NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS
International College of Economics and Finance

Danila Karapsin

Anton Ilchuk

Rental prices forecasting

Project

38.04.01 ECONOMICS

Master's Programme **'Financial Economics'**

Moscow 2024

Contents

1 Introduction

Our project is devoted to the rental prices forecasting using both geographical features (geo features) and apartments' features (e.g. floor, square meters, etc). Note that we are not trying to predict future prices, instead we predict current prices based on a dataset which consists of ads placed on the cian.ru at the end of September 2024. So, there is no time-series component in our project.

As has been mentioned, we parsed a dataset from Cian.ru, Russia's leading real estate platform. Data collection was performed using the Python package cianparser, which allowed us to scrape property listings. We encountered several challenges during the scraping process. For example, the platform limits the number of pages that can be accessed for a single query to 54, with each page displaying 28 listings. We solved this by filtering ads by proximity to certain subway stations (cianparser has a list of all train/subway stations in Moscow). However, Cian employs IP-blocking mechanisms to limit scraping, so the script for ads collection is only semi-automatic.

The initial dataset includes over 10,000 listings (after duplicates removal), with each entry containing apartments' features such as rental price, the number of rooms, floor level, total area in square meters, and location information (district, street). To get coordinates for each flat we used its address and capabilities of packages DaData and geopy (if first fails, the second one usually does the job). For problematic entries, manual corrections were made. We also identified and removed outliers, such as listings with unrealistically low or high rental prices, and logged these cases in outliers_to_delete.csv for transparency and reproducibility.

2 Algorithms selection, metrics and submission plan

For the second submission our goals are the following:

- **Remove outliers and fix duplicates:** some ads are duplicated, i.e. multiple ads from different real estate agents for the same apartment. And we also do not want to have too many luxury apartments in our dataset, since we can not predict prices for them using our features.
- **Feature engineering:** work with existing features and add new geo features using apartments' coordinates (proximity to subway stations, parks and other objects)
- **Feature selection and transformation:** some features, like number of rooms, should be transformed to categorical.
- **Exploratory data analysis**
- **Holdout sample selection.**
- **Preliminary results from models' fitting:** our plan is to try Random Forest, KNN and Gradient Boosting algorithms. At this submission we are planning to provide results from fitting at least 1 algorithm.

We use absolute errors for scoring during cross validation to select the best model. In terms of business goals it is important to have lower absolute error since we want to have more accurate predictions. However, as a business metric we may use an interval from 5% quantile to 95% quantile calculated on errors from using our model on the holdout sample. It is much easier to explain such a metric to managers, e.g. in 90% of cases our error is within that interval.

3 Outliers and missing values

Note that some apartments are close to many subway stations, hence we may have duplicates. However it is quite easy to get rid of them, see the **data_consolidation.py** file for details. We also removed ads in which street name, house number, floor count and other apartments' features are not specified. Also for some apartments rental prices are too low, in most cases people are trying to rent a room instead of the whole apartment (see the **outliers_to_delete.csv** file for details). As has been mentioned, for some flats we have multiple ads from many real estate agents. Our methodology to deal with that problem is the following: if we have more than one ad at the same coordinates and floor, we are averaging apartments' features (see flats_data_final_clean.py for details). To get rid of luxury real estate we are removing apartments for which rental prices are higher than 95% quantile (that is done in the geo_features_creation.py file which is mostly devoted to feature engineering).

4 Feature engineering

In our project geo features are the related to the proximity, area or count of certain geographical objects, such as

- Schools and vocational schools
- Boilers, dumps, factories, thermal power plants, waste plants
- Bus stops
- Hospitals
- Subway and train stations
- Train and bus terminals

For almost all features except 2) we used data.mos.ru. For ecological geo features, we used https://mwmoskva.ru/ekologicheskaya-karta-moskvy.html page, which has a map of such places. Using F12 key it is quite easy to download json file and parse it for our needs (see the eco_load.py file for details)

When data is collected, we only need to compute needed features. There are 3 ways to do it:

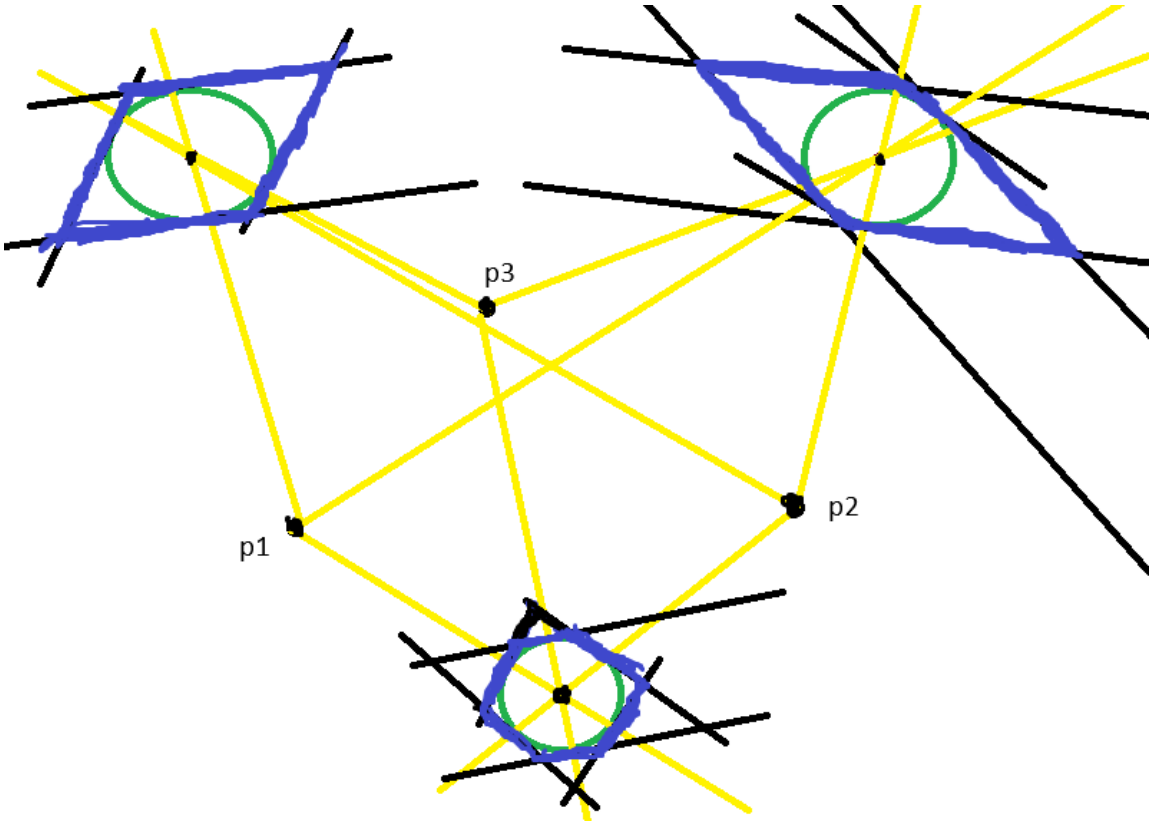
1. **«Brute force»**: every time, for every geo object we may calculate the distance between it and every flat. And then compute required geo features. Note that for some objects (e.g. parks) we have not points, but polygons. In that cases we are checking distance from the flat to every point which forms that polygon. No need to explain why that it is very computationally intensive.

2. **«Clustering»**: we can group flats and geo objects by clusters and then compute distances and geo features. We are reducing computational intensity, but losing some information: some apartments may be located at the edge of the geo sector, and it can be that there are a lot of geo objects in the neighbouring sector.

3. **«Wise way»**: we need to select 3 points in such a way that we can not draw a line through all of them. Then we are computing a distance from every geo object to that 3 index points. Then, suppose that we want to count number of shops within a 1 km radius from a certain flat.

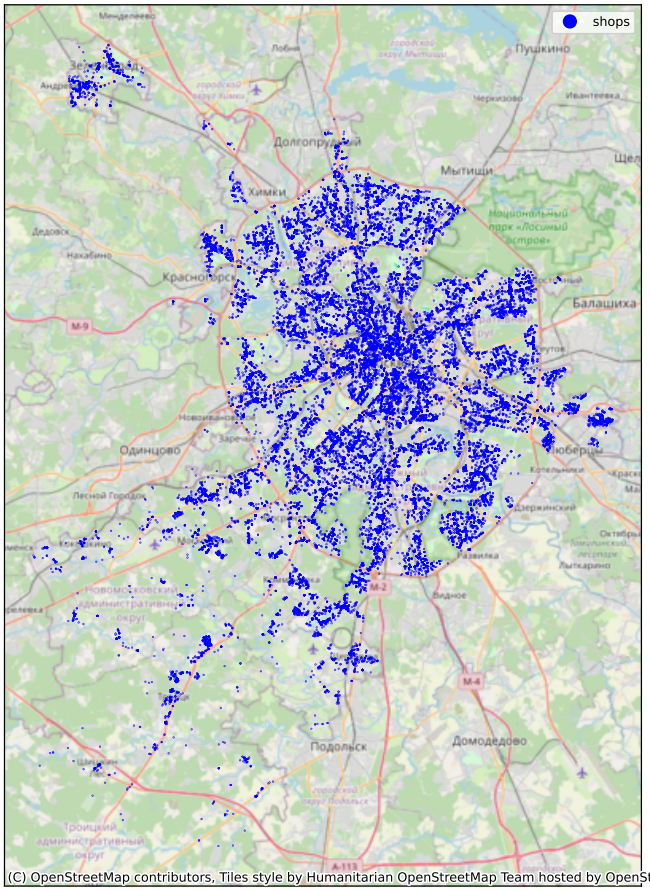
   Suppose that the distance from that apartment to the 1st point is 10km. Then it is easy to see that if the shop is within 1 km radius from the flat its distance from the 1st point should be between 9km and 11km. We are applying the same

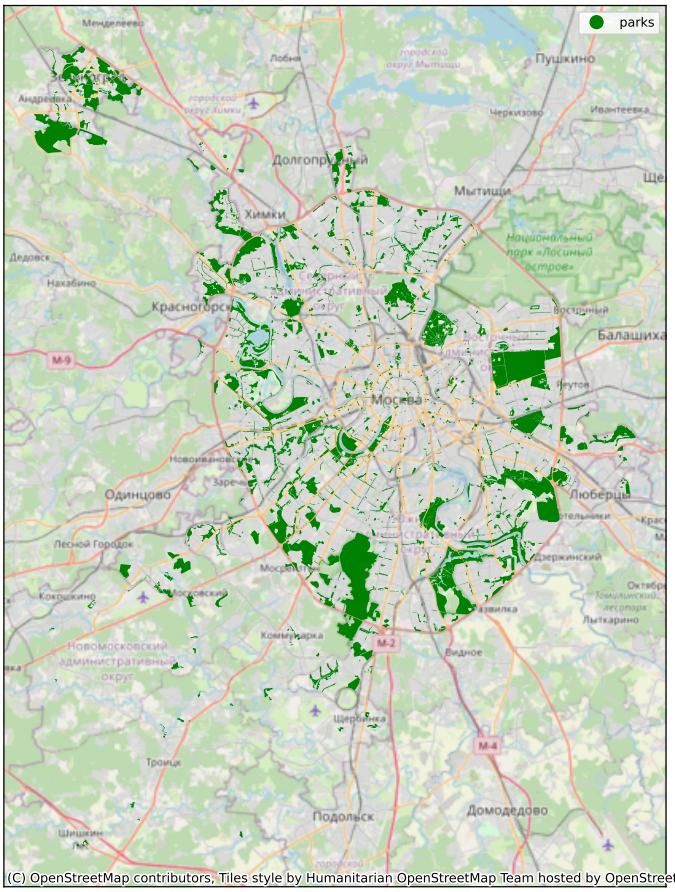
principle for all of our index points. Then instead of computing distance from our apartment to every flat we can just filter index point distances. To see how it limits our search area see the sketch below (blue - search areas, p1, p2, p3 - index points, green - ideal search area).

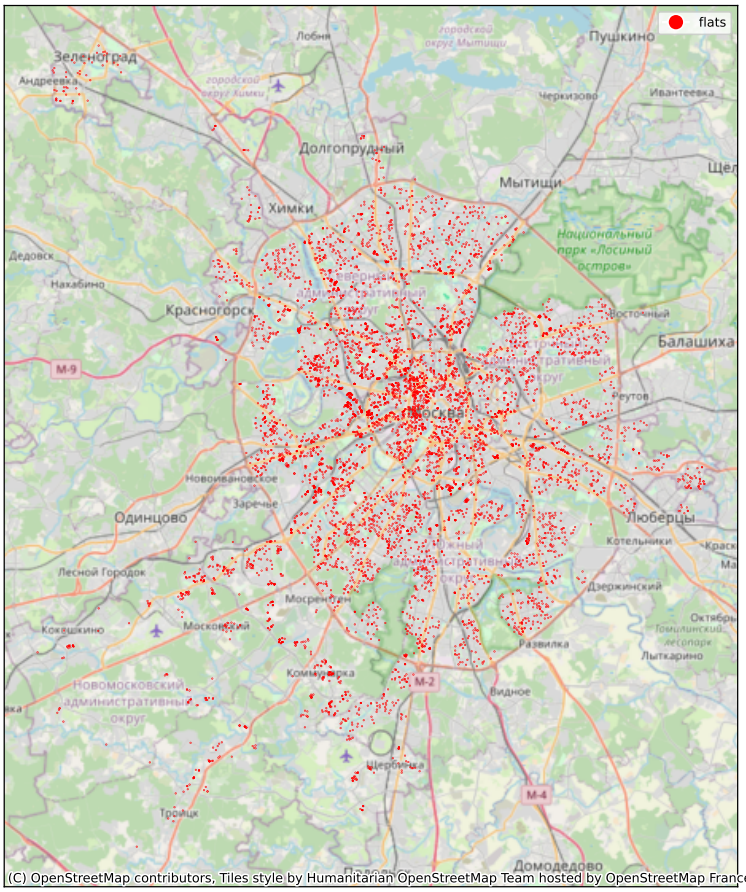


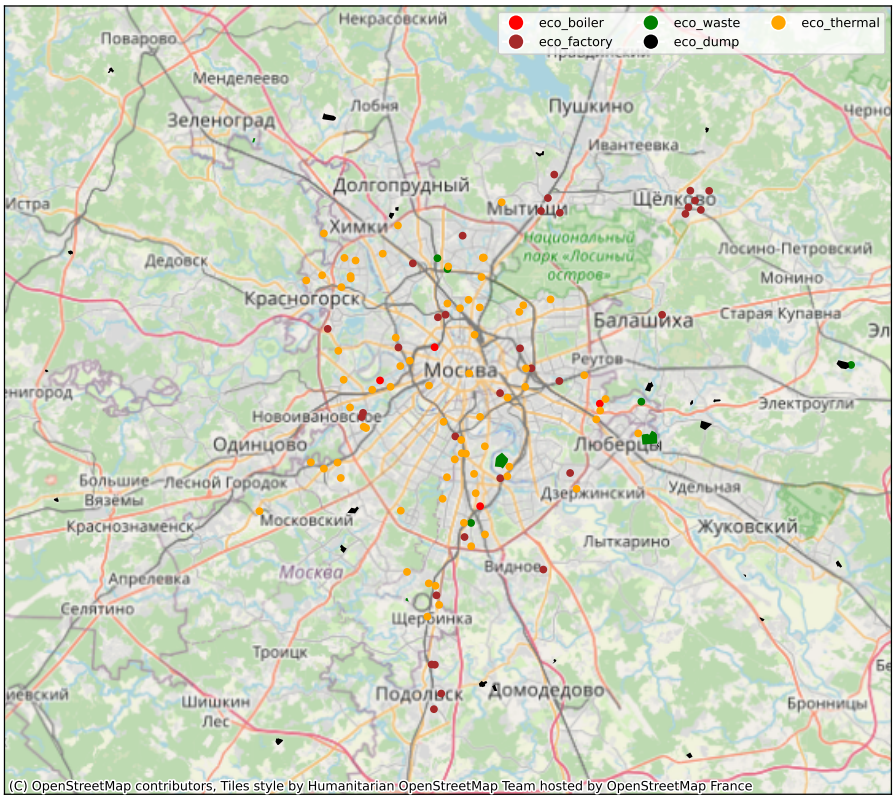
We did not want to lose any info, that is why we did apply a clustering approach. Smart way has been discovered too late. That is why we used a brute force approach in our project, it took almost 4 days to complete computations, but now we know how to do it fast in a wise way.

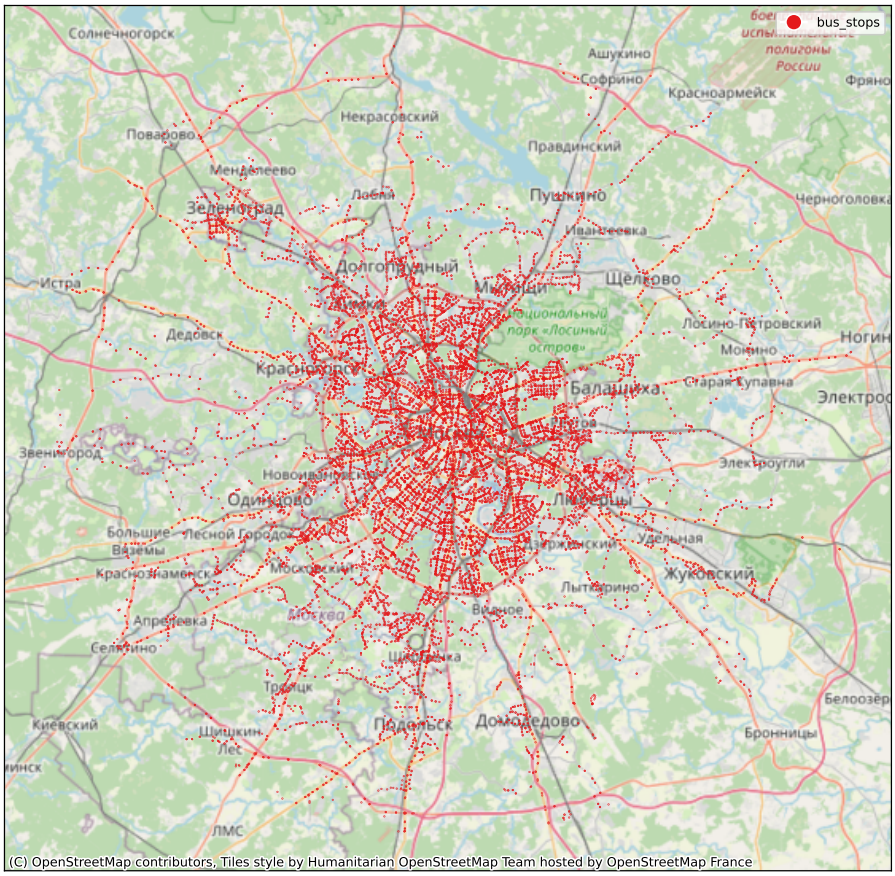
5 EDA, map plots

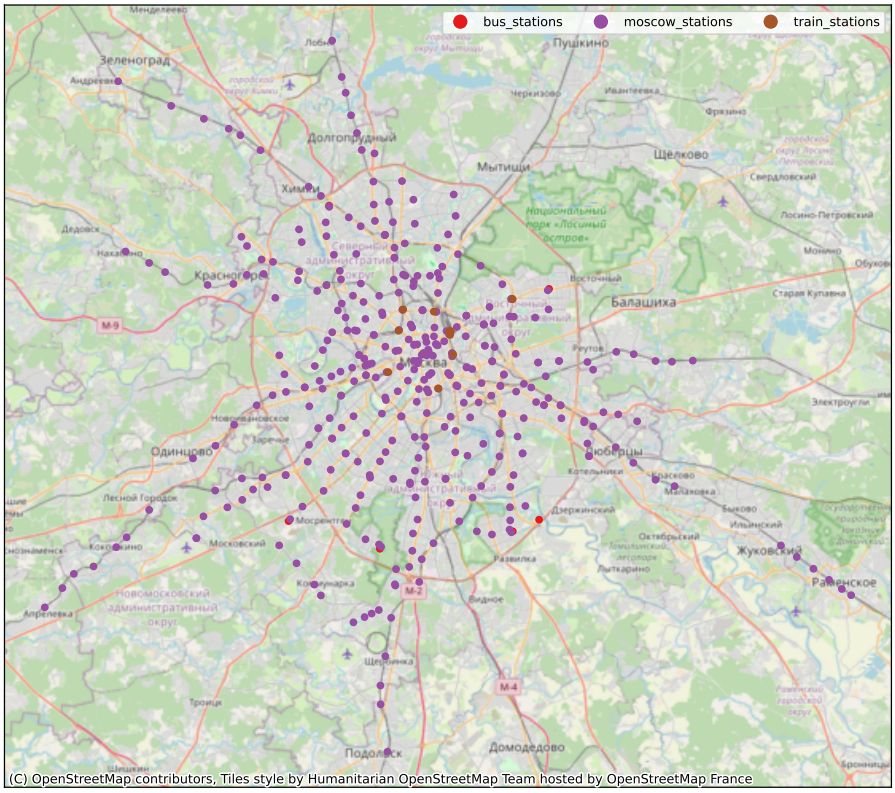


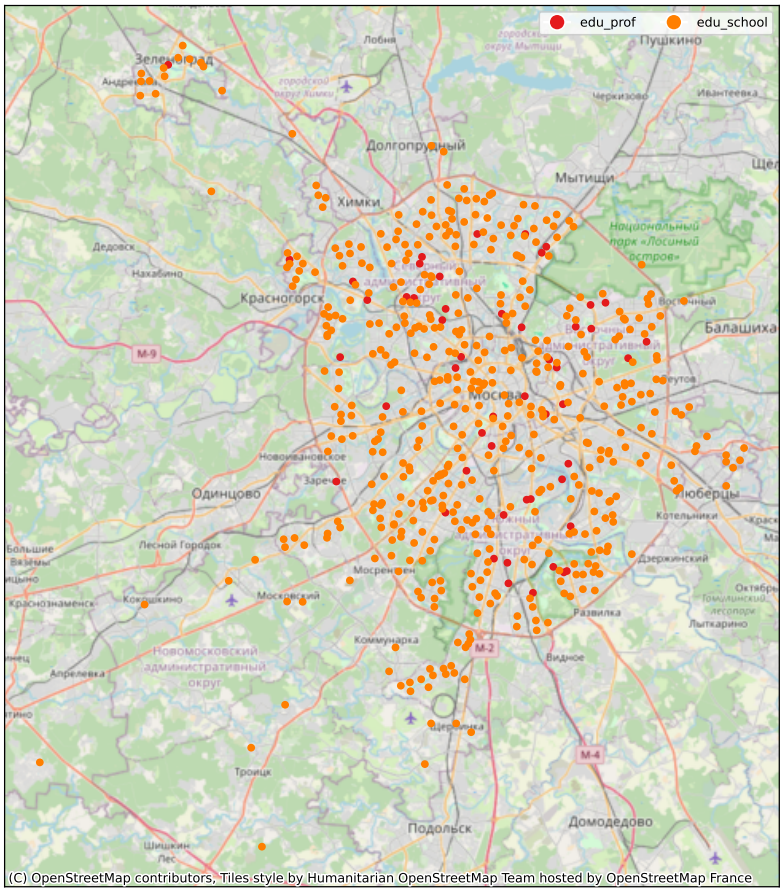

parks
Пушкино
Химки
Мытищи
Балашиха
Москва
Одинцово
Люберцы
Красногорск
Подольск
Домодедово
Видное
(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap Team hosted by OpenStreetMap France

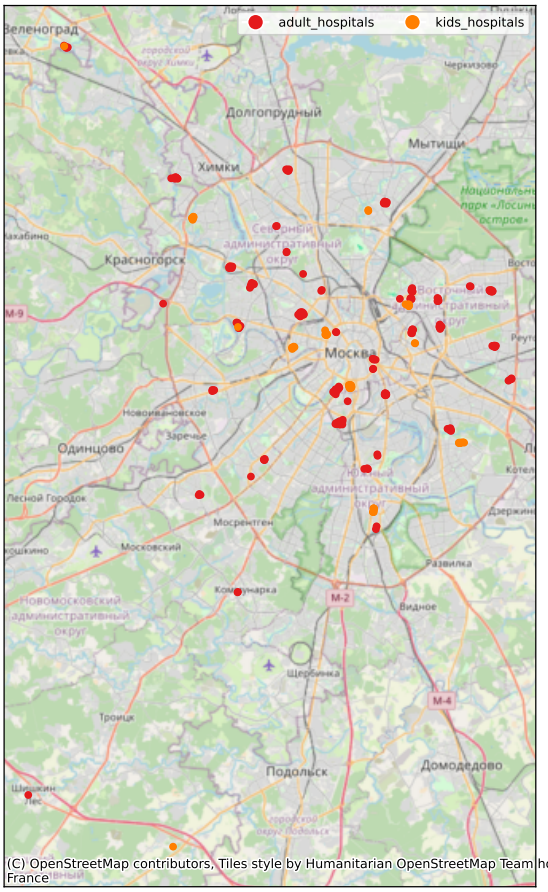

flats
Пушкино
Зеленоград
Долгопрудный
Мытищи
Химки
Красногорск
Одинцово
Люберцы
Балашиха
Москва
Ивантеевка
Черкизово
Дедовск
Нахабино
Восточный
Реутов
Новоивановское
Заречье
Лесной Городок
Мосрентген
Кокошкино
Московский
Развилка
Дзержинский
Лыткарино
Коммунарка
Видное
Щербинка
Троицк
Домодедово
Новомосковский административный округ
городской округ Химки
городской округ Мытищи
Национальный парк «Лосиный остров»
М-9
М-2
М-4
(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap Team hosted by OpenStreetMap France


eco_boiler
eco_factory
eco_waste
eco_dump
eco_thermal
Некрасовский
Поварово
Менделеево
Зеленоград
Лобня
Пушкино
Ивантеевка
Долгопрудный
Мытищи
Щёлково
Химки
Национальный парк «Лосиный остров»
Лосино-Петровский
Монино
Дедовск
Красногорск
Балашиха
Старая Купавна
Реутов
Москва
Электроугли
Новоиваново
Одинцово
Люберцы
Большие Вяземы
Лесной Городок
Дзержинский
Удельная
Красознаменск
Московский
Жуковский
Лыткарино
Видное
Апрелевка
Селятино
Щербинка
Троицк
Подольск
Домодедово
Шишкин Лес
Бронницы
(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap Team hosted by OpenStreetMap France


bus_stops
Солнечногорск
Ашукино
Софрино
Красноармейск
Некрасовский
Поварово
Правдинский
Менделеево
Пушкино
Зеленоград
Черноголовка
Долгопрудный
Щелково
Истра
Дедовск
Лосино-Петровский
Мытищи
Красногорск
Балашиха
Звенигород
Одинцово
Люберцы
Большие Вяземы
Лесной Городок
Удельная
Красноармейск
Жуковский
Апрелевка
Селятино
Видное
Троицк
Белоозёр
Киевский
Щицкин
Подольск
Домодедово
Бронницы
ЛМС
(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap Team hosted by OpenStreetMap France


bus_stations
moscow_stations
train_stations
Зеленоград
Пушкино
Долгопрудный
Мытищи
Щелково
Химки
Красногорск
Балашиха
Одинцово
Люберцы
Жуковский
Раменское
Подольск
Домодедово
(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap Team hosted by OpenStreetMap France


edu_prof
edu_school
Пушкино
Долгопрудный
Мытищи
Химки
Красногорск
Балашиха
Москва
Одинцово
Люберцы
Подольск
Домодедово
(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap Team hosted by OpenStreetMap France


adult_hospitals
kids_hospitals
Зеленоград
Долгопрудный
Мытищи
Химки
Красногорск
Москва
Одинцово
Подольск
Домодедово
Видное
Развилка
Троицк
Щербинка
Новомосковский административный округ
Троицкий административный округ
М-2
М-4
(C) OpenStreetMap contributors, Tiles style by Humanitarian OpenStreetMap Team hosted by OpenStreetMap France

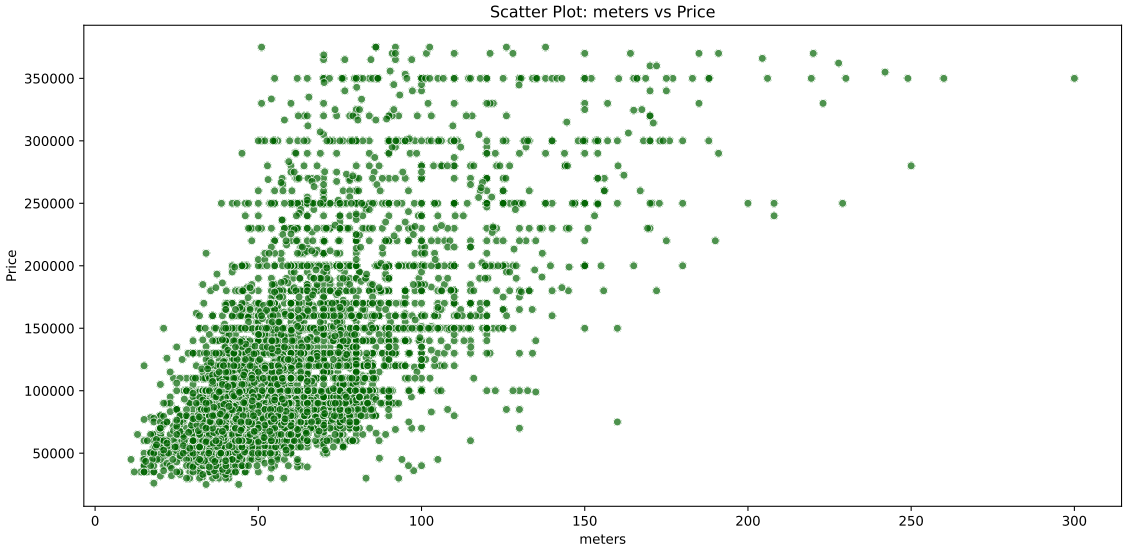
6 EDA, other plots

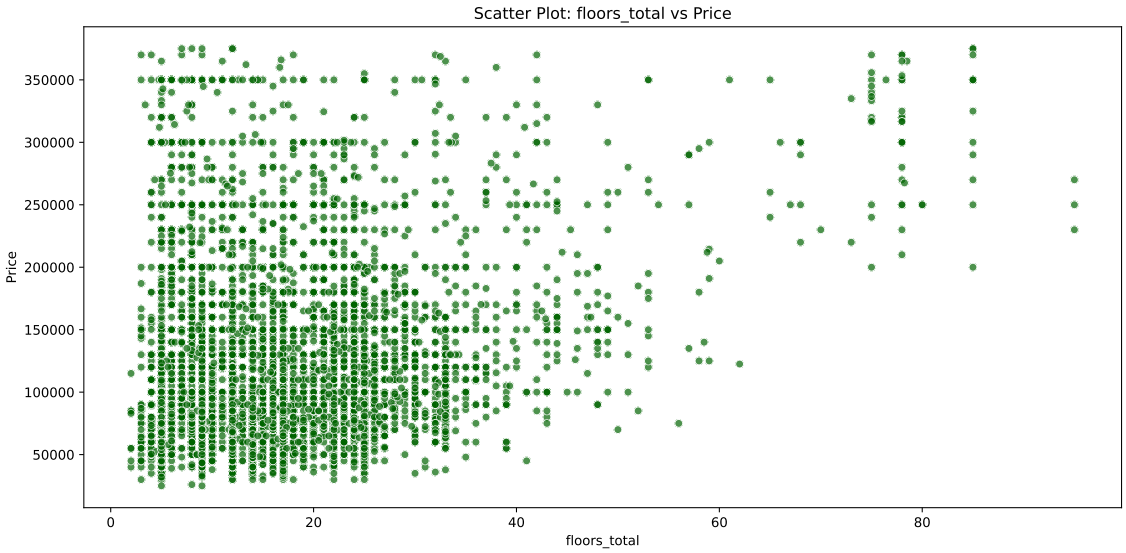
Below is the table with some descriptive statistics for our sample.

	mean	25%	50%	75%	max
price	109846.20	65000.00	85000.00	130000.00	375000.00
floor	8.74	4.00	7.00	12.00	80.00
floors_total	16.73	9.00	15.00	22.00	95.00
meters	57.85	40.00	52.00	68.00	300.00
center_dist	11.17	6.37	10.52	14.29	39.22
adult_hospitals_closest_km	2.55	1.30	2.15	3.37	19.11
bus_stations_closest_km	8.91	5.83	9.29	11.78	24.13
bus_stops_closest_km	0.15	0.08	0.13	0.20	0.85
bus_stops_less500m	13.45	9.00	13.00	17.00	56.00
bus_stops_0.5-1km	36.11	27.00	36.00	45.00	96.00
bus_stops_1-3km	350.32	279.00	356.00	427.00	614.00
bus_stops_3-5km	670.74	524.00	700.00	852.00	1112.00
eco_boiler_closest_km	7.39	3.99	6.94	9.53	34.46
eco_dump_closest_km	11.54	8.77	12.17	15.13	19.17
eco_factory_closest_km	4.12	2.36	3.49	4.77	26.52
eco_thermal_closest_km	2.30	1.22	1.97	3.00	18.43
eco_waste_closest_km	8.34	5.25	8.22	11.01	25.73
edu_prof_closest_km	1.43	0.56	0.96	1.59	10.57
edu_prof_1-3km	4.01	2.00	4.00	6.00	13.00
edu_prof_3-5km	5.53	3.00	5.00	8.00	15.00
edu_school_closest_km	0.23	0.09	0.16	0.28	2.61
edu_school_1-3km	16.64	13.00	16.00	22.00	32.00
edu_school_3-5km	26.72	21.00	29.00	34.00	52.00
kids_hospitals_closest_km	5.06	2.29	4.12	6.31	28.37
moscow_stations_closest_km	0.86	0.48	0.71	1.00	11.23
moscow_stations_less500m	0.39	0.00	0.00	1.00	5.00
moscow_stations_0.5-1km	1.09	0.00	1.00	2.00	10.00
moscow_stations_1-3km	10.32	5.00	8.00	14.00	37.00
moscow_stations_3-5km	19.52	10.00	16.00	28.00	54.00
parks_closest_km	0.30	0.11	0.22	0.39	3.92
parks_less500m	2.34	1.00	2.00	3.00	12.00
parks_0.5-1km	4.64	2.00	4.00	6.00	24.00

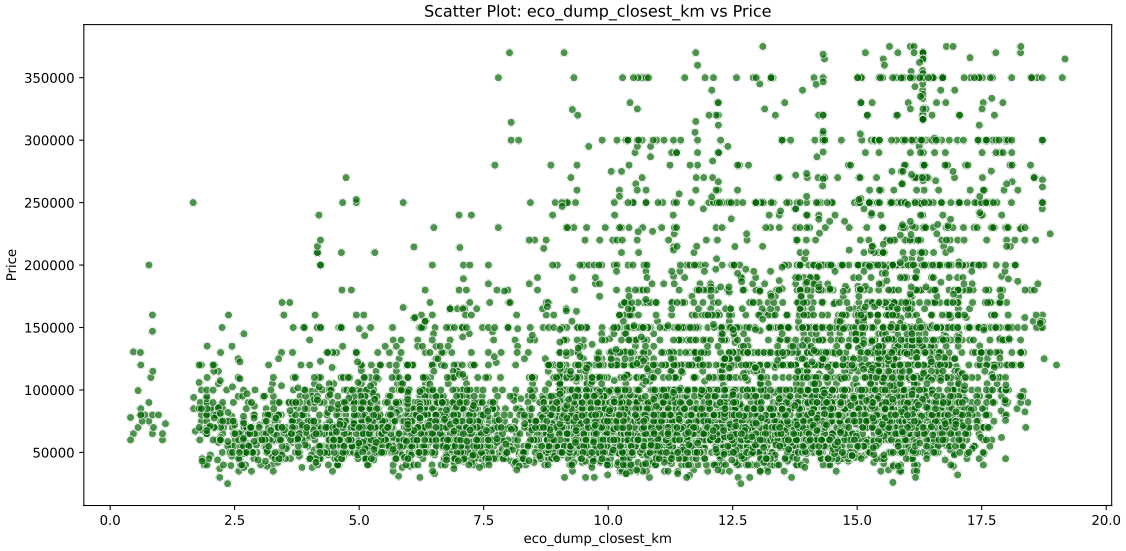
	mean	25%	50%	75%	max
parks_1-3km	39.87	24.00	35.00	46.00	137.00
parks_3-5km	70.27	41.00	63.00	99.00	172.00
parks_area_less500m	1.47	0.03	0.12	0.55	31.35
parks_area_0.5-1km	1.81	0.10	0.28	1.19	31.96
parks_area_1-3km	10.37	3.04	7.90	15.18	40.33
parks_area_3-5km	14.94	7.58	12.21	19.65	52.48
shops_closest_km	0.09	0.03	0.07	0.12	0.95
shops_less500m	73.81	27.00	49.00	87.00	1356.00
shops_0.5-1km	187.45	85.00	143.00	230.00	1819.00
shops_1-3km	1706.74	1057.00	1588.00	2267.00	4990.00
shops_3-5km	3148.32	2023.00	3136.00	4217.00	7463.00
train_stations_closest_km	7.88	2.72	6.83	10.72	35.03
other_flats_less500m	8.73	5.00	8.00	11.00	43.00
other_flats_0.5-1km	17.36	9.00	15.00	23.00	78.00
other_flats_1-3km	159.66	90.00	138.00	207.00	469.00
other_flats_3-5km	288.40	158.00	271.00	398.00	624.00

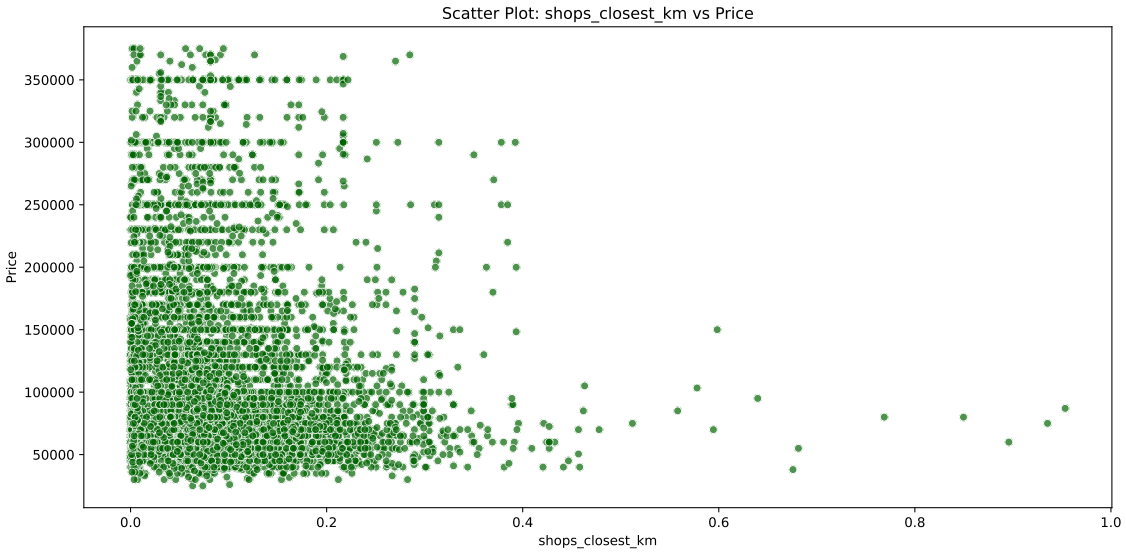
The most noticeable relationship is between size of the apartment and rental price and total floor count in the building (but not always), see the plots below.



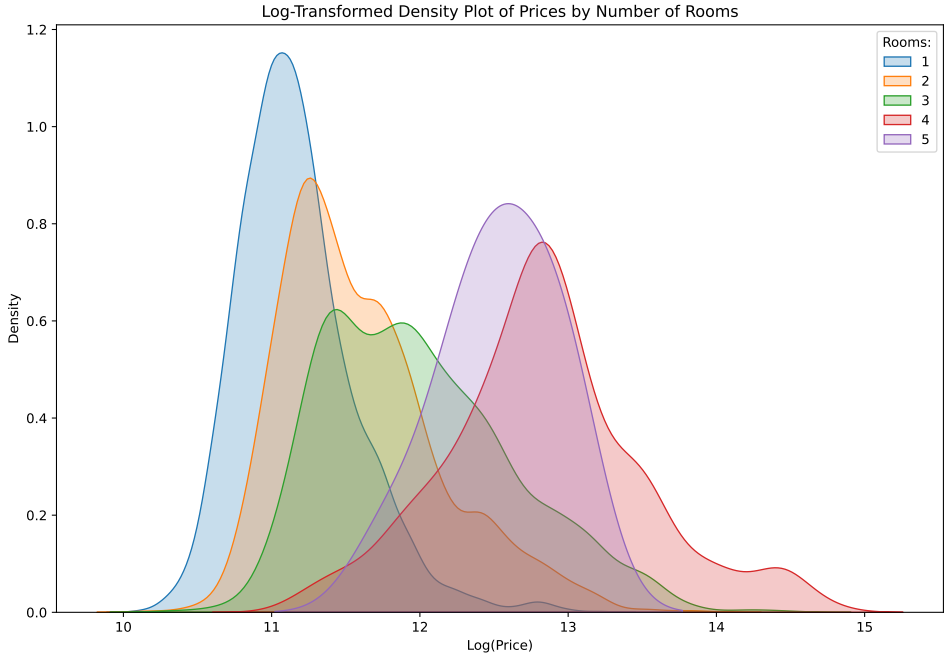


Also we can see that there are no apartments with high rental price which are very close to dumps. Also if apartment is to far from the closest shop, its rental price is never too high.

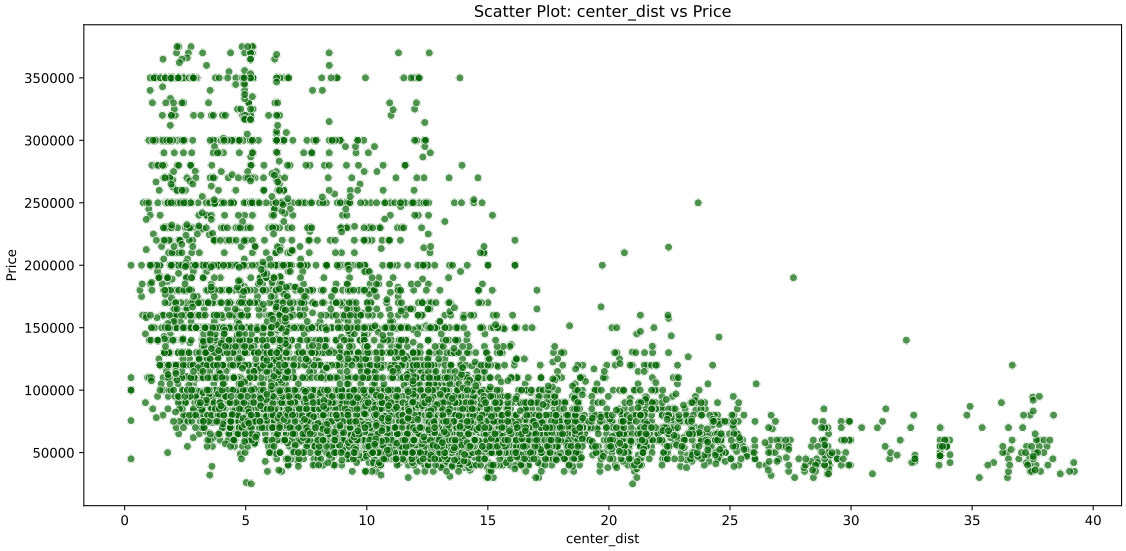




The graph below presents density plots for apartments with different numbers of rooms. As can be seen, on average, a higher number of rooms implies higher rental price, but five-room apartments break this rule. It might be because we have only 10 ads for such apartments in our sample (see the bar chart above).



Notice that there are no high price apartments which are really far from the city center.


Scatter Plot: center_dist vs Price
Price
350000
300000
250000
200000
150000
100000
50000
0
5
10
15
20
25
30
35
40
center_dist

7 Feature selection and transformation + holdout sample

- Features related to distances higher than 5km were removed (we can check map of Moscow and realize that that is too much)
- Using pd.get_dummies() we transformed rooms count to categorical variable
- Random 1000 rows from our dataset were removed and placed into the file holdout.csv

8 ML models preliminary results

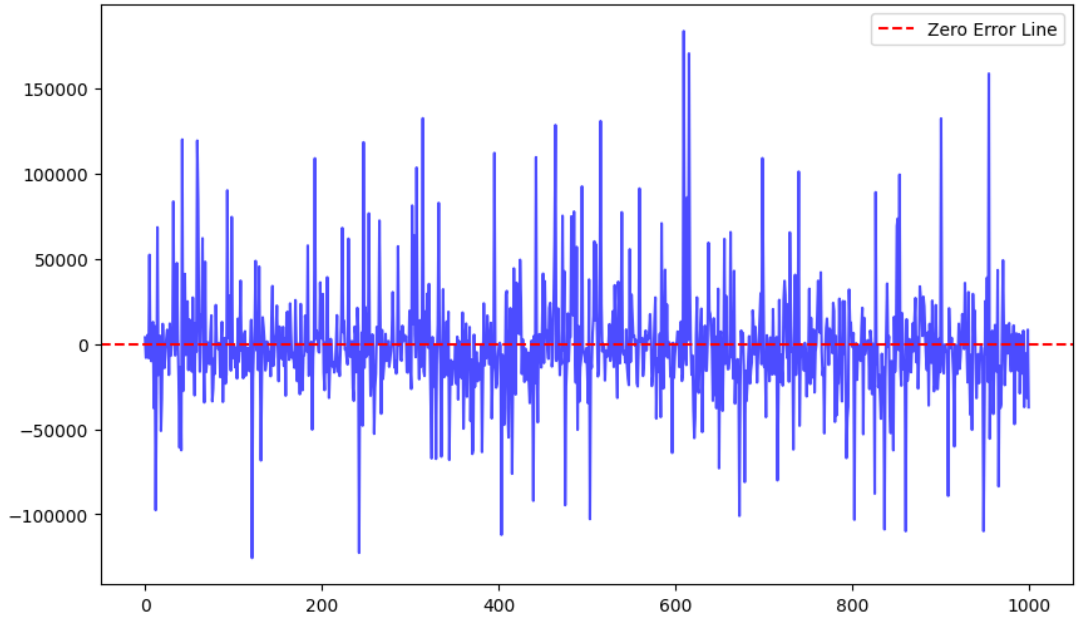
During this stage we found that the gradient boosting algorithm provided too good errors on a holdout sample (in 95% of cases errors were within -30;30 rubles interval). After investigation we discovered that the information leak happened.

Features related to the number of places in which there are rental ads ('other_flats' prefix) had been calculated for the whole sample, and then it was splitted. That means that the training sample 'knows' something from the holdout sample in advance, and that leads to overfit.

To fix that we recalculated these features for the training sample pretending that we do not have a holdout sample. That fixed the issue.

Below are the summary statistics and plot for the errors which are derived from the applying gradient boosting model on our holdout sample. To do preliminary fine tuning, we did grid search and 5 fold cross validation. See the ipynb folder, "models" notebook for the details (it is quite messy though).

```
count      1000.0000
mean      -1430.6160
std       32274.0013
min     -125682.9858
25%      -15859.3409
50%       -3720.1515
75%        8608.7532
max      183819.9949
2.5%     -64589.5754
97.5%     81395.7991
Name: price, dtype: float64
```



9 Code

Code is available at this github repo: https://github.com/Karapsin/ICEF_ML_Project Below is the brief description of its structure.

Folders:

- **data load:** csv files from cian parser
- **ipynb:** some Jupyter notebooks for visualizations
- **json:** for now single json file with info about ecological geo features
- **csv:** different kind of csv
- **py:** python scripts for data load, cleaning and features engineering

Py folder:

- **Data load folder:** various py scripts for data loading (names are pretty self explanatory)
- **Geo_funs.py:** functions for coordinates loading and working with points, multipoints, polygon and multipolygons objects from geopandas package
- **Features_funs.py:** functions used for geo features creation
- **Geo_plot_funs.py:** functions for plotting maps with points and polygons on top
- **Flats_data_final_clean.py:** data cleaning procedure
- **Meta_geo_df.py:** creation of the df for plots with map
- **map_graph.py:** interactive map creation

Csv folder:

- **Csv_to_split folder:** during features creation this csv files are splitted
- **Finished_geo_features folder**: files with calculated geo features
- **Object coords folder:** csv with geo objects coordinates, used to calculate geo features
- **Clean_data.csv:** data after duplicates and missing values removal, with coordinates

- **consolidated_data.csv:** data before duplicates removal, but with duplicates. Composed from csv files from data load folder
- **Data_for_nodelling.csv:** data to be used for ML models fit
- **Data_for_report.csv:** data for this submission
- **Flats_data_coords.csv:** data with coordinates and duplicates, but without missing values
- **Holdout.csv:** holdout sample
- **Manual_coords.csv:** locations for which coordinates had been found manually
- **Meta_geo_df.csv:** file with all geo objects, their coordinates and types, used for map plots
- **Outliers_to_delete.csv:** pretty self explanatory
- **Same_coords.csv:** ads at the same coords and floor as at least 1 another ad