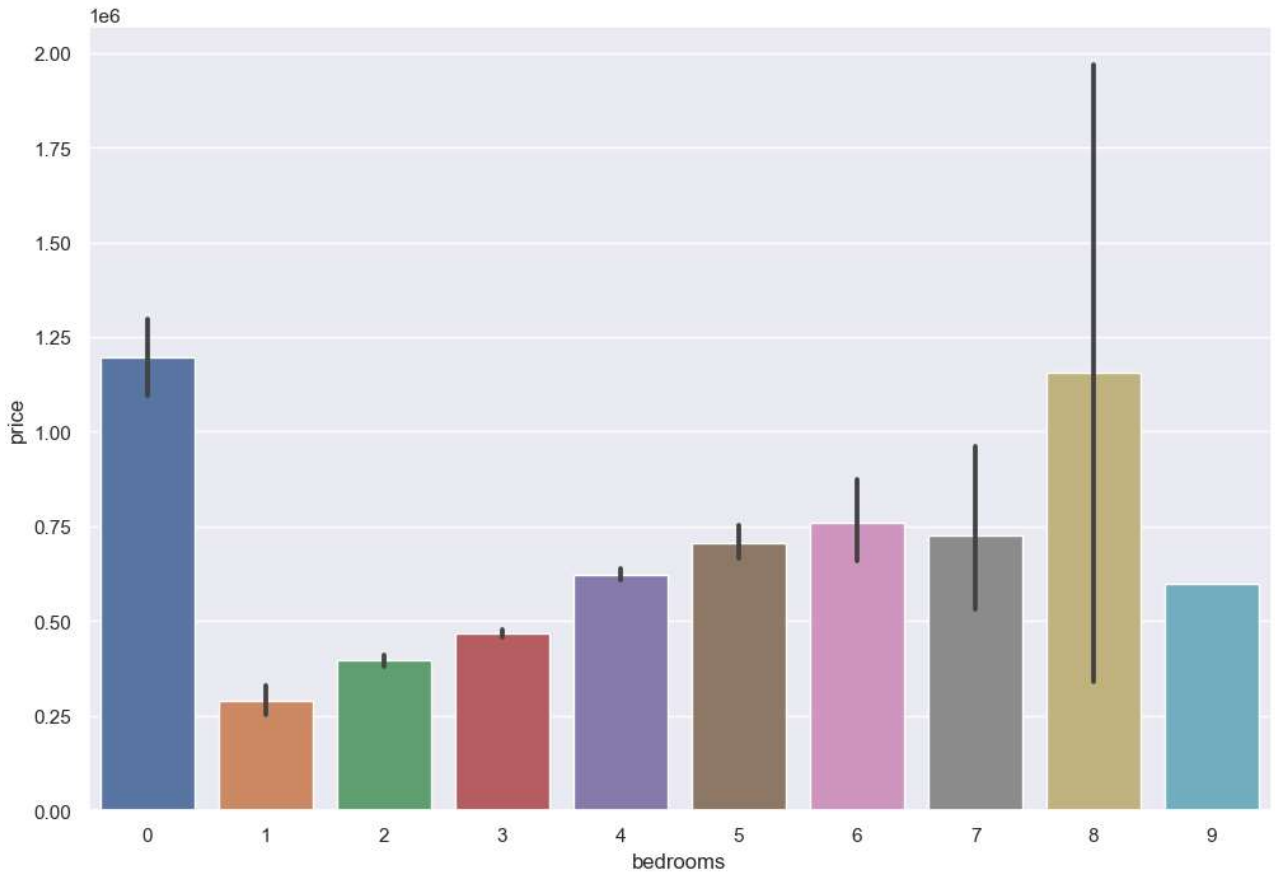
In [21]:
```
df.sqft_living.describe()
```

Out[21]:
```
count     4600.000000
mean      2139.346957
std        963.206916
min        370.000000
25%       1460.000000
50%       1980.000000
75%       2620.000000
max      13540.000000
Name: sqft_living, dtype: float64
```

In [22]:
```
df['sqft_living'] = np.where((df.sqft_living >6000 ), 6000, df.sqft_living)
```

In [23]:
```
sns.barplot(x=df.bedrooms , y = df.price)
```

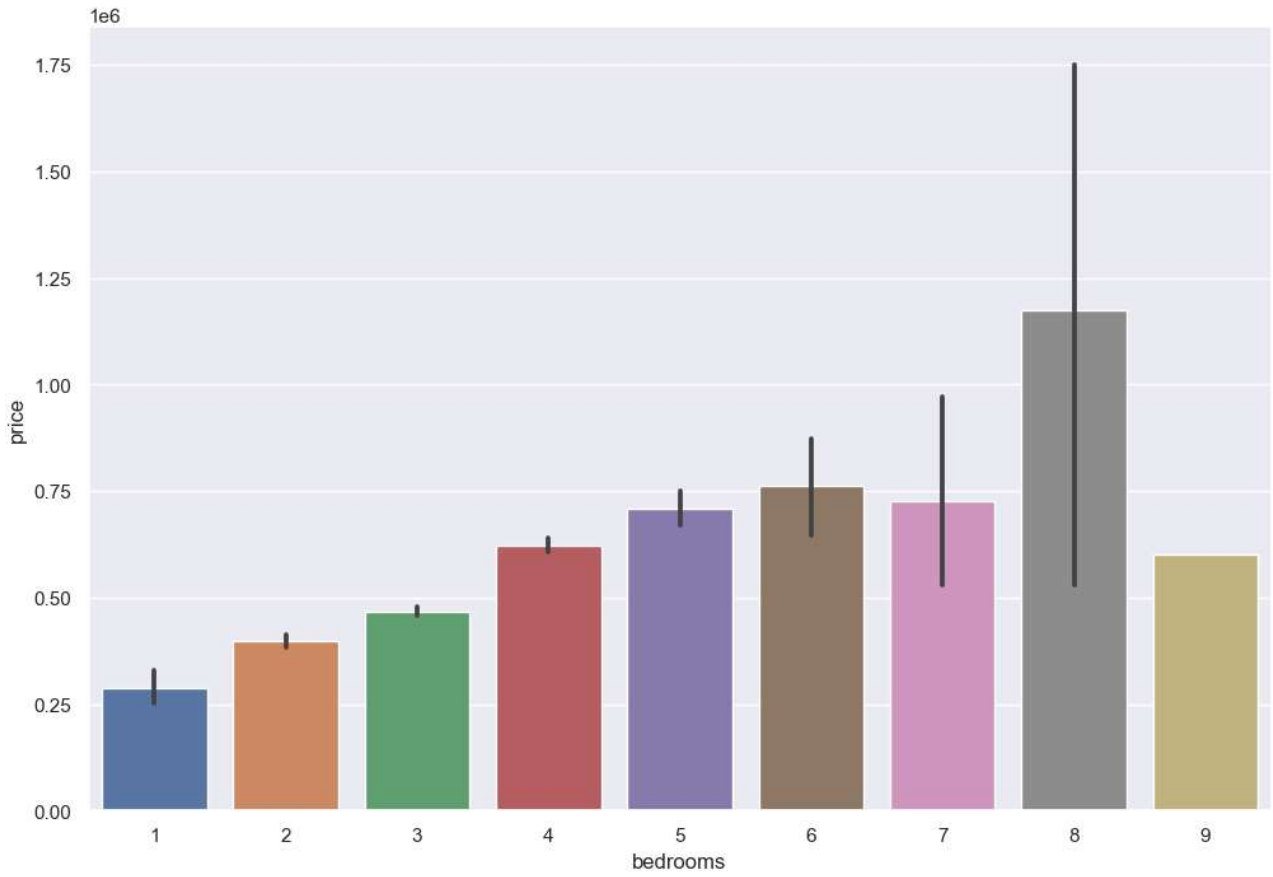
Out[23]: <AxesSubplot:xlabel='bedrooms', ylabel='price'>



In [24]:
```
df.bedrooms.replace(0, 8,inplace = True)
```

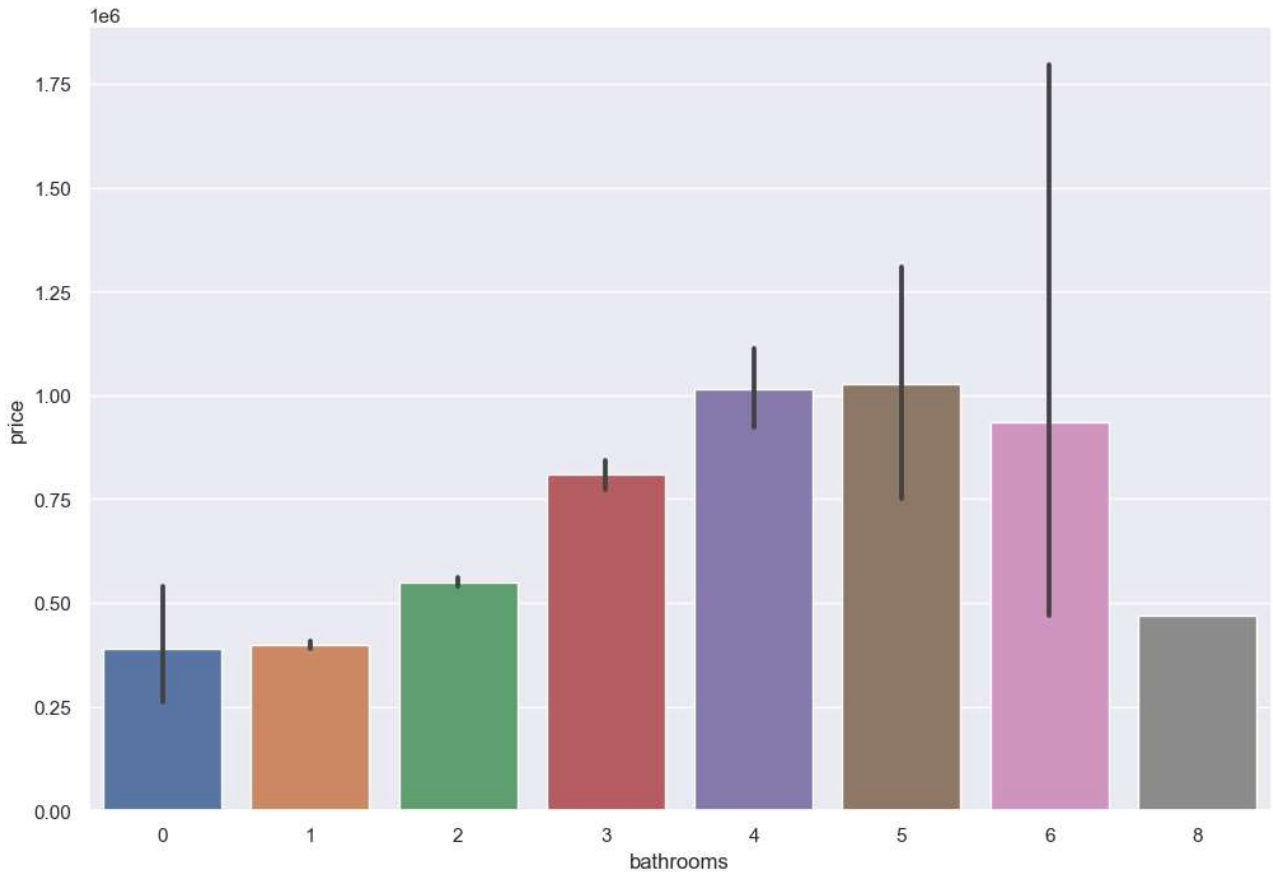
In [25]:
```
sns.barplot(x=df.bedrooms , y = df.price)
```

Out[25]: <AxesSubplot:xlabel='bedrooms', ylabel='price'>



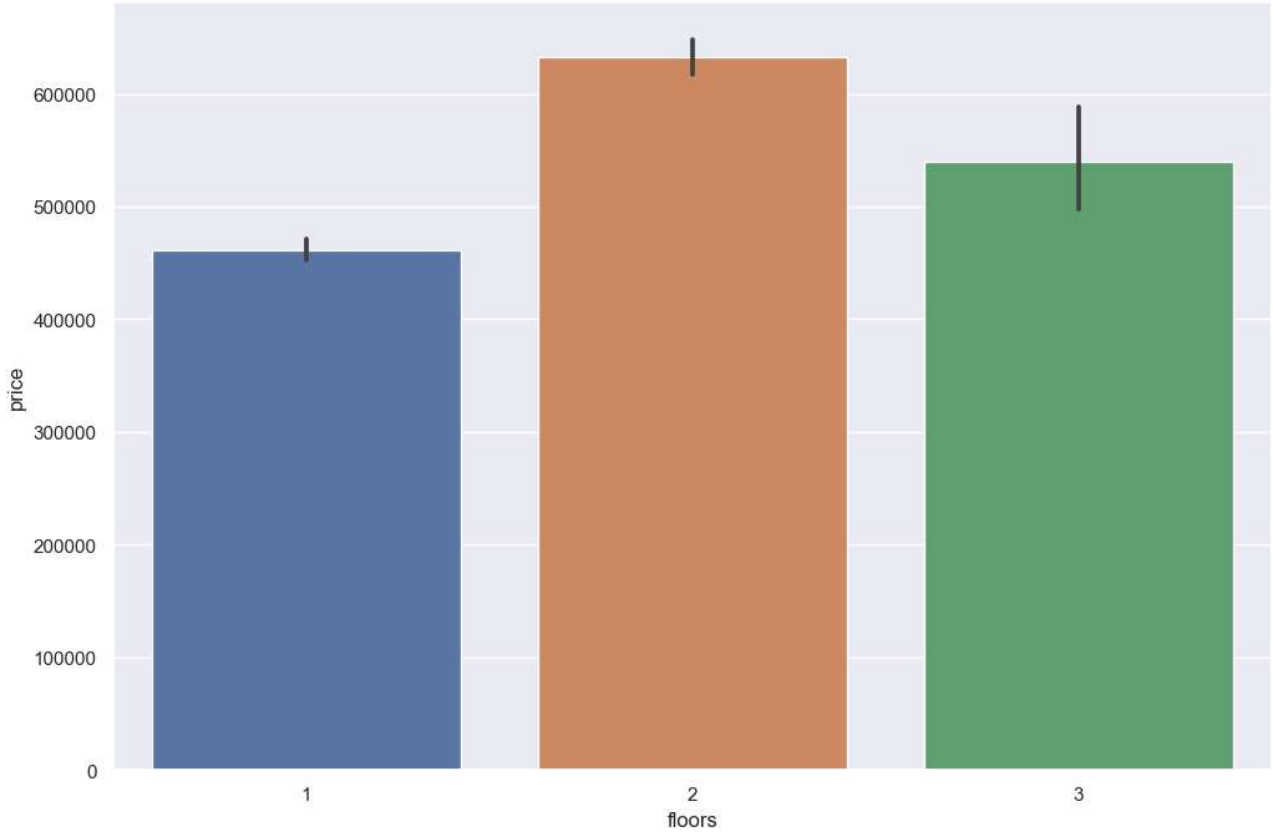
In [27]:
```
sns.barplot(x=df.bathrooms , y = df.price)
```

Out[27]: <AxesSubplot:xlabel='bathrooms', ylabel='price'>



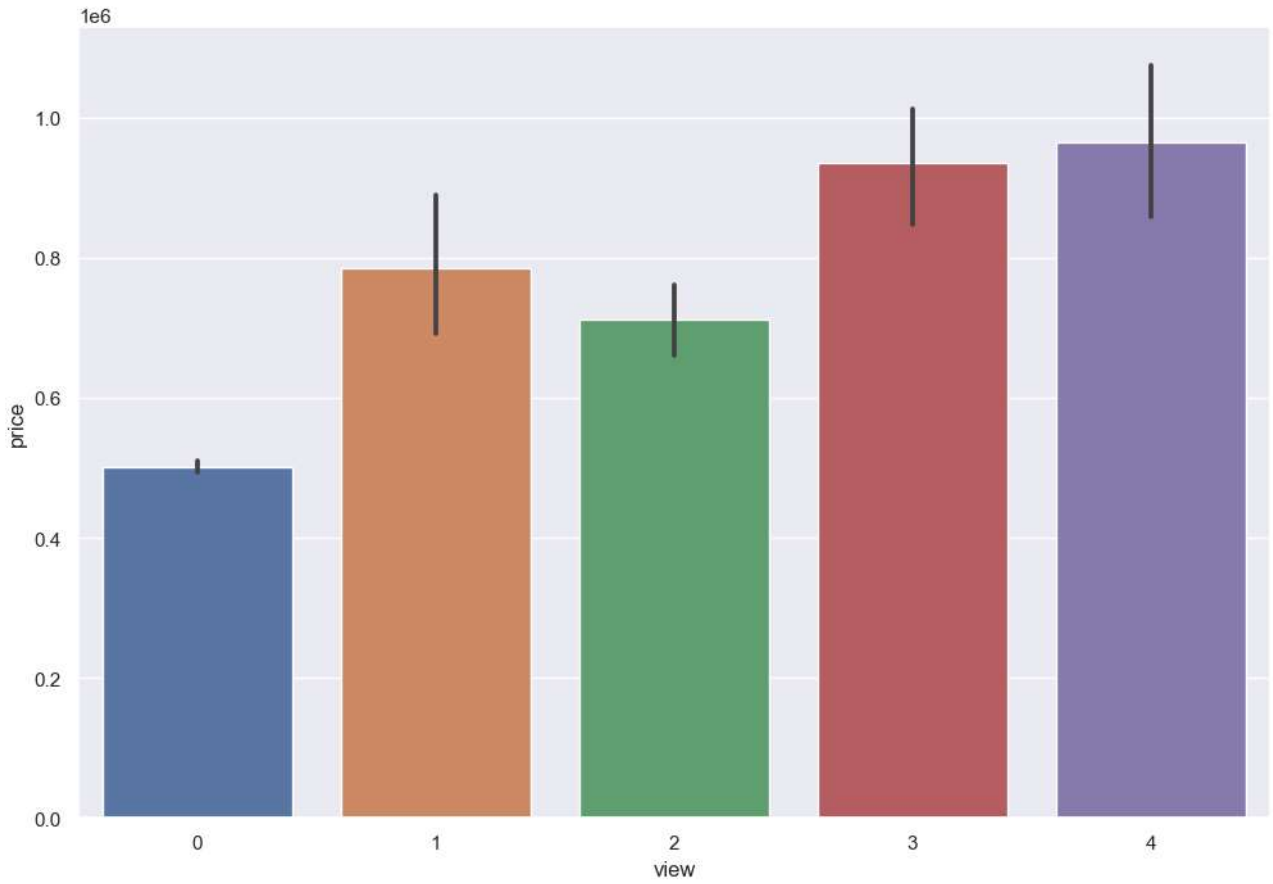
In [28]:
```
sns.barplot(x=df.floors , y = df.price)
```

Out[28]: <AxesSubplot:xlabel='floors', ylabel='price'>



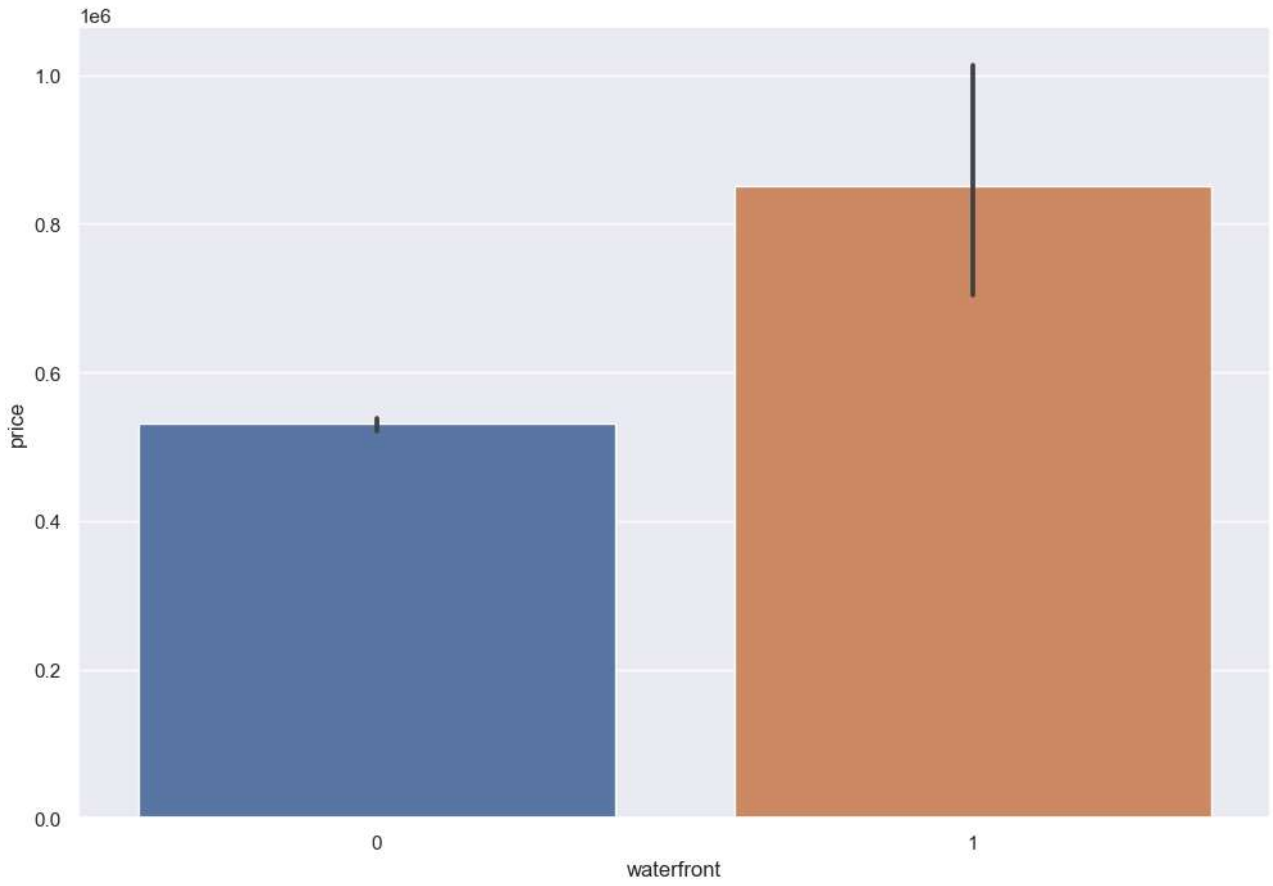
In [29]:
```
sns.barplot(x=df.view , y = df.price)
```

Out[29]: <AxesSubplot:xlabel='view', ylabel='price'>



In [30]:
```
sns.barplot(x=df.waterfront , y = df.price)
```

Out[30]: <AxesSubplot:xlabel='waterfront', ylabel='price'>



In [31]:
```
sns.scatterplot(x='sqft_basement', y ='price',data= df)
```

Out[31]: <AxesSubplot:xlabel='sqft_basement', ylabel='price'>

In [32]:
```
df['sqft_basement'] = np.where((df.sqft_basement >2000 ), 2000, df.sqft_basement)
```

In [33]:
```
df.head()
```

Out[33]:

	date	price	bedrooms	bathrooms	sqft_living	floors	waterfront	view	sqft_basement	street	city	statezip	country
0	02-05-2014 00:00	313000.0	3	1	1340	1	0	0	0	18810 Densmore Ave N	Shoreline	WA 98133	USA
1	02-05-2014 00:00	468750.0	5	2	3650	2	0	4	280	709 W Blaine St	Seattle	WA 98119	USA
2	02-05-2014 00:00	342000.0	3	2	1930	1	0	0	0	26206-26214 143rd Ave SE	Kent	WA 98042	USA
3	02-05-2014 00:00	420000.0	3	2	2000	1	0	0	1000	857 170th Pl NE	Bellevue	WA 98008	USA
4	02-05-2014 00:00	550000.0	4	2	1940	1	0	0	800	9105 170th Ave NE	Redmond	WA 98052	USA

In [34]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 13 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   date           4600 non-null   object 
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   int32  
 3   bathrooms      4600 non-null   int32  
 4   sqft_living    4600 non-null   int64  
 5   floors         4600 non-null   int32  
 6   waterfront     4600 non-null   int64  
 7   view           4600 non-null   int64  
 8   sqft_basement  4600 non-null   int64  
 9   street         4600 non-null   object 
 10  city           4600 non-null   object 
 11  statezip       4600 non-null   object 
 12  country        4600 non-null   object 
dtypes: float64(1), int32(3), int64(4), object(5)
memory usage: 413.4+ KB
```

In [35]:
```
df = pd.get_dummies(df, columns=['city'], prefix = ['city'])
X = df.drop(["date",'street', 'statezip', 'country','price'], axis=1)
y = df[['price']]
```

In [36]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [37]:
```
lr = linear_model.LinearRegression()
lr.fit(X_train, y_train)  #phase d'apprentissage
print(lr.score(X_train, y_train))
print(lr.score(X_test,y_test))
```

```
0.6146718898676132
0.5659003045853999
```

In [38]:
```
l = linear_model.Lasso()
l.fit(X_train, y_train)  #phase d'apprentissage
predl = l.predict(X_test)
scorel=r2_score(y_test,predl)
scorel
```

```
C:\Users\99210\anaconda3\lib\site-packages\sklearn\linear_model\_coordinate_descent.py:647: ConvergenceWarni
ng: Objective did not converge. You might want to increase the number of iterations, check the scale of the
features or consider increasing regularisation. Duality gap: 2.025e+13, tolerance: 2.847e+10
  model = cd_fast.enet_coordinate_descent(
```

Out[38]:
```
0.5661901911100673
```

In [39]:
```
r = linear_model.Ridge()
r.fit(X_train, y_train)  #phase d'apprentissage
predr = r.predict(X_test)
scorer=r2_score(y_test,predr)
scorer
```

Out[39]:
```
0.5638948252673996
```

In [40]:
```
lr = linear_model.LinearRegression()

scores = cross_val_score(lr, X, y, cv=5)

print("score",scores)

print("moyenne :%0.03f , deviation: :%0.03f" % (scores.mean(), scores.std()))
```

```
score [0.64991269 0.61701447 0.63198355 0.65896884 0.3101221 ]
moyenne :0.574 , deviation: :0.133
```

In [41]:
```
from sklearn.ensemble import RandomForestRegressor

# Define the model. Set random_state to 1
rf_model = RandomForestRegressor()

# fit your model
rf_model.fit(X_train, y_train)
rf_model.score(X_train, y_train)
```

```
C:\Users\99210\AppData\Local\Temp\ipykernel_30764\2626062114.py:7: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using rav
el().
  rf_model.fit(X_train, y_train)
```

Out[41]: 0.9351156289941331

In [42]:
```
rf = RandomForestRegressor()

scores = cross_val_score(rf, X, y, cv=5)

print("score",scores)

print("moyenne :%0.03f , deviation: :%0.03f" % (scores.mean(), scores.std()))
```

```
C:\Users\99210\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:680: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\99210\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:680: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\99210\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:680: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\99210\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:680: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)
C:\Users\99210\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:680: DataConversionWarnin
g: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,),
for example using ravel().
  estimator.fit(X_train, y_train, **fit_params)

score [0.61341309 0.61893321 0.61335355 0.58788427 0.34777901]
moyenne :0.556 , deviation: :0.105
```

In [43]:
```
rf_model = RandomForestRegressor(n_estimators=5, max_depth=5)

# fit your model
rf_model.fit(X_train, y_train)
print(rf_model.score(X_train, y_train))
print(rf_model.score(X_test, y_test))
```

```
0.6045465290011636
0.4986676277428299

C:\Users\99210\AppData\Local\Temp\ipykernel_30764\1519264609.py:4: DataConversionWarning: A column-vector y
was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using rav
el().
  rf_model.fit(X_train, y_train)
```

In [*]:

In [ ]: