Государственное бюджетное профессиональное
образовательное учреждение Московской области
«Физико-технический колледж»

Аналитический отчет

Модель оценки квартиры на рынке продажи в Москве, Новой Москве и Московской области

Работу выполнил:
Студент группы ИСП-21
Мисюк Вячеслав

Долгопрудный, 2024

Введение: данный аналитический отчет посвящен исследованию факторов, влияющих на стоимость квартир. Для проведения анализа была разработана модель на языке программирования Python с использованием библиотек Pandas и Matplotlib. Модель позволила проанализировать обширный набор данных о квартирах, выявив ключевые факторы, определяющие их стоимость.

Цель: Целью данного аналитического отчета является изучение и выявление ключевых факторов, влияющих на стоимость квартир, с использованием модели, разработанной на языке программирования Python с библиотеками Pandas и Matplotlib.

Задачи:

1) Используя открытые источники, такие как Циан, проанализировать и создать базу данных с характеристиками квартир

2) Очистить лишние и «грязные» данные, количество которых недостаточно для анализа, либо они являются не нужными

3) Визуализировать готовые данные в виде гистограмм и графиков

Основная часть

В качестве отправной точки для нашего анализа была выбрана платформа Циан - один из крупнейших онлайн-сервисов по поиску недвижимости в России. Изначально, для сбора данных о квартирах, необходимых для анализа, было решено применить метод парсинга. Парсинг - это процесс автоматического извлечения данных с веб-сайтов. Для этого был разработан специальный скрипт на языке Python, который загружал информацию о квартирах, доступную на Циан.

Для парсинга Циана была использована библиотека CianParser

Код, который был использован для парсинга данных:

```
import cianparser

moscow_parser = cianparser.CianParser(location="Москва")
data = moscow_parser.get_flats(deal_type="sale", rooms=1, with_saving_csv=True, with_extra_data=True, additional_settings={"start_page":1, "end_page":54})

print(data[0])
```

В результате парсинга был готов файл размером в ~9500 строк, где содержались такие значения, как: общая площадь, цена, этаж, район, метро и другие.

Теперь перейдем к python:

Для начала работы нам необходимо импортировать нужные библиотеки; импортируем numpy, pandas, matplotlib, seaborn:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Загружаем файл в программу и выводим первые пять строк, чтобы убедиться, что всё загрузилось и работает верно:

Загружаем файл

```
df = pd.read_csv('main_v2.csv') # Читаем файл
df.head()
```

Python

	author	author_type	url	location	deal_type	accommodation_type	floor	floors_count	rooms_count	total_meters	...	heating_type	finish_type	living_meters	kitchen_meters	phone	district	street	house_number	underground	residential_complex
0	Метражи group	real_estate_agent	https://www.cian.ru/sale/flat/308167237/	Москва	sale	flat	5.0	7.0	1.0	34.6	...	-1.0	-1	18	8	7.985041e+10	Северное Измайлово	15-я Парковая	54	Щёлковская	NaN
1	Alliance Agency Real Estate	real_estate_agent	https://www.cian.ru/sale/flat/298254403/	Москва	sale	flat	4.0	33.0	1.0	42.9	...	-1.0	-1	23	15	7.965188e+10	Останкинский	Годовикова	11к2	Алексеевская	iLove
2	Зиля Карамова	realtor	https://www.cian.ru/sale/flat/263316279/	Москва	sale	flat	1.0	16.0	1.0	37.7	...	-1.0	-1	-1	-1	7.916094e+10	Чертаново Центральное	Варшавское шоссе	142К2	Пражская	NaN
3	Омега-Эстейт	real_estate_agent	https://www.cian.ru/sale/flat/303983520/	Москва	sale	flat	13.0	17.0	1.0	37.6	...	-1.0	-1	18	15	7.985243e+10	NaN	Бачуринская	7к2	Бачуринская	Новая Звезда
4	Dream Realty	real_estate_agent	https://www.cian.ru/sale/flat/307043068/	Москва	sale	flat	9.0	28.0	1.0	34.7	...	-1.0	-1	9	14	7.915350e+10	Богородское	Открытое шоссе	14Д	Бульвар Рокоссовского	Талисман на Рокоссовского

5 rows × 24 columns

В результате парсинга было добыто много лишней информации, такие как номер дома, имя автора, ссылка на объявление, номер телефона…

Такие данные нужно удалить и удаляем колонки с этими лишними данными, а также сразу отсортируем наш файл, чтобы в нем остались здания только 1950 года постройки и младше

Удаляем лишние колонки

```
df.drop(['author'], axis=1,inplace=True) # Имя владельца не интересно
df.drop(['deal_type'], axis=1,inplace=True) # Тип сделки нам не важен
df.drop(['accommodation_type'], axis=1,inplace=True) # Тип здания нам не важен
df.drop(['phone'], axis=1,inplace=True) # Номер телефона нам тоже не нужен
df.drop(['house_number'], axis=1,inplace=True) # Номер дома не нужен
df.drop(['heating_type'], axis=1,inplace=True) # Тип отопления не нужен
df.drop(['object_type'], axis=1,inplace=True) # Тип дома не нужен
df.drop(['street'], axis=1,inplace=True) # Улица не нужна, т.к. важна станция метро и район
df.drop(['residential_complex'], axis=1,inplace=True) # Жилой комплекс не важен
df.drop(['url'], axis=1,inplace=True) # Ссылка на объявление не нужна
df = df[df['year_of_construction'] >= 1955] # Убираем старинные здания
```

В нашем файле много неизвестных данных, которые не удалось запарсить и они помечены как «-1» в ячейках, они нам будут мешать, поэтому заменяем все возможные «-1» ячейки на Nan-значения и проверяем, всё ли правильно:

Заполнем пропуски

```
df = df.replace(-1, np.nan) # Заполняем пустые ячейки Nan
df = df.replace('-1', np.nan) # Заполняем пустые ячейки Nan
df = df.replace(-1.0, np.nan) # Заполняем пустые ячейки Nan
df = df.replace('-1.0', np.nan) # Заполняем пустые ячейки Nan
```

```
df.head() # После удаления проверяем целостность информации
```

	author_type	location	floor	floors_count	rooms_count	total_meters	price	year_of_construction	house_material_type	finish_type	living_meters	kitchen_meters	district	underground
0	real_estate_agent	Москва	5.0	7.0	1.0	34.6	9000000.0	1978	NaN	NaN	18.0	8.0	Северное Измайлово	Щёлковская
2	realtor	Москва	1.0	16.0	1.0	37.7	9450000.0	1982	NaN	NaN	NaN	NaN	Чертаново Центральное	Пражская
3	real_estate_agent	Москва	13.0	17.0	1.0	37.6	11000000.0	2018	NaN	NaN	18.0	15.0	NaN	Бачуринская
4	real_estate_agent	Москва	9.0	28.0	1.0	34.7	14500000.0	2023	NaN	NaN	9.0	14.0	Богородское	Бульвар Рокоссовского
6	real_estate_agent	Москва	5.0	9.0	1.0	34.1	10990000.0	1970	NaN	NaN	17.0	8.0	Чертаново Центральное	Пражская

В файле есть возможность образования дубликатов, поэтому по хорошему их надо удалить, что мы и делаем:

Проверяем и удаляем дубликаты

```
num_duplicates = df.duplicated().sum()
print("Кол-во дубликатов:", num_duplicates)
```

```
Кол-во дубликатов: 36
```

```
df.drop_duplicates()
print(f'Датафрейм после удаления дубликатов имеет {df.shape[0]} строк ')
```

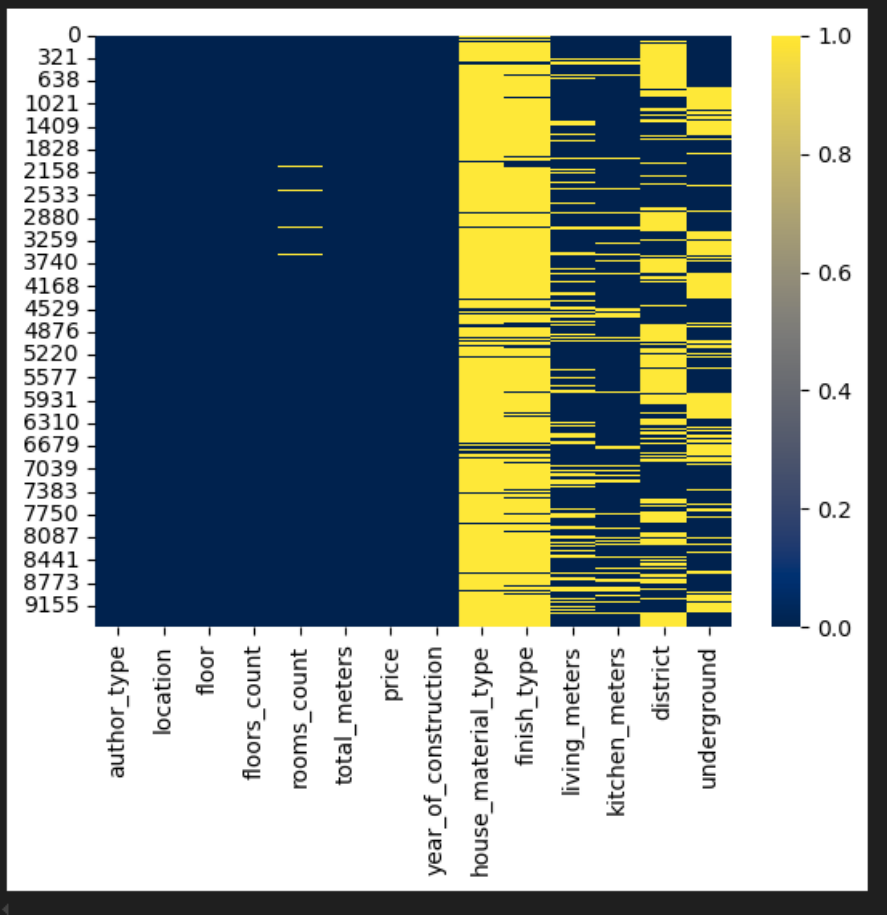
```
Датафрейм после удаления дубликатов имеет 6981 строк
```

Далее выводим тепловую карту с пропусками в файле:

Проверяем пропуски

```
sns.heatmap(df.isnull(), cmap='cividis')
```

```
<Axes: >
```



Изобразим эту карту в виде строк, чтобы пустые данные было легче подсчитать:

Пустые колонки в процентах

```
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))
```

```
author_type - 0%
location - 0%
floor - 0%
floors_count - 0%
rooms_count - 1%
total_meters - 0%
price - 0%
year_of_construction - 0%
house_material_type - 94%
finish_type - 90%
living_meters - 22%
kitchen_meters - 11%
district - 45%
underground - 34%
```

По результатам этого анализа пустых колонок мы узнали, что колонки с типом материала дома (house_material_type) и финишной отделкой дома (finish_type) являются практически пустыми, т. к. их пустота оценивается в 94% и 90% соответственно. Поэтому их надо удалить:

Удаление пустых колонок

```
df.drop(['house_material_type'], axis=1,inplace=True) # Удаляем материал дома, т.к. недостаточно данных
df.drop(['finish_type'], axis=1,inplace=True) # Удаляем финишную отделку квартиры, т.к. недостаточно данных
df.head(5)
```

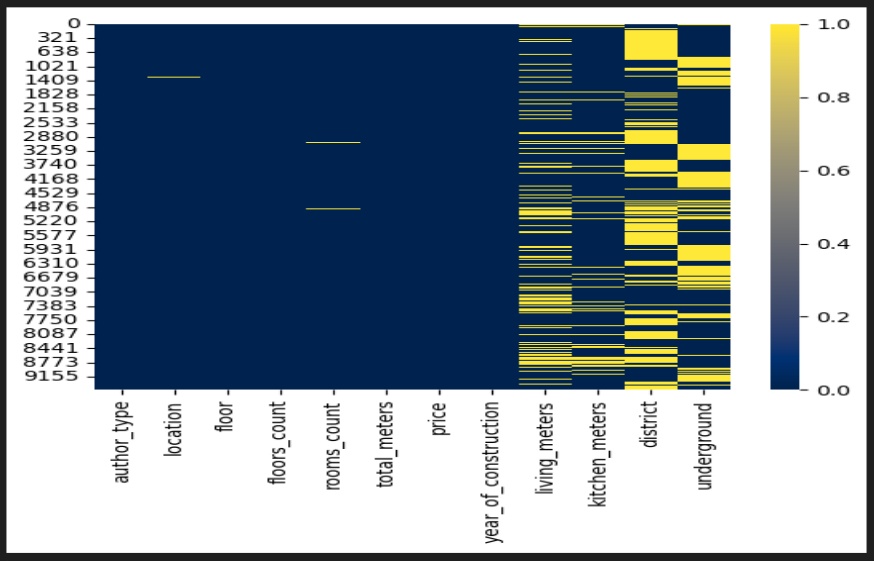
	author_type	location	floor	floors_count	rooms_count	total_meters	price	year_of_construction	living_meters	kitchen_meters	district	underground
0	real_estate_agent	Москва	5.0	7.0	1.0	34.6	9000000.0	1978	18.0	8.0	Северное Измайлово	Щёлковская
2	realtor	Москва	1.0	16.0	1.0	37.7	9450000.0	1982	NaN	NaN	Чертаново Центральное	Пражская
3	real_estate_agent	Москва	13.0	17.0	1.0	37.6	11000000.0	2018	18.0	15.0	NaN	Бачуринская
4	real_estate_agent	Москва	9.0	28.0	1.0	34.7	14500000.0	2023	9.0	14.0	Богородское	Бульвар Рокоссовского
6	real_estate_agent	Москва	5.0	9.0	1.0	34.1	10990000.0	1970	17.0	8.0	Чертаново Центральное	Пражская

Проверяем тепловую карту пропусков и понимаем, что колонка «district» так же является практически пустой и ее надо удалить, но удалять эту колонку я не решил, т.к. эту колонку я использовал в построении графика

Вторая проверка на пустые значения

```
sns.heatmap(df.isnull(), cmap='cividis')
```

```
<Axes: >
```




```
for col in df.columns:
    pct_missing = np.mean(df[col].isnull())
    print('{} - {}%'.format(col, round(pct_missing*100)))
```

```
author_type - 0%
location - 0%
floor - 0%
floors_count - 0%
rooms_count - 1%
total_meters - 0%
price - 0%
year_of_construction - 0%
living_meters - 22%
kitchen_meters - 11%
district - 45%
underground - 34%
```

Для дальнейшей работы нам понадобится колонка, которая показывает цену квадратного метра каждой квартиры. Вычисляется она таким способом: цена квартиры/общий метраж квартиры.

Создаем колонку с ценой квадратного метра каждой квартиры

```
df['price_per_sqm'] = df['price']/df['total_meters'].astype(int) # Создаем колонку с ценой квадратного метра
```

В корреляционной карте понадобятся данные не равные типу «object», поэтому с помощью библиотеки sklearn я закодировал весь файл

Кодируем наши object данные

```
from sklearn import preprocessing
def df_decode(df_encode):
    result = df_encode.copy()
    encoders = {}
    for column in result.columns:
        if result.dtypes[column] == object:
            encoders[column] = preprocessing.LabelEncoder()
            result[column] = encoders[column].fit_transform(result[column])
    return result, encoders
encoded_df, encoders = df_decode(df)
encoded_df.head()
```

	author_type	location	floor	floors_count	rooms_count	total_meters	price	year_of_construction	living_meters	kitchen_meters	district	underground	price_per_sqm
0	3	32	5.0	7.0	1.0	34.6	9000000.0	1978	18.0	8.0	160	318	264705.882353
2	4	32	1.0	16.0	1.0	37.7	9450000.0	1982	NaN	NaN	203	214	255405.405405
3	3	32	13.0	17.0	1.0	37.6	11000000.0	2018	18.0	15.0	301	19	297297.297297
4	3	32	9.0	28.0	1.0	34.7	14500000.0	2023	9.0	14.0	16	35	426470.588235
6	3	32	5.0	9.0	1.0	34.1	10990000.0	1970	17.0	8.0	203	214	323235.294118

Теперь перейдем к графикам. Первый график, который я вывел — зависимость средней цены квадратного метра от года постройки дома, где находится эта квартира

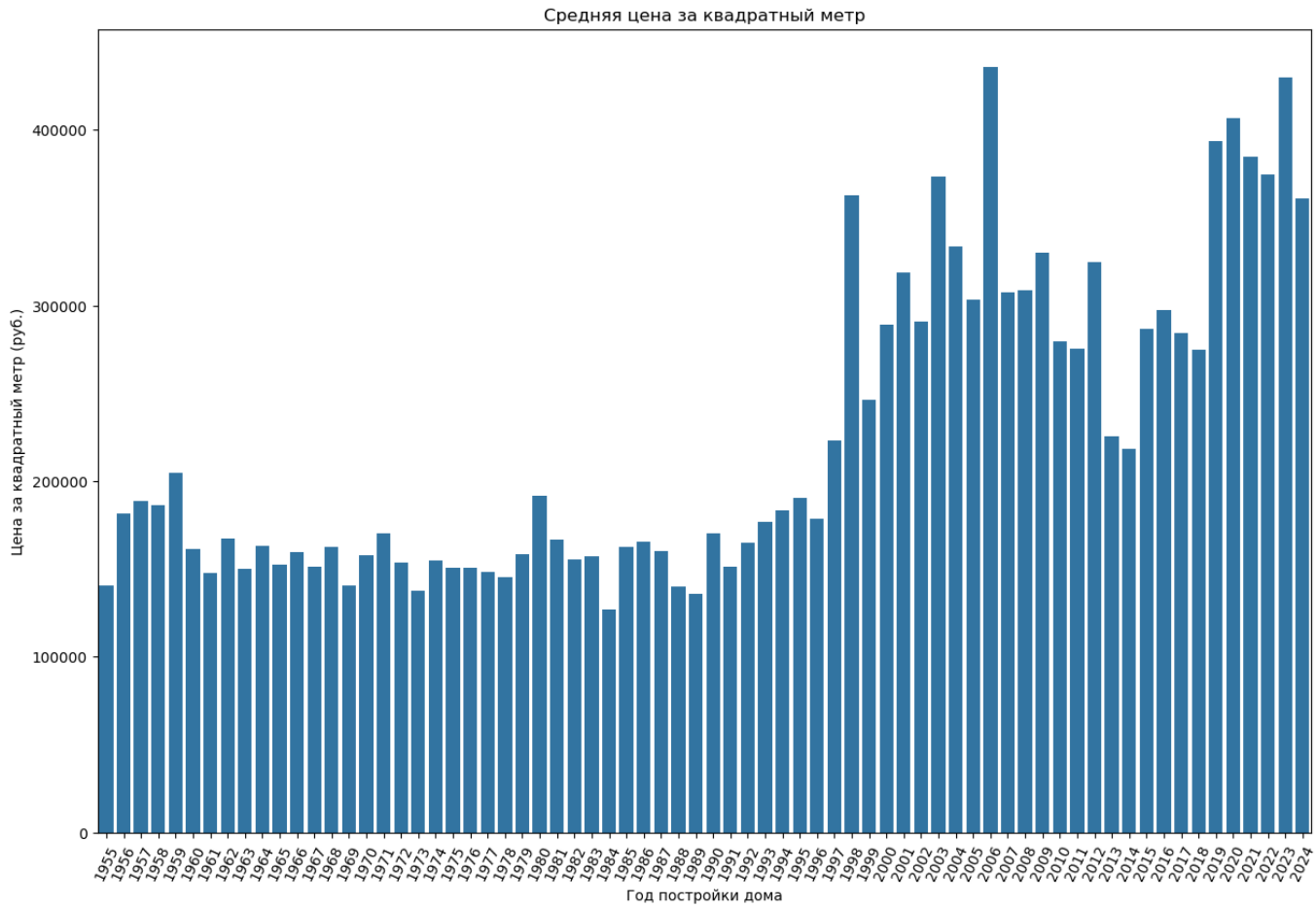
Код к этому графику:

График зависимости цены квадратного метра от года постройки дома

```
average_prices = df.groupby('year_of_construction')['price_per_sqm'].mean().reset_index().astype(int) # Группируем год постройки дома и среднюю цену квадратного метра в году
```

```
plt.figure(figsize=(15, 10))
sns.barplot(x='year_of_construction', y='price_per_sqm', data=average_prices)
plt.title('Средняя цена за квадратный метр')
plt.xlabel('Год постройки дома')
plt.ylabel('Цена за квадратный метр (руб.)')
plt.xticks(rotation=65)
plt.show()
```

Результат этого кода и сам график:



Глядя на график можно понять, что новостройки, построенные за последние 5 лет являются самыми дорогими при счете на квадратных метрах, а самые дешевые это старые здания до ~1995 года.

Далее я построил график зависимости цены квадратного метра от количества комнат в квартире

Код к этому графике:

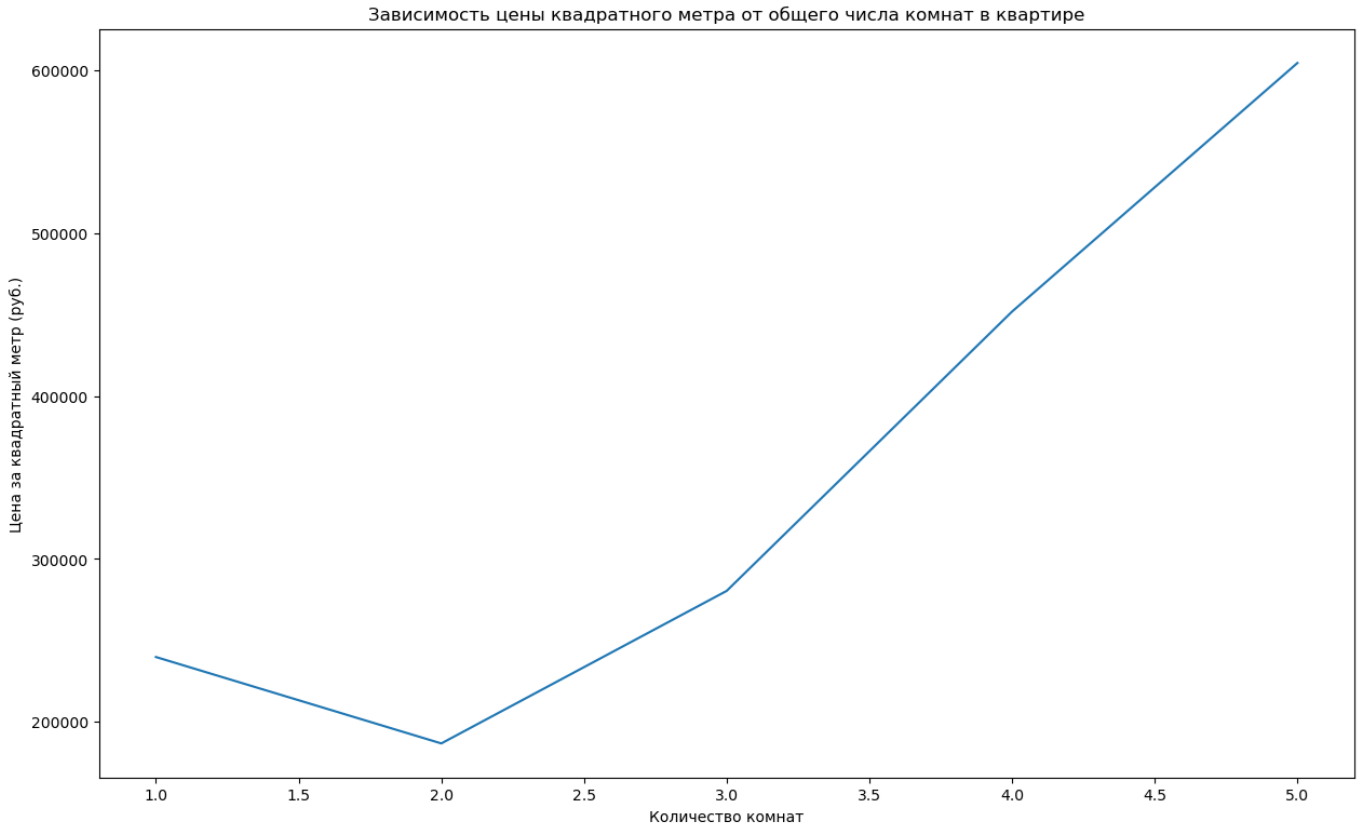
График зависимости цены квадратного метра от количества комнат в квартире

```
rooms_metr = round(df.groupby('rooms_count')['price_per_sqm'].mean().reset_index(), 0).astype(int).sort_values(by='price_per_sqm')
rooms_metr
```

	rooms_count	price_per_sqm
1	2	186732
0	1	239773
2	3	280361
3	4	451794
4	5	604393

```
plt.figure(figsize=(15, 9))
sns.lineplot(data=rooms_metr, x='rooms_count', y='price_per_sqm')
plt.title('Зависимость цены квадратного метра от общего числа комнат в квартире')
plt.xlabel('Количество комнат')
plt.ylabel('Цена за квадратный метр (руб.)')
plt.xticks(rotation=0)
plt.show()
```

Результат этого кода и сам график:



Результатом этого графика является линия, показывающая зависимость средней цены квадратного метра от количества комнат в квартирах. Квартиры с 1 и 2

комнатами являются достаточно дешевыми, а квартиры с 4 и 5 комнатами являются достаточно дорогими в сравнении с 1 и 2 комнатами.

Далее я построил график зависимости средней цены квадратного метра от города, где расположены эти квартиры.

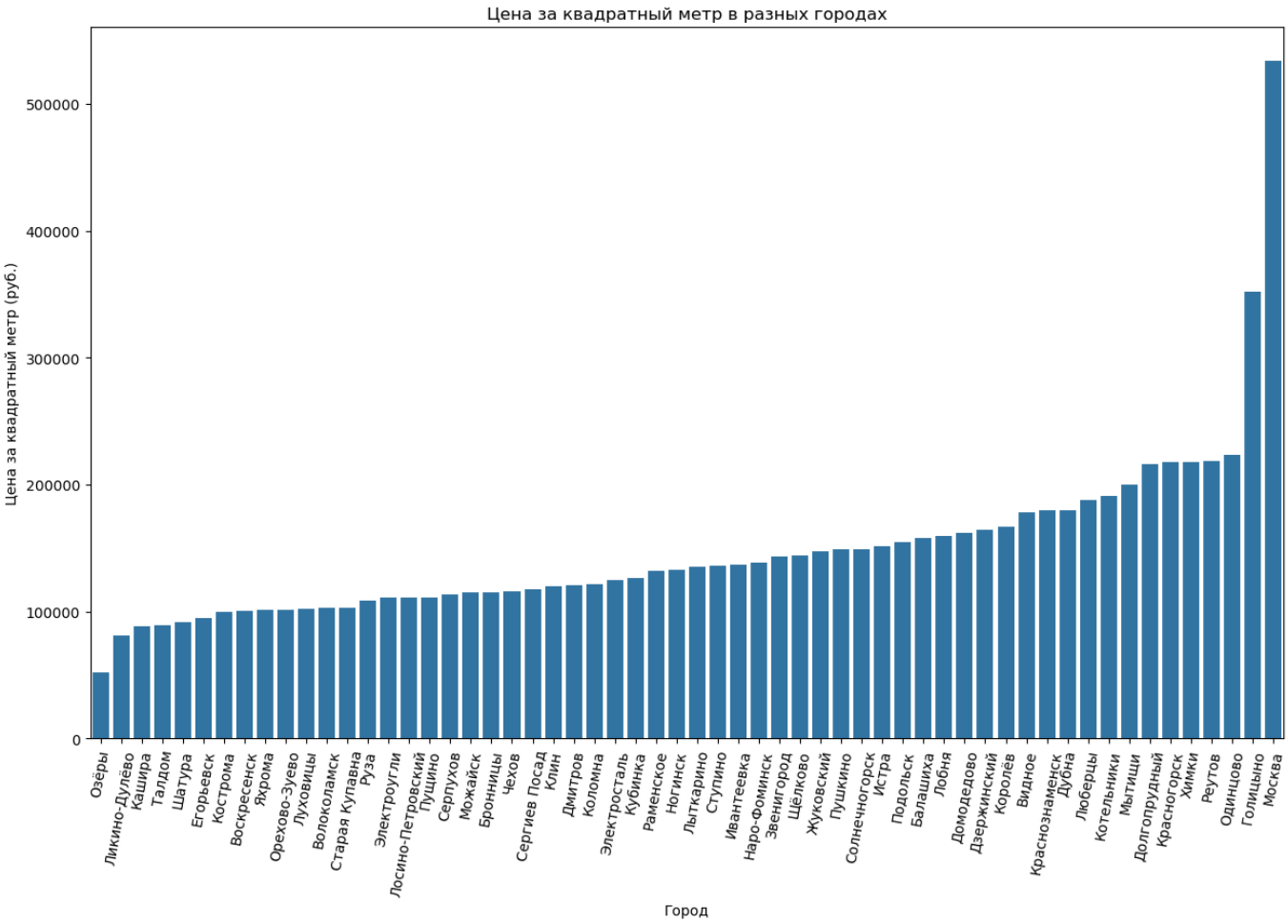
Код к этому графику:

График зависимости цены квадратного метра от города

```
city_pricemetr = round(df.groupby('location')['price_per_sqm'].mean().reset_index(), 0).sort_values(by='price_per_sqm')
# Группируем город и среднюю цену квадратного метра в городе, а так же сортируем по возрастанию
```

```
plt.figure(figsize=(15, 9))
sns.barplot(data=city_pricemetr, x='location', y='price_per_sqm')
plt.title('Цена за квадратный метр в разных городах')
plt.xlabel('Город')
plt.ylabel('Цена за квадратный метр (руб.)')
plt.xticks(rotation=80)
plt.show()
```

Результат этого кода и сам график:



На графике отчетливо видно, что в таких городах как Москва и Голицыно средняя цена за квадратный метр является самой большой, а в городах Озёры, Ликино-Дулёво, Кашира и других цена квадратного метра маленькая и эти города отлично подходят к покупке квартиры, если у покупателя не так много денег.

Далее я построил график зависимости средней цены квадратного метра от района Москвы.

Код к этому графику:

График зависимости цены квадратного метра от района Москвы

```
moscow_df = df[df['location'] == 'Москва']

# Группируем по району и находим среднюю цену квадратного метра
grouped_df = round(moscow_df.groupby('district')['price_per_sqm'].mean().reset_index(), 0).sort_values(by='price_per_sqm')
```

```
plt.figure(figsize=(20, 9))
sns.barplot(data=grouped_df, x='district', y='price_per_sqm')
plt.title('Цена за квадратный метр каждого района Москвы')
plt.xlabel('Район')
plt.ylabel('Цена за квадратный метр (руб.)')
plt.xticks(rotation=90)
plt.ticklabel_format(style='plain', axis='y')
plt.show()
```

Результат этого кода и сам график:

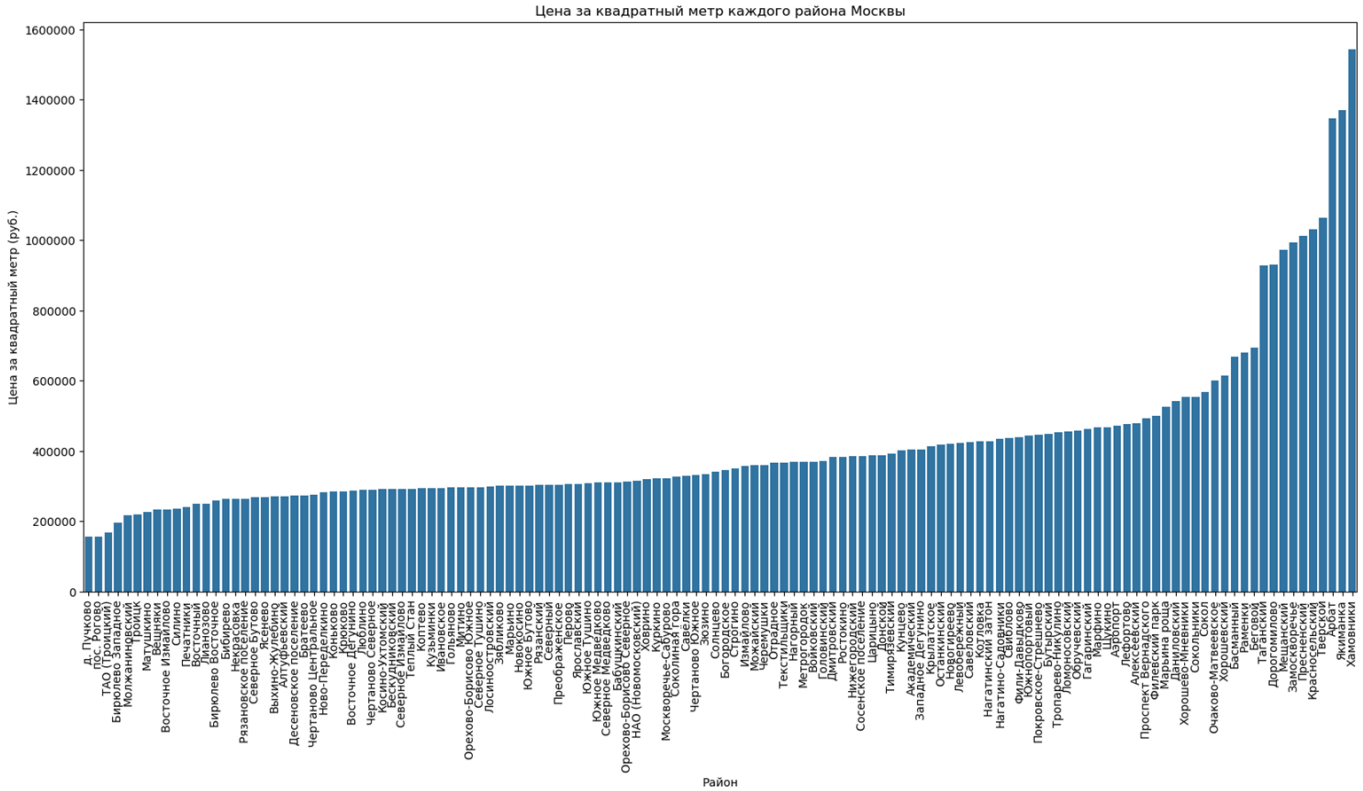


График показывает в каких районах Москвы самые дорогие цены на квадратный метр: это Хамовники, Якиманка и Арбат. А самые дешевые квадратные метры в деревне Пучково и поселке Рогово и т. д. Некоторые значения не входят в районы Москвы, но являются частью Новой Москвы.

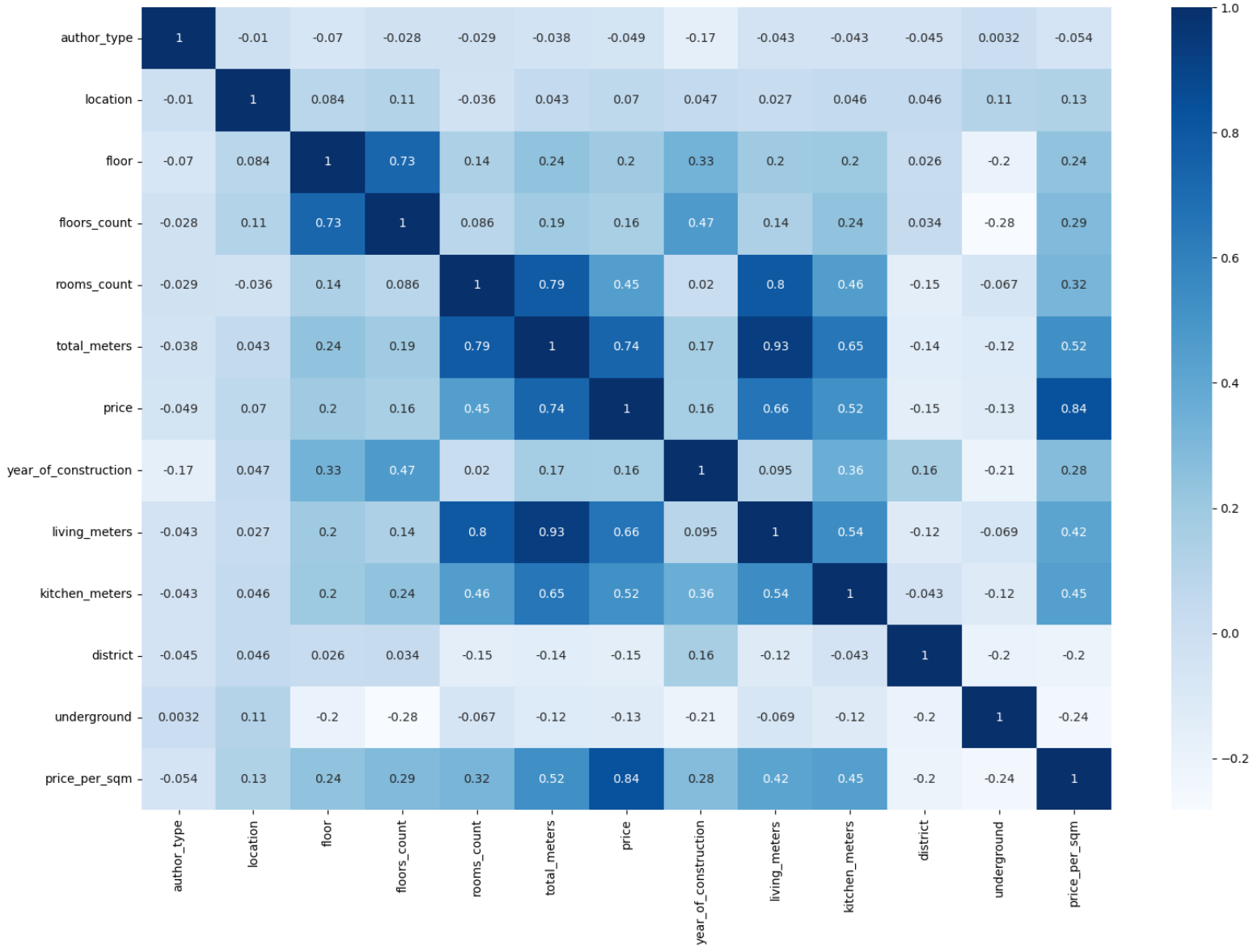
Последним я сделал корреляционную матрицу, которая показывает корреляции между всеми переменными.

Код этой корреляционной матрицы:

Корреляционная матрица

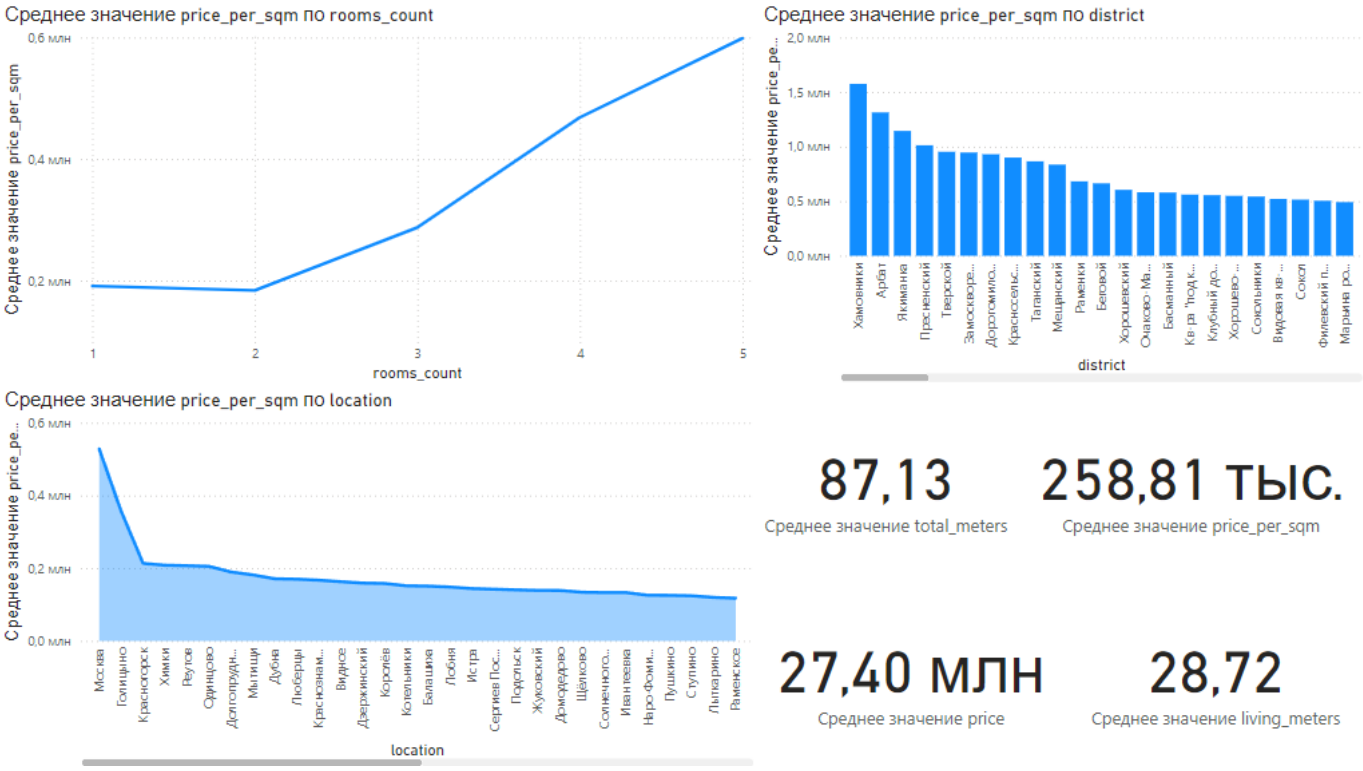
```
plt.figure(figsize=(18, 12))
sns.heatmap(encoded_df.corr(), annot=True, cmap='Blues')
plt.show()
```

Результат этого кода и сама корреляционная матрица:



Цель нашего отчета - это понять от чего зависит цена на квартиру. Т. к. цена может быть разной от общей площади квартиры, то лучше всего смотреть от чего зависит какое-то универсальное значение, в нашем случае это — цена за квадратный метр. Смотря на корреляционную матрицу и на последнюю колонку, которая отвечает именно за корреляцию цены за квадратный метр, можно сказать что **цена квадратного метра зависит от общей площади квартиры, количества комнат в квартире, года постройки дома, местоположения, жилищного метража и метража кухни.**

Так же была выполнена работа в Power BI. Там я создал 3 графика и 4 значения:



Слева-сверху график показывающий зависимость цены квадратного метра от кол-ва комнат в квартире; справа-сверху график показывающий зависимость цены квадратного метра от района, где расположена квартира; слева-снизу график показывающий цену квадратного метра в городах; справа-снизу 4 значений, показывающие среднюю общую площадь квартир, среднюю цену квадратного метра, среднюю цену квартиры и среднюю жилищную площадь.

Заключение

В этом аналитическом отчете были проделаны работы, такие как: парсинг данных, отбор и чистка данных, визуализация и построение модели. Главная цель отчета была выполнена и определяется как: **цена квадратного метра зависит от общей площади квартиры, количества комнат в квартире, года постройки дома, местоположения, жилищного метража и метража кухни.**

В будущем в этот аналитический отчет можно добавить новые параметры и данные, доработать очистку данных, сгенерировать новую визуализацию и исправить недочеты.