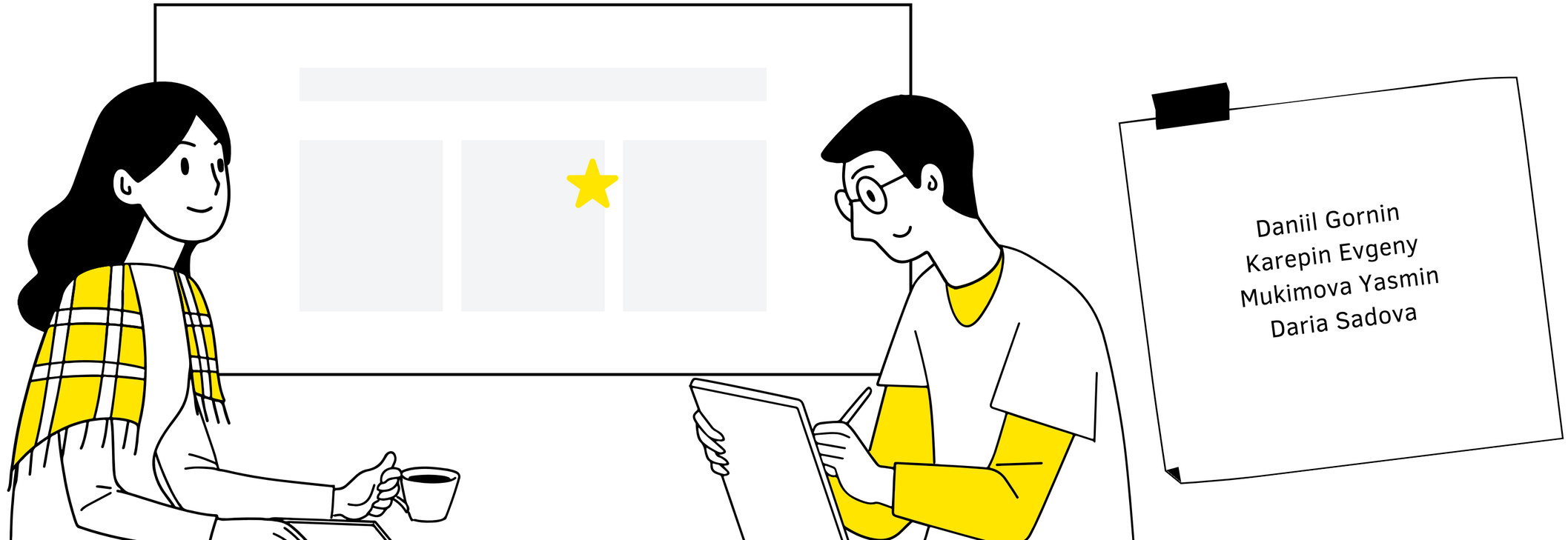
HSE GSB. Apartment price prediction



Daniil Gornin
Karepin Evgeny
Mukimova Yasmin
Daria Sadova

Participants

1. Daniil Gornin (ML Algorithms)
2. Karepin Evgeny (EDA)
3. Mukimova Yasmin (API)
4. Daria Sadova (Data exploration)

Our project

dgornin/
Apartment_price_predic...

HSE GSB. Apartment price prediction

1 Contributor · 0 Issues · 0 Stars · 0 Forks

dgornin/Apartment_price_prediction: HSE GSB. Apartment price prediction

HSE GSB. Apartment price prediction. Contribute to dgornin/Apartment_price_prediction development by creating an account on GitHub.

GitHub

We have APT dataset to predict prices

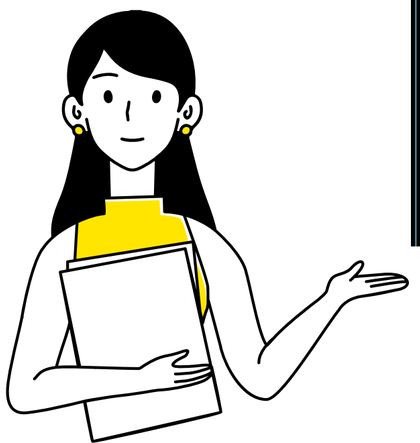
```
df_train = pd.read_csv("./hse-gsb-apartment-price-prediction/train.csv", delimiter=";")
df_test = pd.read_csv("./hse-gsb-apartment-price-prediction/test.csv", delimiter=";")
df_sample_submission = pd.read_csv("./hse-gsb-apartment-price-prediction/sample_submission.csv", delimiter=",")
df_train.head()
```

[2] ✓ 1.1s Python

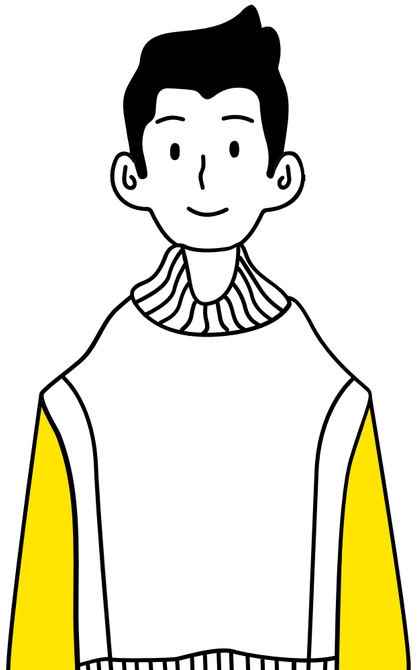
	ID	Категория	Заголовок	Опции продажи	Тип	Общая площадь	Жилая площадь	Площадь кухни	Этаж	Кол-во этажей в доме	...	Кол-во совмещенных санузлов	Ремонт	Вид из окон	Расстояние до метро	
0	16592911	Недвижимость в Москве/ Продажа/ Продажа 2-комнат...	2-комн. квартира, 70.03 м2	Возможна ипотека	Новостройка	70.03	41.8	13.7	2	17	...	NaN	NaN	NaN	12 мин. на транспорте	(Новомосковский),Ще
1	17242255	Недвижимость в Москве/ Продажа/ Продажа 3-комнат...	3-комн. квартира, 76.47 м2	Возможна ипотека	Новостройка	76.47	43.4	11.2	11	17	...	NaN	NaN	Во двор	12 мин. на транспорте	(Новомосковский),Ще
2	193433104	Недвижимость в Москве/ Продажа/ Продажа 2-комнат...	2-комн. квартира, 60.0 м2	NaN	Новостройка	60.00	38.0	12.0	5	20	...	NaN	NaN	На улицу и двор	3 мин. на транспорте	Москва,СЗАО,райс Мневники
3	140334219	Недвижимость в Москве/ Продажа/ Продажа 2-комнат...	2-комн. квартира, 65.2 м2	Возможна ипотека	Новостройка	65.20	0.0	0.0	2	5	...	NaN	NaN	На улицу и двор	NaN	Московская область горо
4	189844059	Недвижимость в Москве/ Продажа/ Продажа 1-комнат...	1-комн. квартира, 38.18 м2	Возможна ипотека	Новостройка	38.18	17.3	8.1	1	3	...	NaN	NaN	Во двор	33 мин. на транспорте	Московская с городской

5 rows × 30 columns

+ Code + Markdown



We have a lot of addresses what o do with them?



We have 1776 unique address, let's find coordinates of them

> Note: if you want to parse data about coordinates by yourself then you will need to use `Yandex API` to use it create `API key` here crate api and then pass it to `Client function`

> Note: if you want you can use already parsed data from json file here go to cell

Load data by your self

```python
from yandex_geocoder import Client

adr_to_cord = {}
regions_cord = {}
t_adr_to_cord = {}
t_regions_cord = {}
client = Client("Yur API key")
```

08]

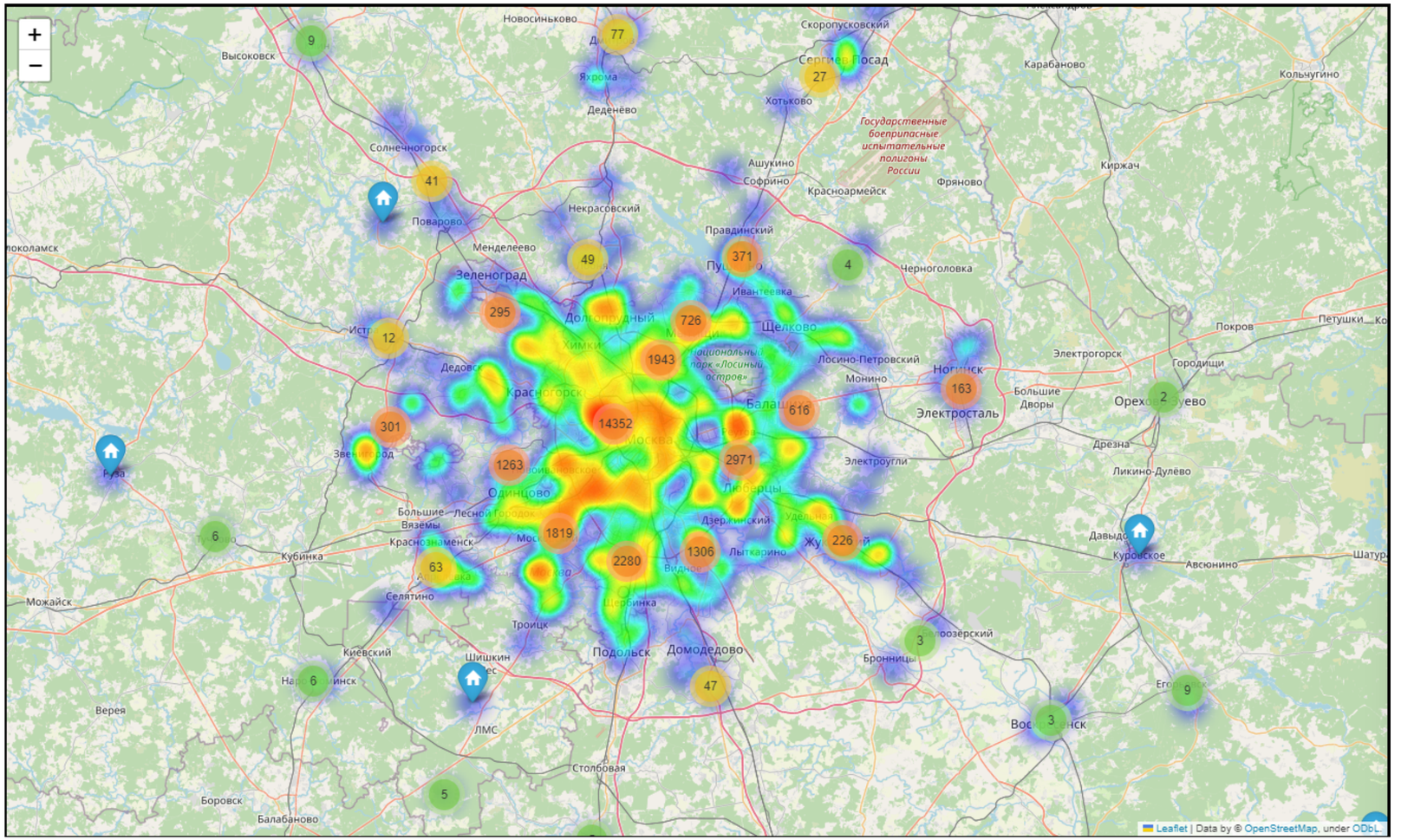
Python

```python
for i in range(0, len(train_address)):
    coordinates = client.coordinates(train_address[i])
    if coordinates:
        adr_to_cord[train_address[i]] = {"latitude": float(coordinates[1]), "longitude": float(coordinates[0])}
    else:
        adr_to_cord[train_address[i]] = {"latitude": None, "longitude": None}
```

]

Python

How data distributed?



We can notice that all apts are placed in Moscow, or near it, so for the model we can add range to Moscow centre, and another metric as range to the local centre

Add distance

```python
def haversine_np(lon1, lat1, lon2, lat2):
    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = np.sin(dlat/2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2.0)**2
    c = 2 * np.arcsin(np.sqrt(a))
    km = 6367 * c
    return km
```

0] ✓ 0.0s Python

```python
to_Moscow = []
to_LocalCentr = []
for index, row in df_train.iterrows():
    to_Moscow.append(haversine_np(row['longitude'], row['latitude'], 37.6156, 55.7522))
    to_LocalCentr.append(haversine_np(row['longitude'], row['latitude'], regions_cord[row['Регион']]['longitude'], regions_cord[row['Регион']]['latitude']))

df_train['to_Moscow'] = to_Moscow
df_train['to_LocalCentr'] = to_LocalCentr
```

1] ✓ 1.0s Python

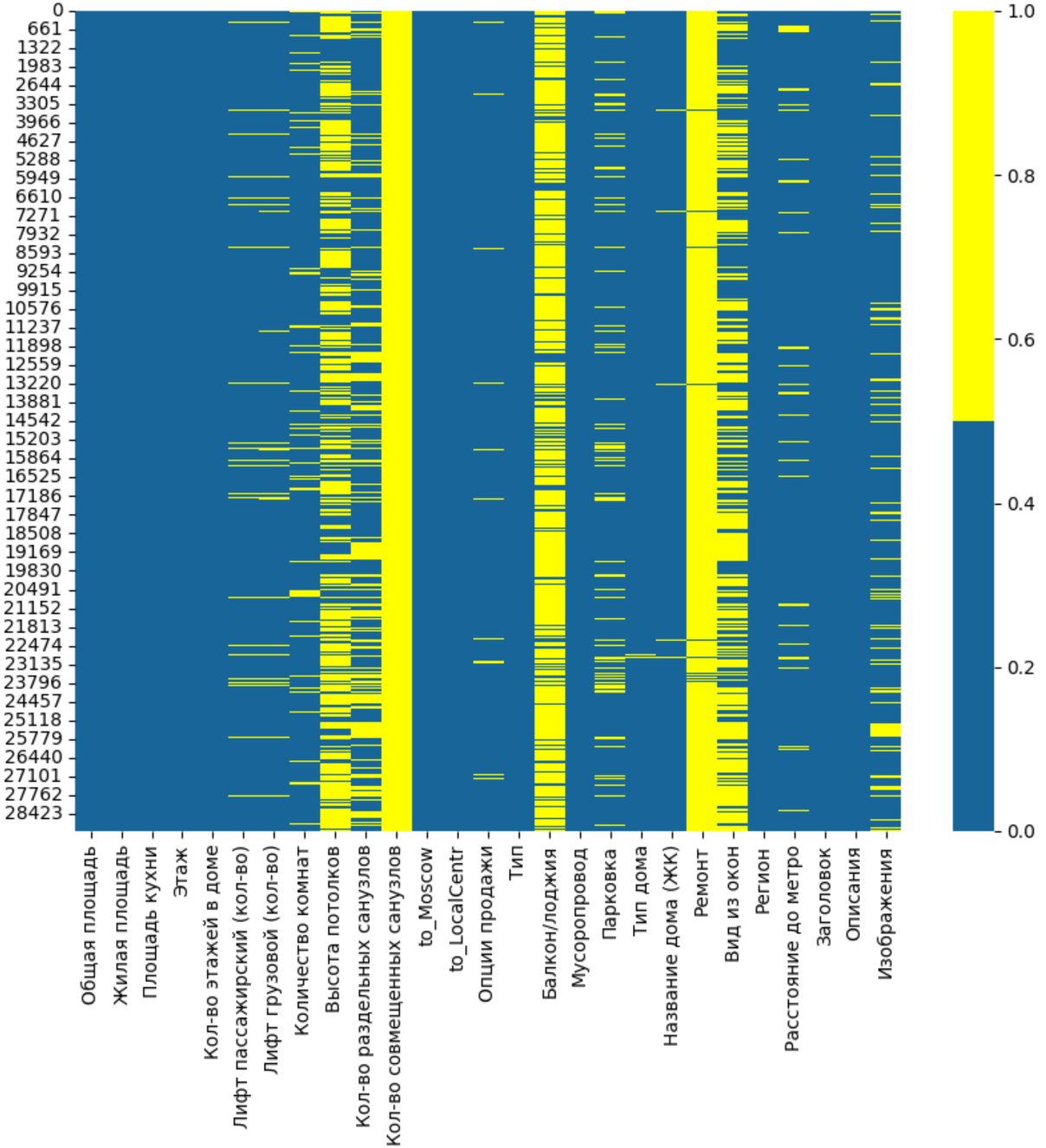
```python
to_Moscow = []
to_LocalCentr = []
for index, row in df_test.iterrows():
    to_Moscow.append(haversine_np(row['longitude'], row['latitude'], 37.6156, 55.7522))
    to_LocalCentr.append(haversine_np(row['longitude'], row['latitude'], t_regions_cord[row['Регион']]['longitude'], t_regions_cord[row['Регион']]['latitude']))

df_test['to_Moscow'] = to_Moscow
df_test['to_LocalCentr'] = to_LocalCentr
```

2] ✓ 0.2s Python

Data exploration

Identify which parameters are posable helpful for the model:

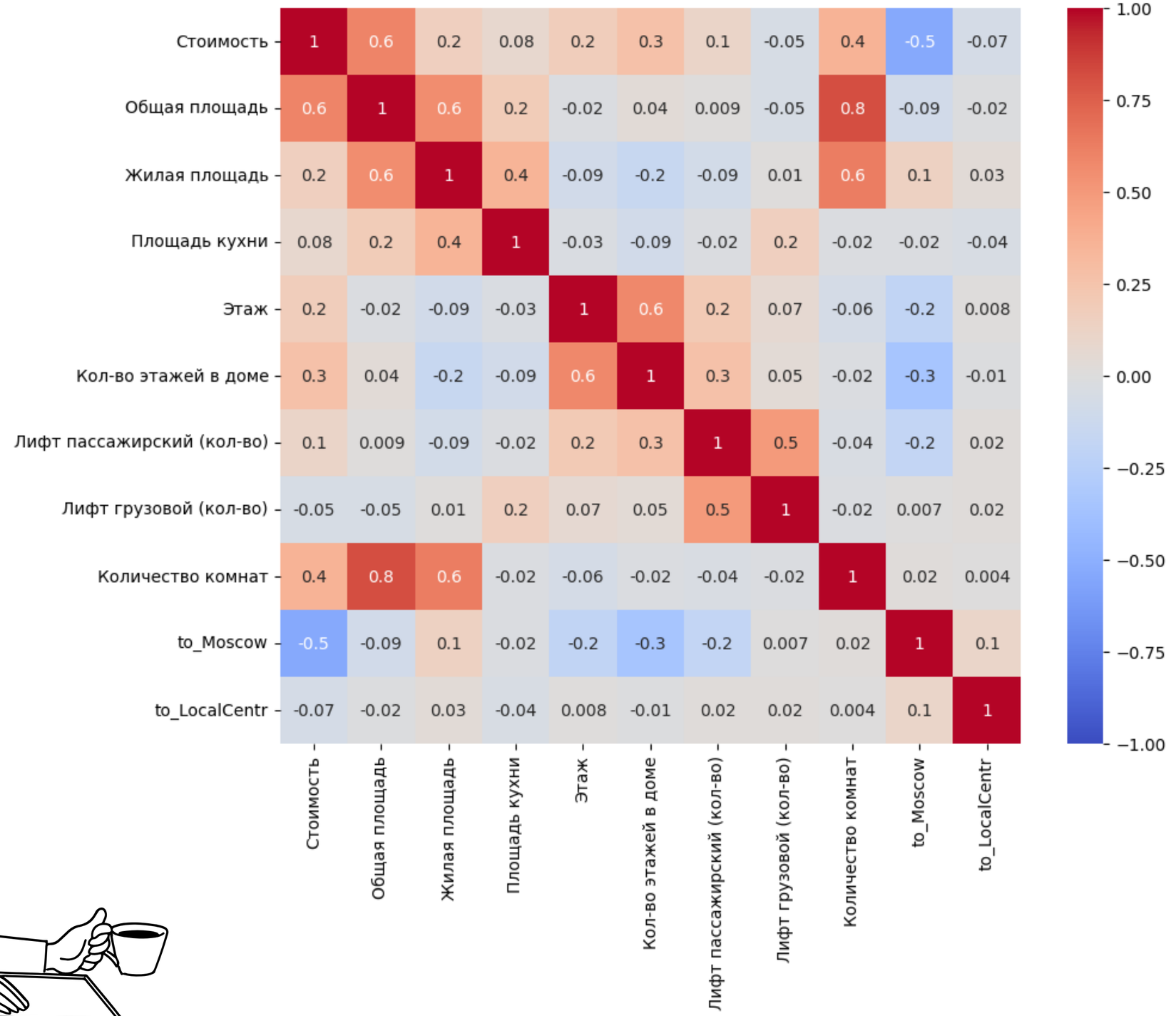
- Target
  - 'Стоимость'
- Continuous
  - 'Общая площадь'
  - 'Жилая площадь'
  - 'Площадь кухни'
  - 'Этаж'
  - 'Кол-во этажей в доме'
  - 'Лифт пассажирский (кол-во)'
  - 'Лифт грузовой (кол-во)'
  - 'Количество комнат'
  - 'Высота потолков'
  - 'Кол-во раздельных санузлов'
  - 'Кол-во совмещенных санузлов'
  - 'to_Moscow'
  - 'to_LocalCentr'
- Categorical
  - 'Опции продажи'
  - 'Тип'
  - 'Балкон/лоджия'
  - 'Мусоропровод'
  - 'Парковка'
  - 'Тип дома'
  - 'Название дома (ЖК)'
  - 'Ремонт'
  - 'Вид из окон'
  - 'Регион'
- For revue and posable changes
  - 'Расстояние до метро'
  - 'Заголовок'
  - 'Описания'
  - 'Изображения'



What to do?

We can see a lot of missing data in columns: 'Высота потолков', 'Кол-во раздельных санузлов', 'Кол-во совмещенных санузлов', 'Балкон/лоджия', 'Ремонт', 'Вид из окон'

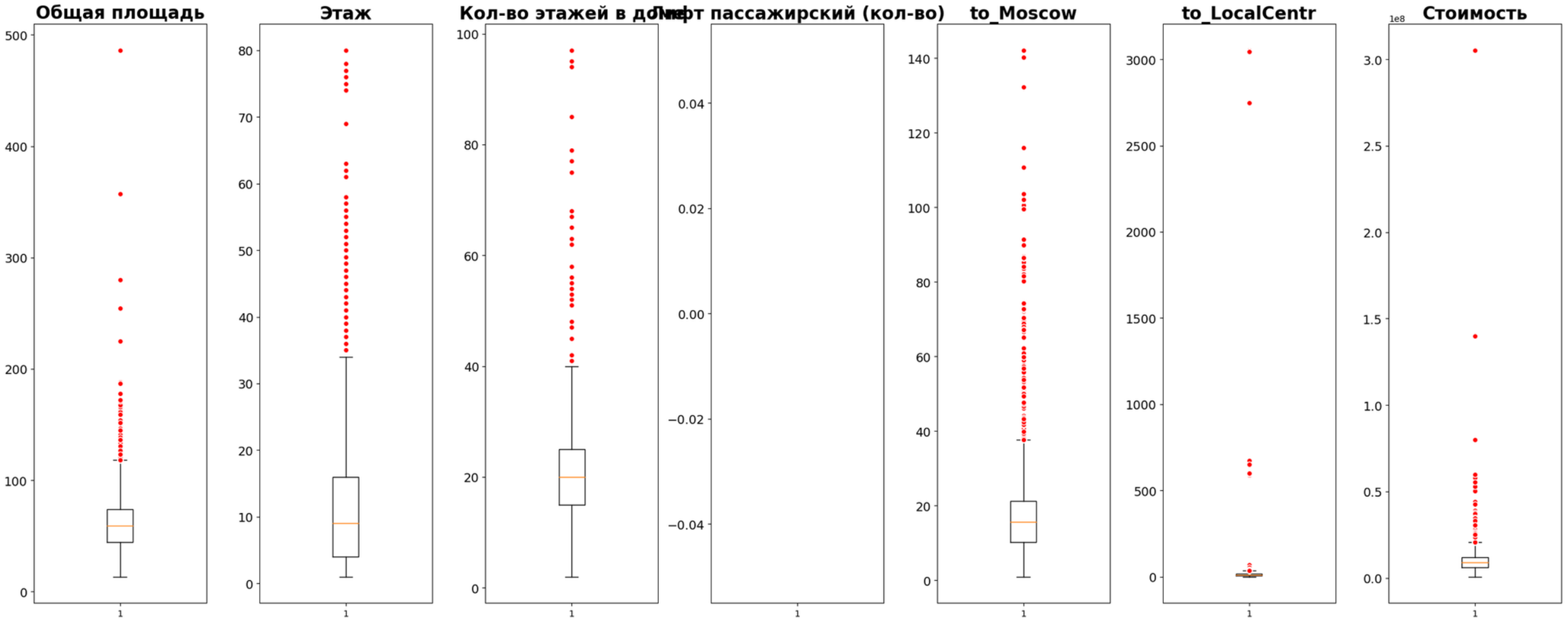
We should exclude them from our model because this parameters will not give us relative the information

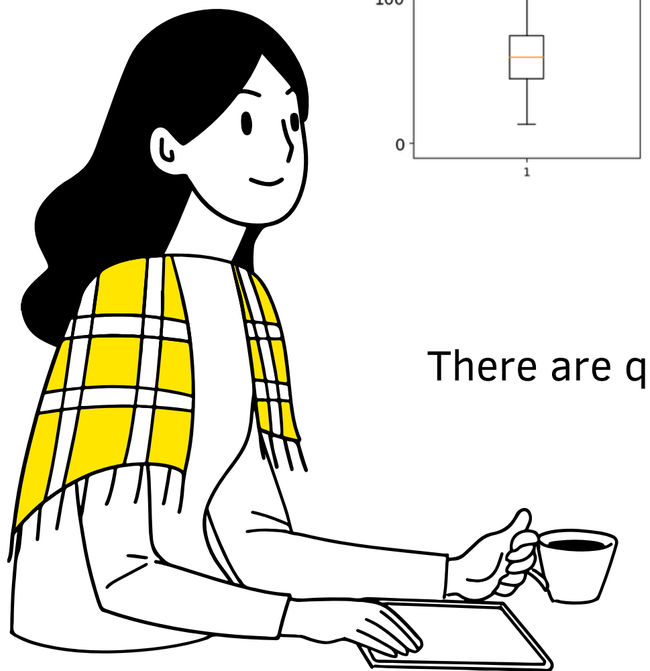

Стоимость 1 0.6 0.2 0.08 0.2 0.3 0.1 -0.05 0.4 -0.5 -0.07
Общая площадь 0.6 1 0.6 0.2 -0.02 0.04 0.009 -0.05 0.8 -0.09 -0.02
Жилая площадь 0.2 0.6 1 0.4 -0.09 -0.2 -0.09 0.01 0.6 0.1 0.03
Площадь кухни 0.08 0.2 0.4 1 -0.03 -0.09 -0.02 0.2 -0.02 -0.02 -0.04
Этаж 0.2 -0.02 -0.09 -0.03 1 0.6 0.2 0.07 -0.06 -0.2 0.008
Кол-во этажей в доме 0.3 0.04 -0.2 -0.09 0.6 1 0.3 0.05 -0.02 -0.3 -0.01
Лифт пассажирский (кол-во) 0.1 0.009 -0.09 -0.02 0.2 0.3 1 0.5 -0.04 -0.2 0.02
Лифт грузовой (кол-во) -0.05 -0.05 0.01 0.2 0.07 0.05 0.5 1 -0.02 0.007 0.02
Количество комнат 0.4 0.8 0.6 -0.02 -0.06 -0.02 -0.04 -0.02 1 0.02 0.004
to_Moscow -0.5 -0.09 0.1 -0.02 -0.2 -0.3 -0.2 0.007 0.02 1 0.1
to_LocalCentr -0.07 -0.02 0.03 -0.04 0.008 -0.01 0.02 0.02 0.004 0.1 1
Стоимость
Общая площадь
Жилая площадь
Площадь кухни
Этаж
Кол-во этажей в доме
Лифт пассажирский (кол-во)
Лифт грузовой (кол-во)
Количество комнат
to_Moscow
to_LocalCentr
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

What to do?

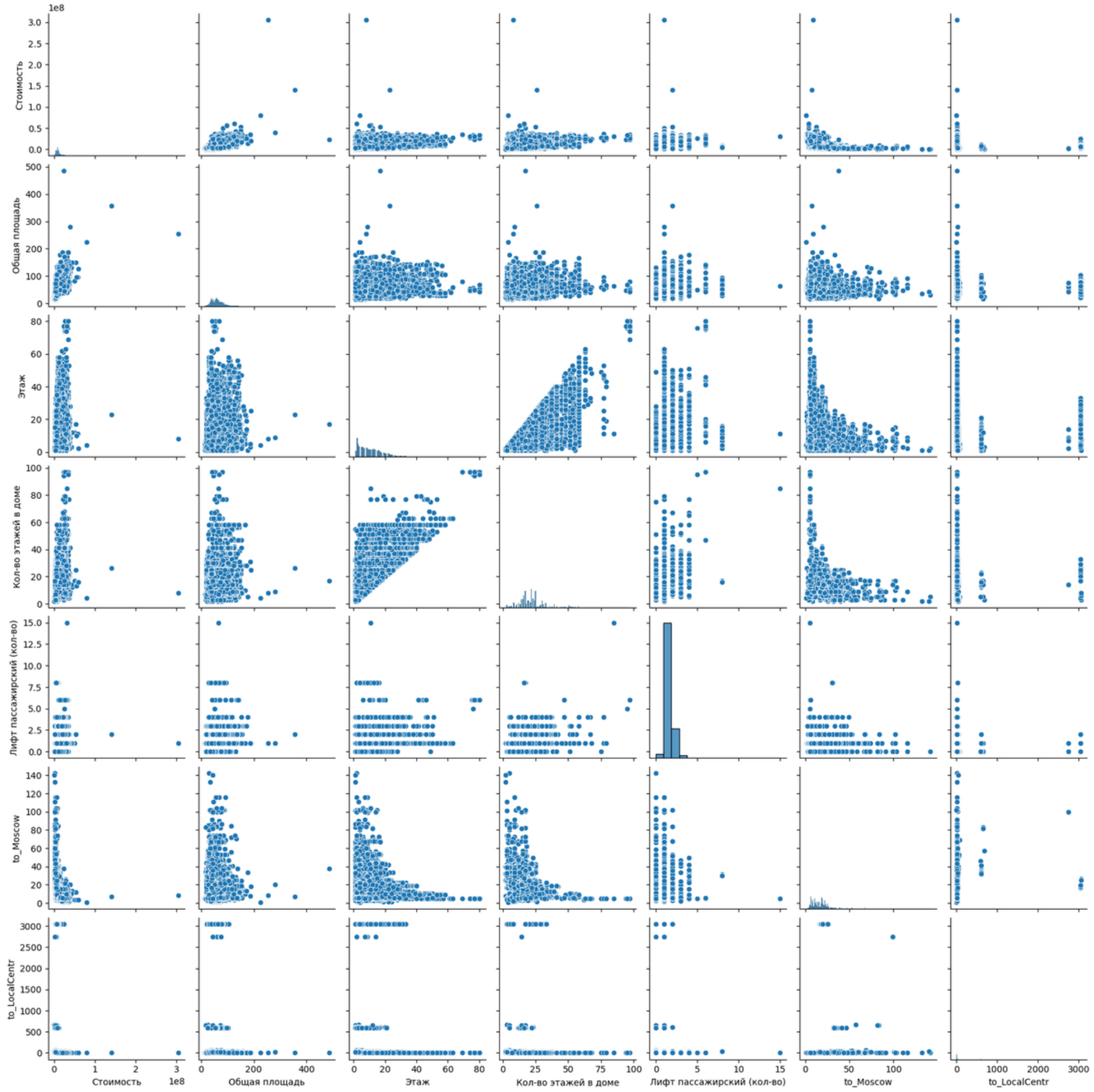
You can notice that all the parameters depending on the area will cause multicollinearity, so you will have to leave only one of them in the model, the total area correlates with the price best of all, then we subtract it and throw it out 'Жилая площадь', 'Площадь кухни', 'Количество комнат'

Also there is multicollinearity in amount of different types of elevator so instead of different types our command used sum of types of elevator, but the result of the correlation was lower than just passenger elevator, so i lefted just amount of passenger elevators





There are quite a lot of outliers in the data, but they cannot be deleted due to the specifics of the data



linear dependence is observed for the area and the distance to Moscow to the price

Cope with null data

+ Code + Markdown

[23]
```
df_train[continuous + target].isnull().sum()
```
✓ 0.0s Python

```
Общая площадь                  0
Этаж                           0
Кол-во этажей в доме           0
Лифт пассажирский (кол-во)    1096
to_Moscow                      0
to_LocalCentr                  0
Стоимость                      0
dtype: int64
```

[24]
```
df_train['Лифт пассажирский (кол-во)'].mean()
```
✓ 0.0s Python

```
1.2066337483898668
```

[25]
```
df_train['Лифт пассажирский (кол-во)'] = df_train['Лифт пассажирский (кол-во)'].fillna(1)
```
✓ 0.0s Python

[26]
```
df_test['Лифт пассажирский (кол-во)'] = df_test['Лифт пассажирский (кол-во)'].fillna(1)
```
✓ 0.0s Python

Analysis of categorical data

```
[27]  df_train[categorical].describe()
      ✓ 0.0s                                                    Python
```

	Опции продажи	Тип	Мусоропровод	Парковка	Тип дома	Название дома (ЖК)	Регион
count	28584	29044	29044	25806	28827	28841	29044
unique	7	2	2	3	5	580	57
top	Долевое участие (214-ФЗ), Возможна ипотека	Новостройка	Нет	подземная	Монолитный	ЖК «Хорошевский»	Москва
freq	22208	28655	28884	12427	25788	1158	20462

Column Регион is already partly used in range to Moscow centr column, and it have to many unique values it is better to do not encode it and use it

Column Название дома (ЖК) have to many unique it is inefficient to code it

For other we will use One-hot encoding for higher efficiency

```
[28]  categorical.remove('Название дома (ЖК)')
      categorical.remove('Регион')
      ✓ 0.0s                                                    Python
```

```
[29]  df_train_dummies = pd.get_dummies(df_train[categorical])
      df_train_dummies.head()
      ✓ 0.0s                                                    Python
```

	Опции продажи_Альтернатива	Опции продажи_Альтернатива, Возможна ипотека	Опции продажи_Возможна ипотека	Опции продажи_Долевое участие (214-ФЗ)	Опции продажи_Долевое участие (214-ФЗ), Возможна ипотека	Опции продажи_Свободная продажа	Опции продажи_Свободная продажа, Возможна ипотека	Тип_Вторичка	Тип_Новост
0	False	False	True	False	False	False	False	False	
1	False	False	True	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	
3	False	False	True	False	False	False	False	False	
4	False	False	True	False	False	False	False	False	

Let's implement ML to solve the challenge

Gradient boosting

Gradient boosting

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

X = df_train[continuous].join(df_train_dummies)
y = df_train['Стоимость']
X_test_submit = df_test[continuous].join(df_test_dummies)
scaler = StandardScaler()
X[continuous] = scaler.fit_transform(X[continuous])
scaler = StandardScaler()
X_test_submit[continuous] = scaler.fit_transform(X_test_submit[continuous])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

[36] ✓ 0.3s Python

```
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV

xgb = XGBRegressor(n_estimators=100000, learning_rate=0.001, tree_method='gpu_hist', eval_metric='rmsle')
xgb.fit(X_train, y_train)
predictions = xgb.predict(X_test)
```

[37] ✓ 3m 28.7s Python

```
from sklearn.metrics import mean_squared_log_error
from math import sqrt

sqrt(mean_squared_log_error(y_test, predictions))
```

[38] ✓ 0.0s Python

```
0.08267688900709971
```

Linear regression model

Linear regression model

```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import GridSearchCV

pipeline = Pipeline(
    [
        ('ln', LinearRegression()),
    ]
)
```

[40] ✓ 0.1s Python

```python
pipeline.fit(X_train, y_train)
```

[41] ✓ 0.0s Python

```
Pipeline
LinearRegression
```

```python
from sklearn.metrics import mean_squared_log_error
from math import sqrt


res = pipeline.predict(X_test)
a = res.mean()
for i in range(len(res)):
    if res[i] < 0:
        res[i] = a

sqrt(mean_squared_log_error(y_test, predictions))
```

[42] ✓ 0.0s Python

```
0.08267688900709971
```

Submit results

Submit results

Best result with gradient boosting so let's upload it

```
predictions_submit = xgb.predict(X_test_submit)
df_sample_submission['Стоимость'] = predictions_submit
df_sample_submission.to_csv("result.csv", index=False)
```

[44] ✓ 0.6s Python

```
! kaggle competitions submit -c hse-gsb-apartment-price-prediction -f result.csv -m "result"
```

[45] ✓ 3.4s Python

```
Successfully submitted to HSE GSB. Apartment price prediction

  0%|          | 0.00/173k [00:00<?, ?B/s]
100%|██████████| 173k/173k [00:00<00:00, 1.51MB/s]
100%|██████████| 173k/173k [00:01<00:00, 116kB/s]
```

Results

Leaderboard

Raw Data | Refresh

YOUR RECENT SUBMISSION

result.csv
Submitted by dgornin · Submitted an hour ago

Score: 0.26475
Private score:

Jump to your leaderboard position

Search leaderboard

Public | Private

This leaderboard is calculated with approximately 50% of the test data. The final results will be based on the other 50%, so the final standings may be different.

#	Team	Members	Score	Entries	Last	Solution
1	foldl#103688		0.21577	10	1d	
2	**dgornin**		0.24168	6	10h	
3	Nikita Jeanpaul		0.41048	20	10h	
4	Максим Репин		0.42061	3	1d	
5	Error 418		0.43267	3	9h	

Your Best Entry!
Your submission scored 0.26475, which is not an improvement of your previous score. Keep trying!

Thanks for your attention!