PCA

May 23, 2025

1 The regression experiment

1.1 is the price correlated to distance to nearest station

1.2 for each meter away from subway station how does the price vary

we start by importing the usual suspects

[2]:
```
# import data science libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# import advanced plot drawing library
import seaborn as sns
# import data science library with added geographic functions
import geopandas as gpd
# useful to calculate the Ordinary Least Square linear regression
import statsmodels
# useful to get fast distance calculation,

from sklearn.neighbors import BallTree, KDTree
# useful to access functions like median, mean or std
from scipy import stats
# useful to get map background but hard to install...
# If not available REMOVE this
import contextily as cx
sns.set_style("white")
```

[ ]:
```

```

1.3 1 – Real Estate dataset: Anjuke

This dataset has been created in UTSEUS in 2017. The souce page for it is (anjuke)[https://shanghai.anjuke.com/sale]

This dataset is given to you as a (pickle)[https://docs.python.org/3/library/pickle.html] which can be used directly as a variable. The format is called pickle, it is like a zip but can only be read in python.

[5]:
```
real_estate_file = 'utseus-anjuke-real-estate.pk'
anjuke = pd.read_pickle(real_estate_file)
```

the pickle is just a list of list. The first item is the columns name, the rest is the data The dataframe need to have the data first, and the name of the columns separately. We use (list comprehension)[https://docs.python.org/3/tutorial/introduction.html#lists] to achieve this.

[6]:
```
anjuke_df = pd.DataFrame(anjuke[1:],columns=anjuke[0], )
```

Use .info or .describe functions to get a better idea of the dataset

[7]:
```
anjuke_df.info()
anjuke_df.describe()
anjuke_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 467029 entries, 0 to 467028
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   id              467029 non-null  object
 1   address         467029 non-null  object
 2   longitude       467029 non-null  object
 3   latitude        467029 non-null  object
 4   bedroom         467029 non-null  object
 5   room            467029 non-null  object
 6   surface         467029 non-null  object
 7   price           467029 non-null  object
 8   onesquaremeter  467029 non-null  object
 9   tags            467029 non-null  object
 10  district        467029 non-null  object
 11  neighborhood    467029 non-null  object
 12  done            467029 non-null  object
dtypes: object(13)
memory usage: 46.3+ MB
```

[7]:
```
            id              address         longitude          latitude  \
0  A888553302   -     ( )(555 )   121.34392735101  31.3193561718426
1  A888376847           -         121.407324884273  31.3023675431043
2  A885088482      -     (255 )  121.397487377268  31.2908711986862
3  A885392981      -      255   121.397487377268  31.2908711986862
4  A885831305          -         121.421397234518  31.302658471085

  bedroom room surface      price    onesquaremeter  \
0       2    2    30.0  1160000.0  38666.6666666667
1       1    1    38.0  1950000.0  51315.7894736842
2       3    2    92.0  5350000.0  58152.1739130435
3       2    2    68.0  4350000.0  63970.5882352941
4       2    2    80.0  4000000.0           50000.0
```

```
                                tags district neighborhood done
0      **               baoshan       dachang    1
1                    baoshan       dachang    1
2  92+7     30  2  11   20   baoshan       dachang    1
3         45           baoshan       dachang    1
4             2           baoshan       dachang    1
```

all of the field we see in the columns are object and have not been correctly imported as numbers. Consequently we cannot calculate with these columns as they are considered as words. We need numbers. To convert from words to numbers we use a special function from python.

All numeric columns shall transformed into floats using pandas (to_numeric)[https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_numeric.html] function.

[8]:
```
numeric_columns = ['longitude', 'latitude', 'bedroom', 'room', 'surface',␣
 ↪'price', 'onesquaremeter']

# put the function in the following line
anjuke_df[numeric_columns] = anjuke_df[numeric_columns].apply(pd.to_numeric,␣
 ↪errors='coerce')
```

now we can have some numbers that could not be converted. the rows containing these undefined values need to be dropped.

Do that using the function (dropna)[https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html]

[9]:
```
anjuke_df = anjuke_df.dropna()
```

check if the colums have been successfully changed.

You should observe that we 'lost' 3962 records that were incomplete with a non numeric data

use .info() again

[10]:
```
anjuke_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 463067 entries, 0 to 467028
Data columns (total 13 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   id              463067 non-null  object
 1   address         463067 non-null  object
 2   longitude       463067 non-null  float64
 3   latitude        463067 non-null  float64
 4   bedroom         463067 non-null  int64
 5   room            463067 non-null  int64
 6   surface         463067 non-null  float64
```

```
 7   price            463067 non-null  float64
 8   onesquaremeter   463067 non-null  float64
 9   tags             463067 non-null  object
 10  district         463067 non-null  object
 11  neighborhood     463067 non-null  object
 12  done             463067 non-null  object
dtypes: float64(5), int64(2), object(6)
memory usage: 49.5+ MB
```

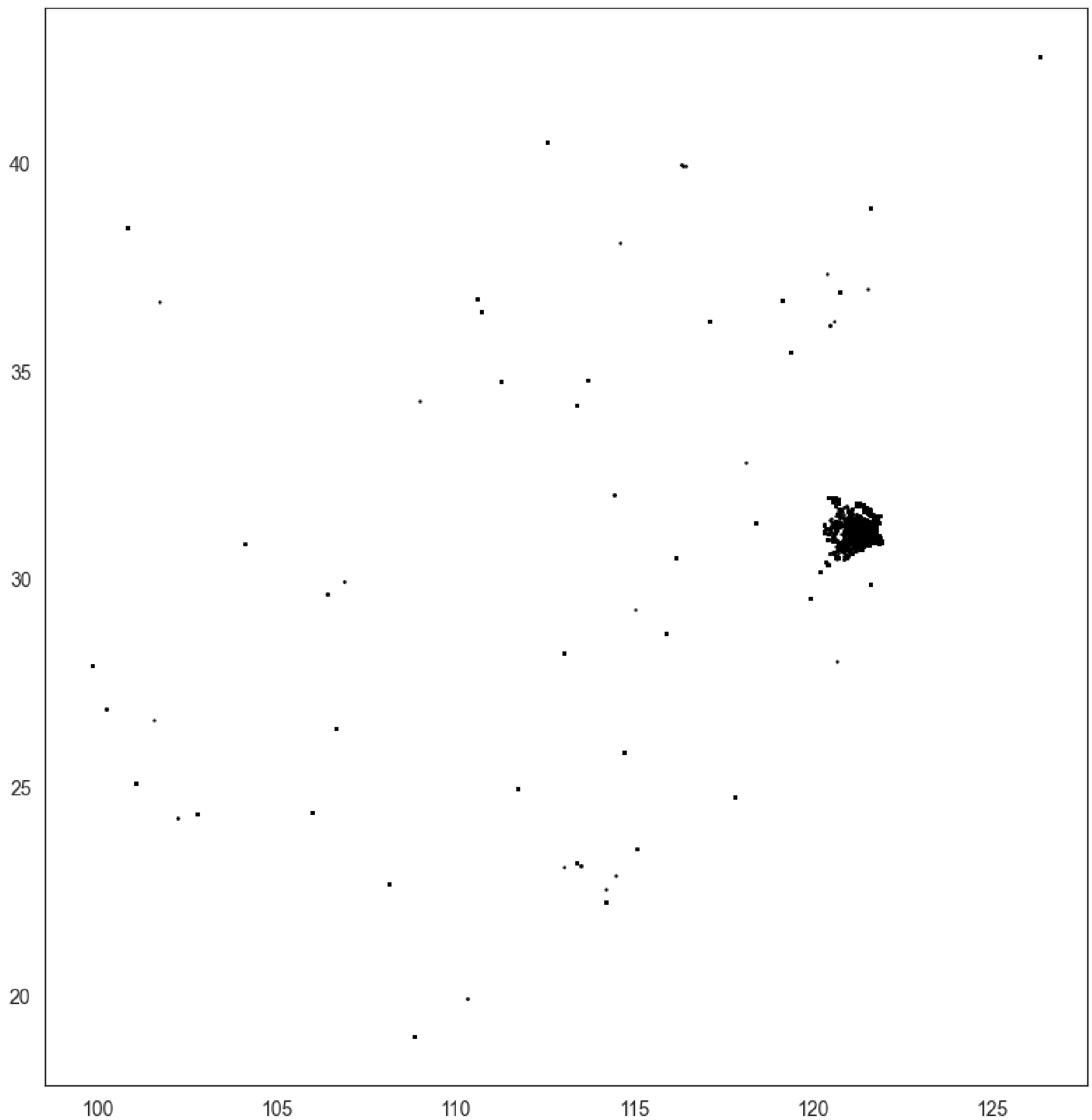
Time to tranform the dataset into a geodataframe now so we can perform geographic operations

Right now we just have GPS has numbers in columns. This is not ready to be used on a map. For that we need to convert it into a geographic object that can be a point, a line or a polygon. In our case it is a point

[11]:
```
anjuke_gdf = gpd.GeoDataFrame(
        anjuke_df,
        geometry=gpd.points_from_xy(anjuke_df.longitude, anjuke_df.latitude),␣
 ↪crs=4326)
```

plot the data on a map using the plot function with the following properties plot(markersize=0.5, color='black', figsize=(10,10))

[12]:
```
import matplotlib.pyplot as plt
anjuke_gdf.plot(markersize=0.5, color='black', figsize=(10, 10))
plt.show()
```



We observe that data are not only in Shanghai.

Write HERE why you think it is like this ?

Sûrement à cause d'erreurs dans les données. On possédait un jeu de données assez conséquent, certains ont pus être mal saisies.

To have an idea of the spread of data, we can print or we can check the boundaries using the property (total_bounds)[https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoSeries.total_bounds.html]

[13]:
```
# use the property and print the result

anjuke_gdf.total_bounds
```

[13]: array([99.82410591, 18.99369628, 126.33039253, 42.54940926])

In both cases, it seems there are data outside Shanghai.

To limit it we have two options: - one is to use *cx* to specify a bounding box - another one is to provide a shape for the area and filter points within

1.3.1 Lets start with bounding box

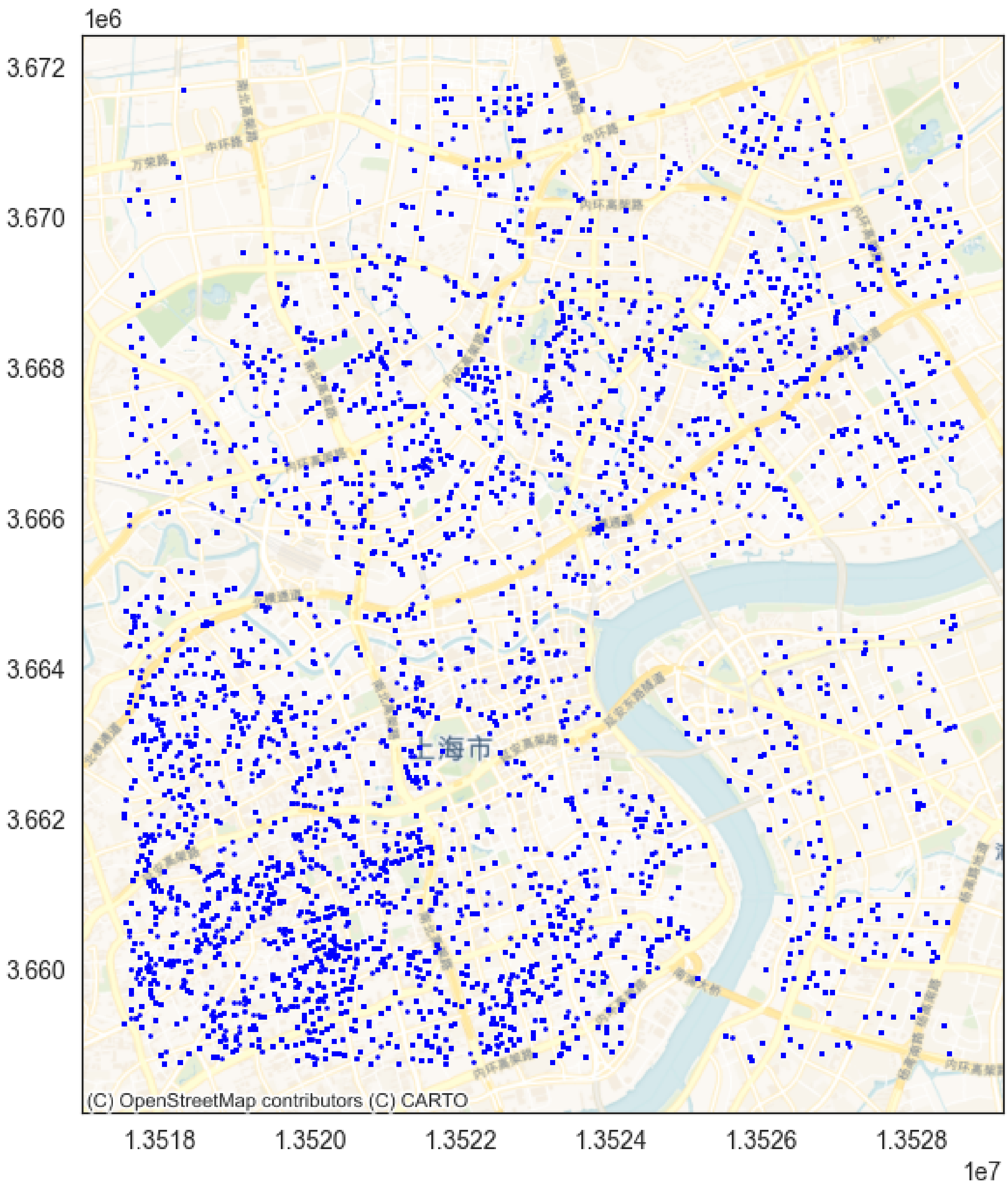
to find the bounding box you can use GIS or search for it. we can use bboxfinder.com for example

[14]:
```
print(f'before {len(anjuke_gdf)}')
west = 120.85
east = 121.99
north = 31.89
south = 30.60

xmin, ymin, xmax, ymax = (west,south,east,north)
anjuke_filterbb_gdf = anjuke_gdf.cx[xmin:xmax, ymin:ymax]
print(f'after {len(anjuke_filterbb_gdf)}')
```

```
before 463067
after 454134
```

we check if the latitude or longitude we have is in GCJ format of GPS format by looking at the huangpu river

[15]:
```
ax = anjuke_filterbb_gdf.cx[121.43:121.53, 31.20:31.30].to_crs(3857).
↪plot(markersize=1, color='b', figsize=(8,8))

# use plot with ax here if you dont have cx
ax.plot(markersize=1, color='b', figsize=(8,8))
# use cx here if you have it
cx.add_basemap(ax,source=cx.providers.CartoDB.Voyager)
```



1.4 2 – POI dataset: Gaode

This dataset has been bought from Gaode for SHU Cendus institute in 2020. We choose the POI from 2017 to be consistant with the anjuke dataset also from 2017

[16]:
```
poi_df = pd.read_csv('shanghaiPOI2017.csv',delimiter=';', decimal=',',␣
↪low_memory=False)
```

[17]:
```
poi_df.columns
```

[17]:
```
Index(['Unnamed: 0', 'ID', 'POIID', 'GCJX', 'GCJY', 'NAME', 'ADD', 'TEL',
       'TYPE', 'AREAID', 'IDKEY', 'GPSX', 'GPSY', 'TYPE1', 'TYPE2', 'TYPE3',
       'TYPE4', 'TYPE5', 'TYPE3.1', 'TYPE3.2', 'TYPE3.3', 'TYPE1.N', 'TYPE2.N',
       'TYPE3.N', 'TYPE3.1.N', 'TYPECODE', 'GBCODE'],
      dtype='object')
```

find the GBCode to all of them

[18]:
```
# write the function to create the geodataframe here usin gthe same method␣
↪as␣before
poi_gdf = gpd.GeoDataFrame(
    poi_df,

    geometry=gpd.points_from_xy(poi_df.GPSX, poi_df.GPSY), crs=4326)
```

[19]:
```
poi_gdf.to_crs(4576, inplace=True)
anjuke_filterbb_gdf.to_crs(4576, inplace=True)
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  super().__setitem__(key, value)
```

[20]:
```
gbcodes = poi_gdf.GBCODE.unique()
len(gbcodes)
```

[20]:
```
323
```

it looks like the subway stations have a GBCode of XXXX.

The complete description of China BGCode used to categorize POI information can be found GB/T 35648-2017

[21]:
```
def what(gbcode):
    return poi_gdf.loc[poi_gdf.GBCODE == gbcode,['TYPE']].head(1)
what(200402)
```

[21]:
```
                 TYPE
69644     ;  ;
```

[22]:
```
anjuke_filterbb_gdf = anjuke_filterbb_gdf[
    (anjuke_filterbb_gdf.onesquaremeter < 3*anjuke_filterbb_gdf.
onesquaremeter.std() )&
    (anjuke_filterbb_gdf.onesquaremeter > -3*anjuke_filterbb_gdf.
```

```
onesquaremeter.std())].copy()
```

[23]:
```
anjuke_filterbb_gdf.geometry.head()
```

[23]:
```
0    POINT (21342354.732 3467628.095)
1    POINT (21348363.197 3465655.143)
2     POINT (21347407.938 3464393.81)
3     POINT (21347407.938 3464393.81)
4    POINT (21349703.629 3465668.134)
Name: geometry, dtype: geometry
```

[24]:
```
final_gbcodes = []
coords2 = np.array([[geom.x, geom.y] for geom in anjuke_filterbb_gdf.geometry])
for gbcode in gbcodes:
    if np.isnan(gbcode):
        continue
    if len(poi_gdf[poi_gdf.GBCODE == gbcode]) < 1:
        continue
    final_gbcodes.append(str(gbcode))

    str(gbcode)
    coords1 = np.array([[geom.x, geom.y] for geom in poi_gdf[poi_gdf.GBCODE␣
 ↪==gbcode].geometry])
    kdtree = KDTree(coords1)
    anjuke_filterbb_gdf[str(gbcode)], _ = kdtree.query(coords2, k=1)
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
```

```
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
```

```
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
```

```
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
```

```
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
```

```
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
```

```
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
```

```
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
```

```
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
```

```
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
```

```
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
```

```
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
```

```
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
```

```
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
```

```
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
```

```
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
```

```
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
```

```
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
```

```
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
```

```
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
```

```
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
```

```
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
```

```
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
```

```
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
```

```
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
```

```
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
```

```
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
```

```
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
```

```
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/geopandas/geodataframe.py:1819: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead. To get a de-fragmented frame, use `newframe =
frame.copy()`
  super().__setitem__(key, value)
```

[25]:

```
for i,t in enumerate(anjuke_filterbb_gdf.columns.to_list()):
    print(i,t)
```

```
0 id
1 address
2 longitude
3 latitude
4 bedroom
5 room
6 surface
7 price
8 onesquaremeter
9 tags
10 district
11 neighborhood
12 done
13 geometry
14 990000.0
15 209900.0
16 201203.0
17 201200.0
18 201202.0
19 201204.0
20 200301.0
21 190600.0
22 130900.0
23 110100.0
24 110503.0
25 201201.0
26 200501.0
```

```
27 200504.0
28 200402.0
29 200401.0
30 200100.0
31 200104.0
32 200102.0
33 200601.0
34 201401.0
35 230100.0
36 170000.0
37 200201.0
38 210501.0
39 219900.0
40 201402.0
41 129900.0
42 160303.0
43 160300.0
44 220100.0
45 220301.0
46 160600.0
47 110500.0
48 120100.0
49 120300.0
50 120400.0
51 120200.0
52 160200.0
53 160112.0
54 160121.0
55 220101.0
56 210800.0
57 210900.0
58 150404.0
59 179900.0
60 119900.0
61 130202.0
62 139900.0
63 190900.0
64 201400.0
65 190000.0
66 240105.0
67 240109.0
68 240300.0
69 240113.0
70 240203.0
71 240202.0
72 190603.0
73 190100.0
74 190401.0
```

75 190104.0
76 190406.0
77 190203.0
78 190202.0
79 190403.0
80 190404.0
81 190407.0
82 190106.0
83 190701.0
84 190501.0
85 190502.0
86 190500.0
87 190503.0
88 190201.0
89 190903.0
90 190902.0
91 190901.0
92 190800.0
93 191100.0
94 201104.0
95 150705.0
96 130100.0
97 240108.0
98 160204.0
99 160113.0
100 131400.0
101 110504.0
102 170401.0
103 160206.0
104 210101.0
105 131003.0
106 160202.0
107 160201.0
108 160400.0
109 150202.0
110 160203.0
111 169900.0
112 160100.0
113 210202.0
114 110501.0
115 110200.0
116 110300.0
117 160302.0
118 160402.0
119 160401.0
120 150405.0
121 160404.0
122 160104.0

```
123 160103.0
124 160111.0
125 160101.0
126 160119.0
127 130802.0
128 160205.0
129 170301.0
130 160114.0
131 160102.0
132 160117.0
133 210111.0
134 130702.0
135 160109.0
136 150101.0
137 160108.0
138 160106.0
139 150106.0
140 160105.0
141 160118.0
142 160116.0
143 160107.0
144 130901.0
145 210204.0
146 150408.0
147 160115.0
148 160120.0
149 211102.0
150 211100.0
151 211103.0
152 211101.0
153 230300.0
154 220201.0
155 220300.0
156 130904.0
157 210801.0
158 211000.0
159 210904.0
160 210107.0
161 211008.0
162 210403.0
163 210401.0
164 150700.0
165 150403.0
166 159900.0
167 150300.0
168 150201.0
169 130102.0
170 131100.0
```

```
171 131001.0
172 130800.0
173 130500.0
174 130201.0
175 180401.0
176 189900.0
177 110502.0
178 240302.0
179 130404.0
180 130603.0
181 130907.0
182 131101.0
183 130304.0
184 130306.0
185 211003.0
186 210402.0
187 230301.0
188 230302.0
189 230304.0
190 230303.0
191 230200.0
192 140500.0
193 140100.0
194 141002.0
195 140511.0
196 140501.0
197 141400.0
198 140508.0
199 140518.0
200 140502.0
201 141402.0
202 140510.0
203 140900.0
204 140503.0
205 140504.0
206 140506.0
207 140516.0
208 141702.0
209 141700.0
210 141701.0
211 141600.0
212 141602.0
213 141601.0
214 141100.0
215 220302.0
216 131103.0
217 131300.0
218 220102.0
```

219 240304.0
220 170300.0
221 150100.0
222 220110.0
223 109205.0
224 130101.0
225 170302.0
226 200507.0
227 201302.0
228 201500.0
229 240200.0
230 240205.0
231 201300.0
232 131301.0
233 201304.0
234 200304.0
235 200305.0
236 240303.0
237 240106.0
238 240112.0
239 240100.0
240 240110.0
241 240204.0
242 240206.0
243 240301.0
244 210806.0
245 130905.0
246 190204.0
247 190101.0
248 190602.0
249 190601.0
250 190400.0
251 190205.0
252 190405.0
253 190300.0
254 190703.0
255 190700.0
256 190200.0
257 180500.0
258 190105.0
259 211001.0
260 190102.0
261 190103.0
262 140600.0
263 130501.0
264 130903.0
265 201110.0
266 201102.0

```
267 201101.0
268 210802.0
269 210805.0
270 210804.0
271 210803.0
272 210701.0
273 210702.0
274 211006.0
275 211002.0
276 211004.0
277 211005.0
278 210504.0
279 210503.0
280 160207.0
281 210201.0
282 130300.0
283 210104.0
284 150402.0
285 130701.0
286 131200.0
287 150703.0
288 130804.0
289 150704.0
290 150702.0
291 150701.0
292 150401.0
293 150406.0
294 150500.0
295 150407.0
296 150103.0
297 150102.0
298 150105.0
299 150107.0
300 150409.0
301 131404.0
302 131107.0
303 130602.0
304 131102.0
305 131202.0
306 130703.0
307 130601.0
308 131105.0
309 130503.0
310 130302.0
311 210500.0
312 130806.0
313 130103.0
314 130502.0
```

```
315 131104.0
316 130307.0
317 180404.0
318 180405.0
319 240000.0
320 130305.0
321 130303.0
322 131201.0
323 200901.0
324 201002.0
325 201001.0
326 180100.0
327 180200.0
328 180402.0
329 180403.0
330 170303.0
331 170204.0
332 170304.0
333 170305.0
334 170105.0
335 170201.0
```

[40]:
```
'''
on fait ça pour vérifier si les distances sont correctes.
Avant, il prenait en compte que la latitude et la longitude,
alors que ça dépend également du type d'endroit, du prix, etc.
'''
```

[40]:
```
"\non fait ça pour vérifier si les distances sont correctes.\nAvant, il prenait
en compte que la latitude et la longitude,\nalors que ça dépend également du
type d'endroit, du prix, etc.\n"
```

[37]:
```
what(990000.0) #activitées évenementielles
```

[37]:
```
            TYPE
0    ;  ;
```

[29]:
```
# On sélectionne uniquement les colonnes correspondant aux distances aux POI
poi_columns = [col for col in anjuke_filterbb_gdf.columns if col.isdigit()]

# On extrait uniquement ces colonnes
poi_data = anjuke_filterbb_gdf[poi_columns]

# On vérifie que toutes les colonnes contiennent bien des valeurs valides
poi_data = poi_data.dropna(axis=1)
```

À ce stade, nous avons calculé la distance entre chaque appartement et chaque type de Point of Interest (POI), identifié par un code unique appelé GBCODE. Cependant, tous les POI ne sont pas représentés de manière significative dans la base de données. Nous filtrons donc les colonnes associées aux POI afin de ne conserver que celles contenant des données valides pour effectuer notre analyse.

[34]:
```
print(poi_data.shape)
print(poi_data.head())

print(anjuke_filterbb_gdf.columns.tolist()[:20])  # ou [:50]

# On sélectionne les colonnes dont le nom est un nombre float (sous forme de␣
↪string)
poi_columns = [col for col in anjuke_filterbb_gdf.columns
               if isinstance(col, str) and col.replace('.', '', 1).isdigit()]

# On extrait les colonnes et supprime celles qui contiennent trop de valeurs␣
↪manquantes
poi_data = anjuke_filterbb_gdf[poi_columns].dropna(axis=1, thresh=int(0.95 *␣
↪len(anjuke_filterbb_gdf)))
```

```
(449599, 0)
Empty DataFrame
Columns: []
Index: [0, 1, 2, 3, 4]
['id', 'address', 'longitude', 'latitude', 'bedroom', 'room', 'surface',
'price', 'onesquaremeter', 'tags', 'district', 'neighborhood', 'done',
'geometry', '990000.0', '209900.0', '201203.0', '201200.0', '201202.0',
'201204.0']
```

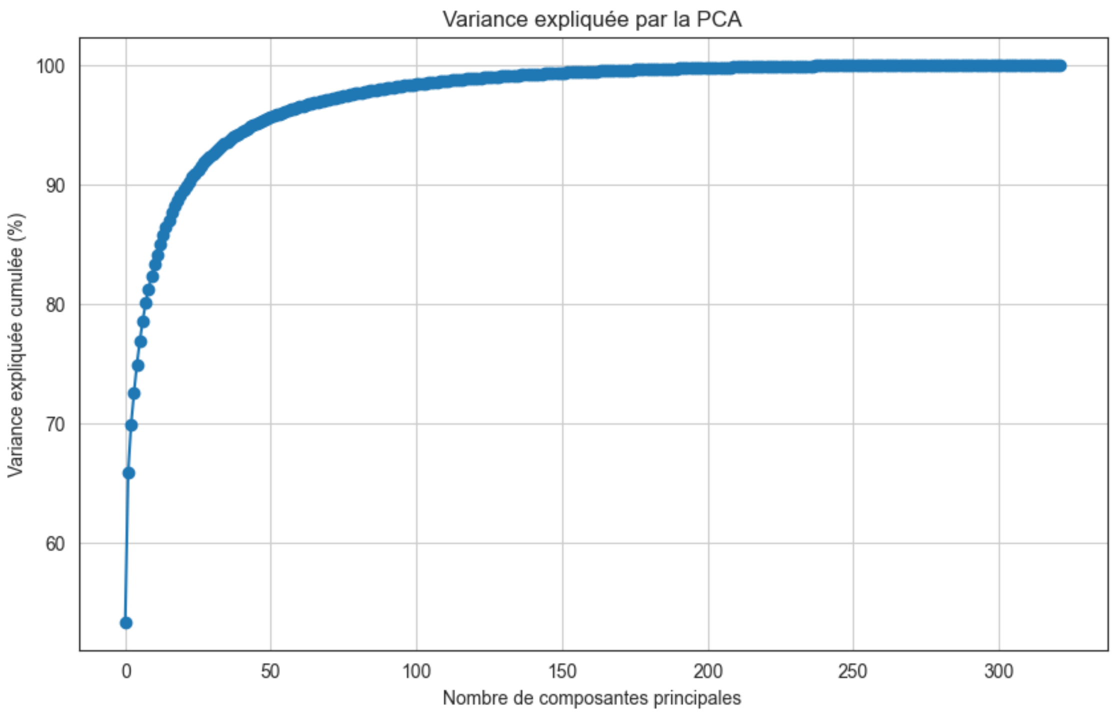
[35]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Standardisation des données (obligatoire avant PCA)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(poi_data)

# PCA
pca = PCA()
X_pca = pca.fit_transform(X_scaled)
```

La PCA (analyse en composantes principales) est une technique de réduction de dimensionnalité. Elle permet d'identifier les directions (composantes principales) qui capturent le plus de variance dans les données. Avant de l'appliquer, nous standardisons les distances pour que toutes les colonnes aient la même échelle, car une distance exprimée en mètres aurait plus de poids qu'une autre simplement à cause de son unité.

3. Visualiser la variance expliquée par les composantes

[36]:
```
plt.figure(figsize=(10,6))
plt.plot(np.cumsum(pca.explained_variance_ratio_)*100, marker='o')
```

```python
plt.xlabel('Nombre de composantes principales')
plt.ylabel('Variance expliquée cumulée (%)')
plt.title('Variance expliquée par la PCA')
plt.grid(True)
plt.show()
```



Le graphique ci-dessous montre la proportion de variance expliquée par les différentes composantes principales. Cela nous permet de déterminer combien de dimensions sont réellement nécessaires pour représenter efficacement nos données. Dans notre cas, les deux premières composantes capturent environ X % de la variance totale (à ajuster selon ton graphique).

4. Afficher les POI les plus influents (Top 10 dans PC1)

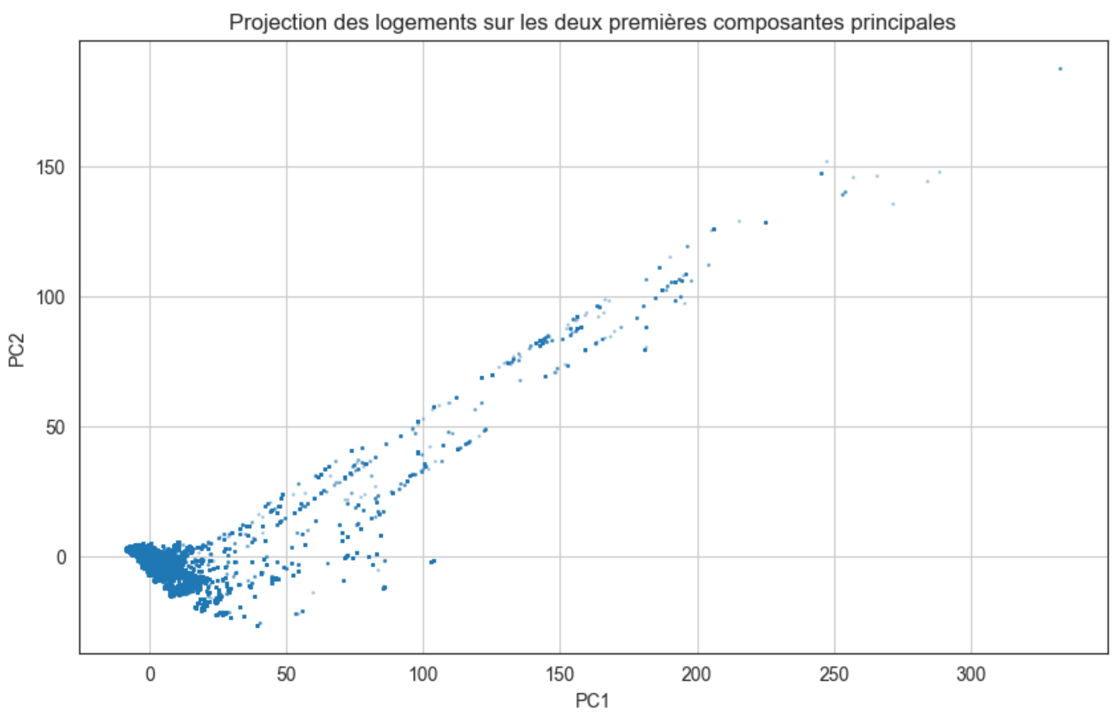
[37]:
```python
# Récupérer les poids de chaque feature dans PC1
loading_scores = pd.Series(pca.components_[0], index=poi_data.columns)
sorted_loading = loading_scores.abs().sort_values(ascending=False)

top_10_features = sorted_loading.head(10).index
print("Top 10 POI les plus influents dans la première composante :")
print(top_10_features)
```

```
Top 10 POI les plus influents dans la première composante :
Index(['210202.0', '130802.0', '130703.0', '130800.0', '210201.0', '130500.0',
       '150700.0', '189900.0', '110503.0', '150405.0'],
      dtype='object')
```

Chaque composante principale est une combinaison linéaire de l'ensemble des POI. En analysant les poids (ou loadings) des features dans la première composante, nous identifions les POI qui expliquent le plus la variance. Voici les 10 POI les plus influents dans PC1, c'est-à-dire ceux dont la présence ou l'absence à proximité influence le plus la classification des logements dans notre analyse.

5. Visualiser les logements projetés sur PC1 et PC2

[38]:
```
plt.figure(figsize=(10, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], s=1, alpha=0.3)
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Projection des logements sur les deux premières composantes␣
 ↪principales')
plt.grid(True)
plt.show()
```



Nous projetons chaque appartement sur les deux premières composantes principales (PC1 et PC2). Cette visualisation 2D permet de détecter des regroupements naturels, voire des tendances spatiales dans les données. Les points proches dans ce graphique ont des profils similaires en termes de distances aux différents POI.