Shahid Beheshti University

Anita Soroush

Dataset : apartment-rental-offers-in-germany

The raw data consists of 49 columns and 268850 rows.

Data Cleaning:

The features are listed below and alongside the percentage of null values for each feature:

regio1	0.000000
serviceCharge	2.569834
heatingType	16.684397
telekomTvOffer	12.132788
telekomHybridUploadSpeed	83.254603
newlyConst	0.000000
balcony	0.000000
picturecount	0.000000
pricetrend	0.681421
telekomUploadSpeed	12.407662
totalRent	15.070485
yearConstructed	21.218151
scoutId	0.000000
noParkSpaces	65.388879
firingTypes	21.188023
hasKitchen	0.000000
geo_bln	0.000000
cellar	0.000000
yearConstructedRange	21.218151
baseRent	0.000000
houseNumber	26.415473
livingSpace	0.000000
geo_krs	0.000000
condition	25.474800
interiorQual	41.906267
petsAllowed	42.615957
street	0.000000
streetPlain	26.413614
lift	0.000000
baseRentRange	0.000000
typeOfFlat	13.618747
geo_plz	0.000000
noRooms	0.000000
thermalChar	39.615399
floor	19.084620

numberOfFloors	36.351869
noRoomsRange	0.000000
garden	0.000000
livingSpaceRange	0.000000
regio2	0.000000
regio3	0.000000
description	7.344988
facilities	19.685326
heatingCosts	68.191185
energyEfficiencyClass	71.066766
lastRefurbish	69.979171
electricityBasePrice	82.575414
electricityKwhPrice	82.575414
date	0.000000

Considering the number of null values and the definition of each feature, I decided to drop some columns at the beginning. The green features are those that I picked to keep and work on.
So the columns are restricted to the following list:

regio1	0.000000
geo_plz	0.000000
heatingType	16.684397
newlyConst	0.000000
yearConstructed	21.218151
cellar	0.000000
livingSpace	0.000000
condition	25.474800
typeOfFlat	13.618747
noRooms	0.000000
garden	0.000000
totalRent	15.070485
hasKitchen	0.000000
lift	0.000000
floor	19.084620

Moving on to **outliers**, these are the numeric features that should be cleaned of outliers: **livingSpace, noRooms, totalRent** and **floor.** (4 columns)

I use 3 standard deviations, but in 3 different ways.

1) One loop **without chunking:**

number of records before putting outliers aside: 268850
whole run time: **0.1577**
number of records after putting outliers aside: 187513

2) **Chunking up** the dataset into **two parts** and detecting their outliers in two **sequential** loops:

number of records before putting outliers aside: 268850
whole run time: **0.3365**
number of records after putting outliers aside: 187513

The result is totally reasonable, because we are processing the two parts consecutively and no parallel.

3) **Chunking up** and **parallelism.**

number of records before putting outliers aside: 268850
run time mean: **3.5848**
number of records after putting outliers aside: 187513

It's a little weird but the result got even worse. It may be because of the runtime overhead related to parallelizing.

There is no algorithmic difference between the 3 ways above; So at the end of each of them and dropping the outliers, we've got **187513 records** to keep, out of 268850.

In order to handle the **null values**, I’m gonna fill the NaN’s of the columns "heatingType", "condition" and "typeOfFlat" with the fixed expression of “NotAvailable”. To achieve this goal, I use 2 different ways.

1) One loop **without partitioning:**

run time : **0.10694**

2) **parallelism** using **Dask:**

Since the columns are independent from each other, I processed them separately at the same time.

run time : **0.00099**

The difference made by dask is impressively significant.

There are still some null records in the feature “yearConstructed” that I’m gonna drop them all.

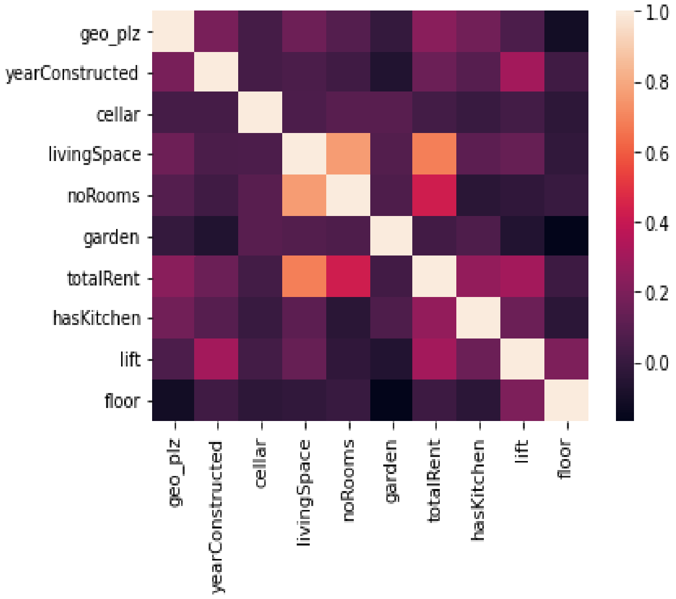
We can also see that 4 records have a “yearConstructed” feature of bigger than 2022, which is **logically impossible**. I drop those rows as well.

After all these cleaning steps the dataset **involves** some duplicated records that I drop altogether.

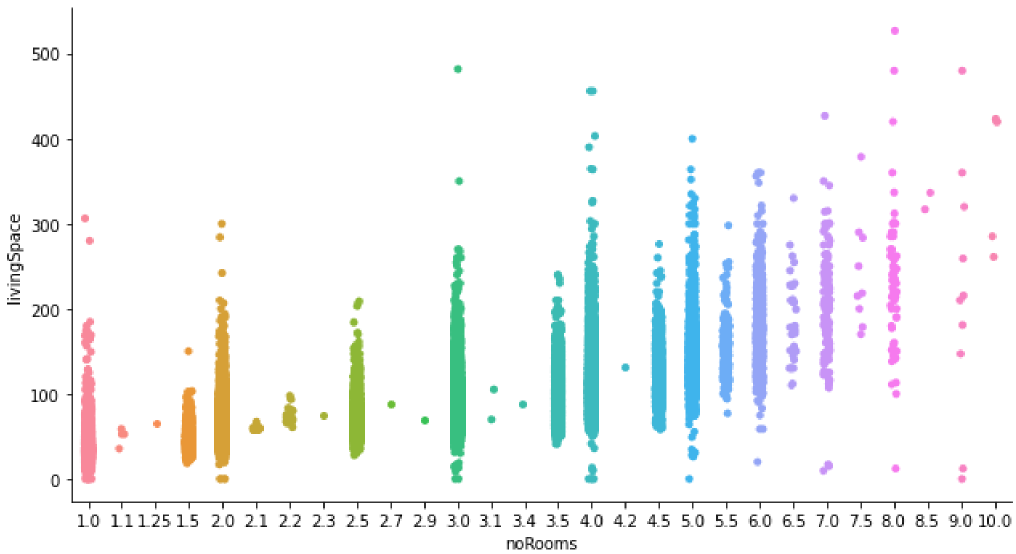
Up to now, we’ve got 146112 rows and 18 columns of cleaned data.

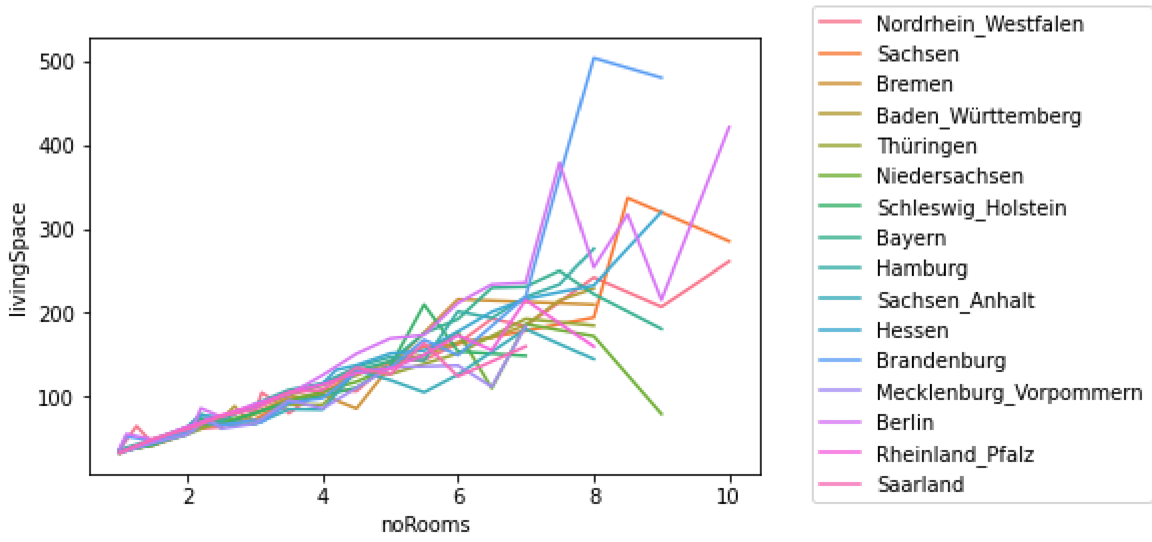
EDA

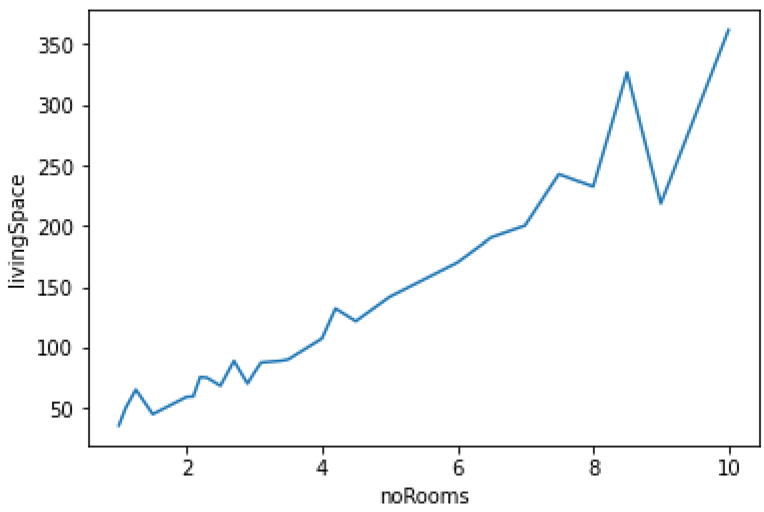
Let’s Start with a correlation heatmap diagram:



The correlation between noRooms and living Space is noticeable. The bigger the living space, the more rooms. The following plots acknowledge this point:

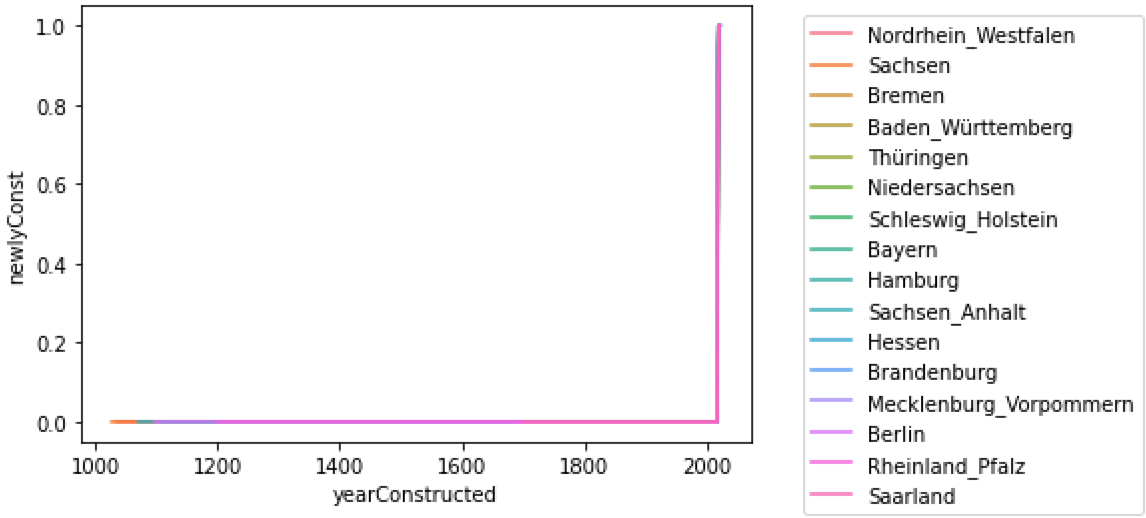




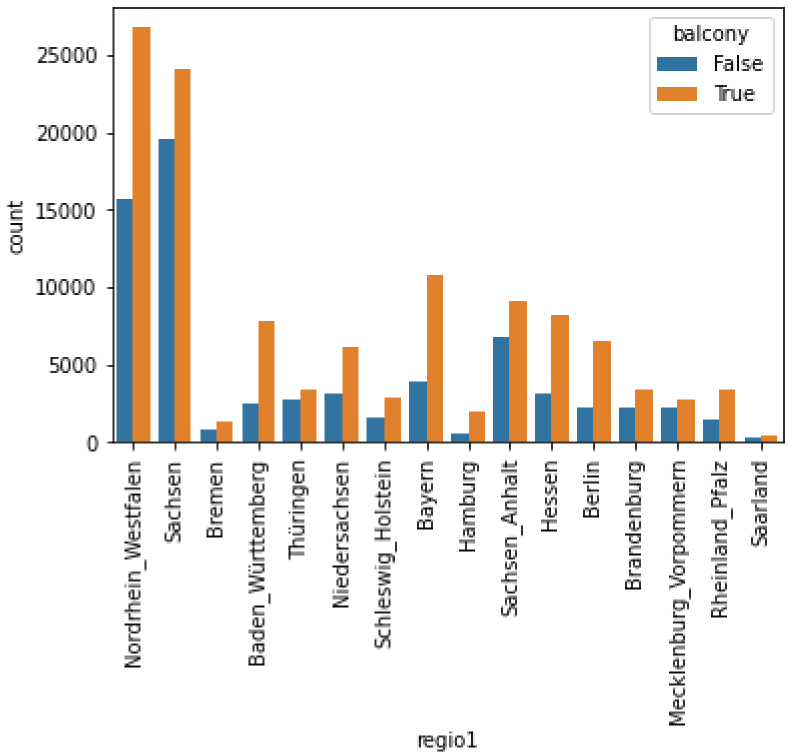


In order to reduce the dependency between the features, I'm gonna drop the noRooms column, because it is represented by livingSpace.

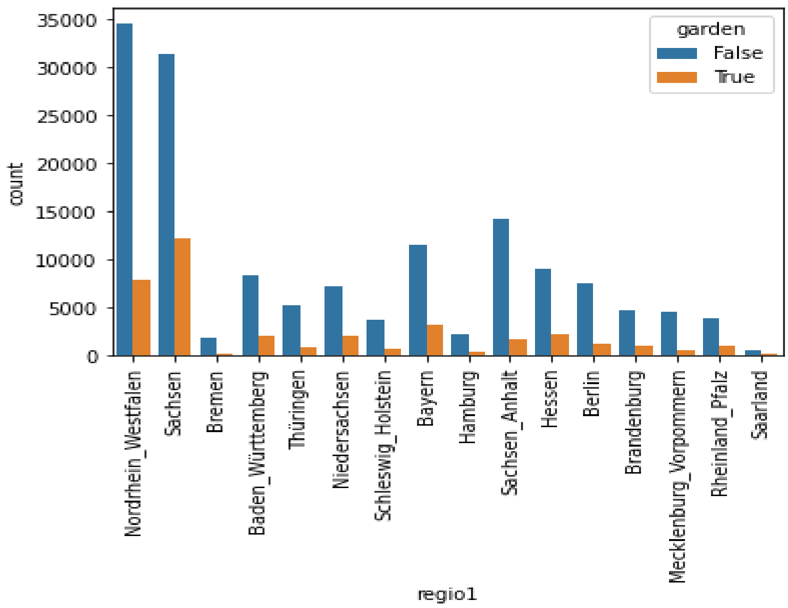
The Line plot below shows that the feature "newlyConst" is 1 only if the construction year is bigger than 2020. So the information of newlyConst can be totally represented by yearConstructed and I'm gonna drop this column as well.



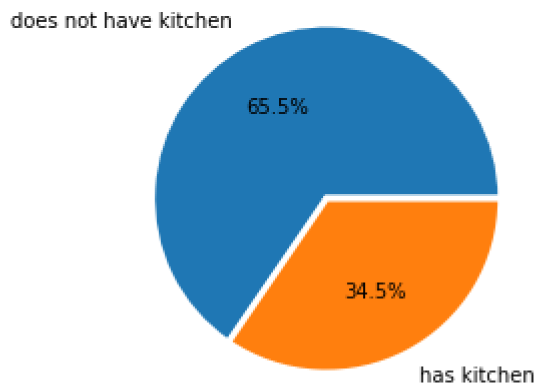
The following barplot shows that in every region the number of apartments that do have balcony is more than the number of those that do not:



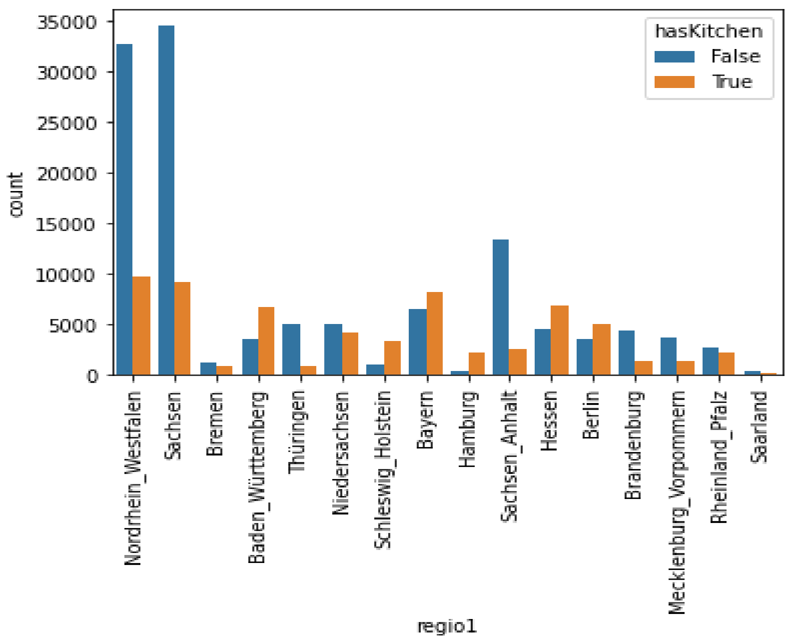
And the apposite is held about garden:



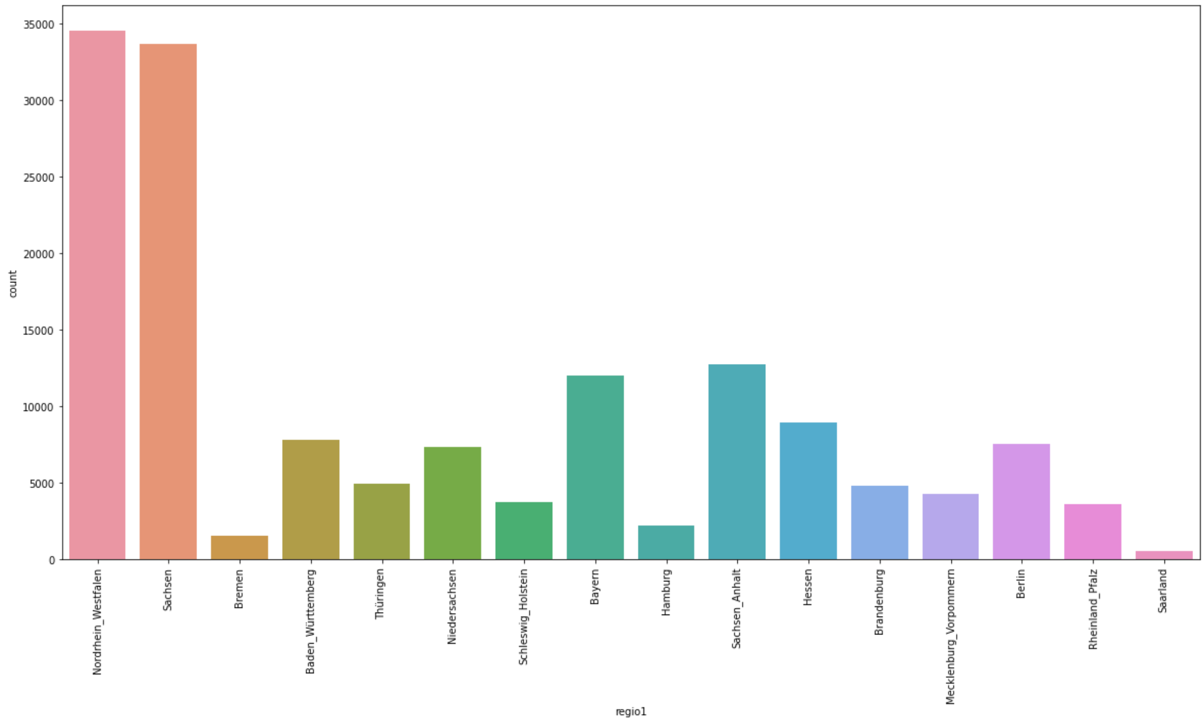
How about the kitchen?
Let's just briefly take a look at the pie chart below:



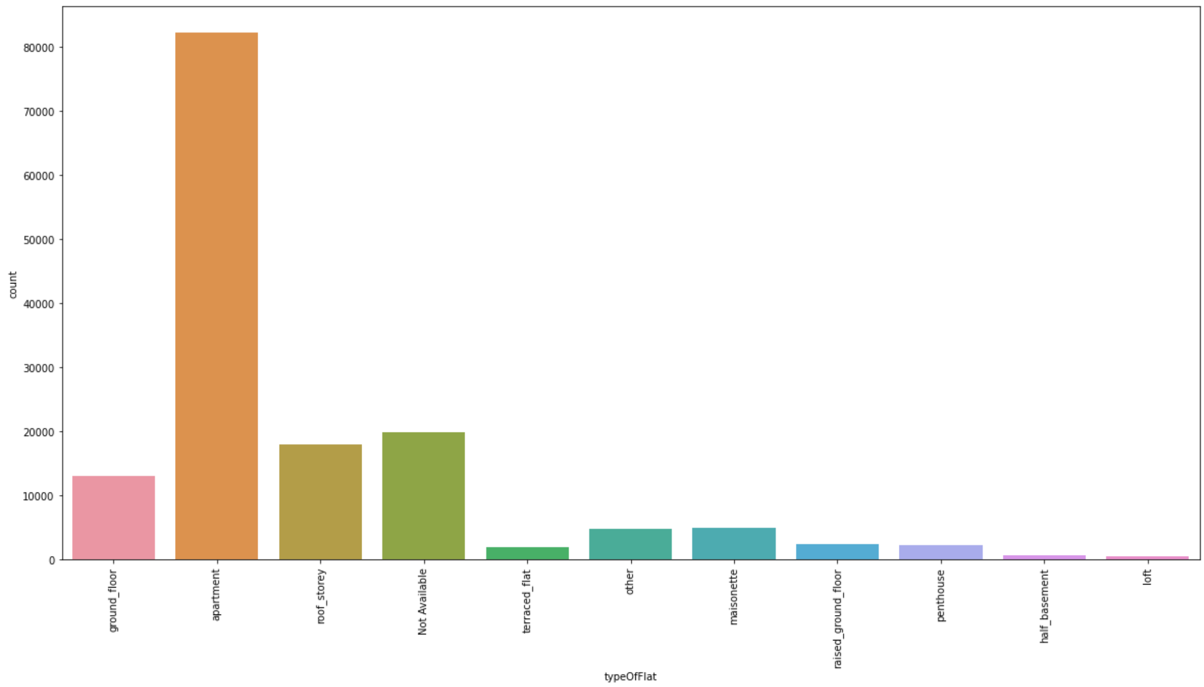
A significantly large proportion of apartments do not have kitchen and based on the following barchart, there is no general rule held for all regions:



The following count plot shows that Nordrhein-Westfalen and Sachsen respectively have the most number of apartment offers among other regions.



And also the apartments make up the biggest part between types of flats:



Model

I used LinearRegression and trained the model using 80% of the data and tested the model on 20% of the data and the result is as follows:

MAE: 173.47531706557996
MSE: 257243.24401621058
R2_score: 0.453644510212342

This model was based on all features, to make the problem a bit more challenging, I want to make a model just using **telekomUploadSpeed**, **serviceCharge** and **heatingType**.

telekomUploadSpeed and serviceCharge are both numerical. But let's take a look at heatingType:
First of all, it is a categorical feature. So if we want to use Regression methods we should apply one hot encoder.
Not only this, the number of categories is 14 which is a relatively large number that can have a negative effect on the regressor performance.

The name of each category and its number of repetition is reported below:

central_heating	65368
NotAvailable	14599
district_heating	14562
gas_heating	10293
self_contained_central_heating	9042
floor_heating	7544
oil_heating	2507
heat_pump	1119
combined_heat_and_power_plant	967
night_storage_heater	689
wood_pellet_heating	459
electric_heating	371
stove_heating	147
solar_heating	95

I merged the gray categories into one category named “other”:

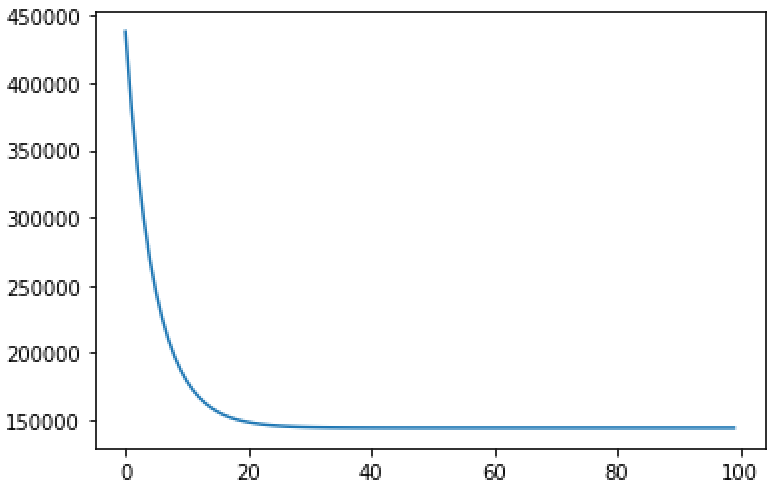
```
central_heating     65368
other               22940
NotAvailable        14599
district_heating    14562
gas_heating         10293
```

Now we can simply use one hot vector encoder and then use different regression methods to see the result.

a) Implementation of Logistic Regression

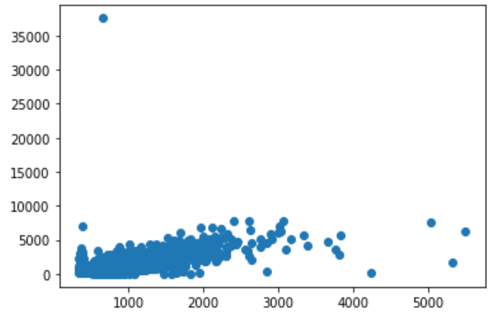
After implementation I initialized the parameters as follow:
w → initialized to zeros
b → initialized to zeros
learning rate → 0.1
epochs → 100

In the following plot, the x-axis is the epoch number and the y-axis is the cost value for the related epoch. It seems that after around 50 iterations the cost value doesn’t change that much. So we better set the epoch parameter as 50 in order to prevent wasting of resources.



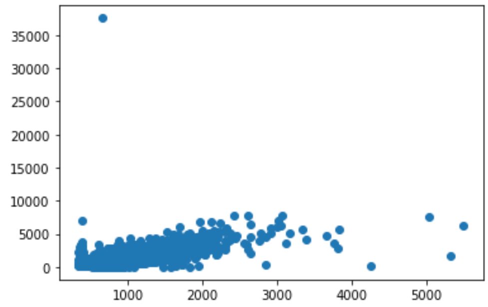
Now let's see the metrics:

MAE: 260.7461396938489
MSE: 209840.88791046795
R2_score: 0.3847120494063474



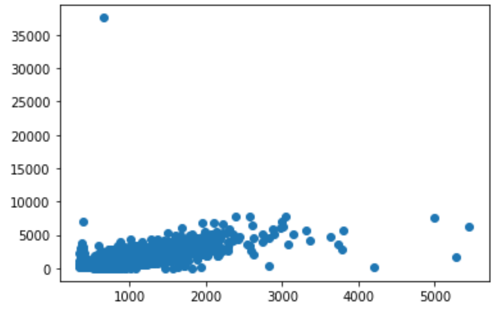
b) Using sklearn implementations:

MAE: 260.7036399979541
MSE: 209843.188996771
R2_score: 0.38470530224334387



c) Ridge Regression:

MAE: 261.39921076264426
MSE: 210514.85394568322
R2_score: 0.38273587028936684



d) Lasso Regression:

MAE: 261.6697035216319
MSE: 210605.12698717648
R2_score: 0.3824711749039854