УДК 004

# Анализ данных о ценах на квартиры в Еврейской автономной области

*Екимова Яна Сергеевна*
*Приамурский государственный университет имени Шолом-Алейхема*
*Студент*

**Аннотация**
В данной научной статье рассматривается разработка и применение моделей машинного обучения для интеллектуального анализа цен на квартиры в Еврейской автономной области с использованием Google Colaboratory. Для достижения поставленной цели использовался набор данных, содержащий информацию о квартирах, год постройки дома, количество комнат, квадратуре и других атрибутах квартиры. В условиях динамично развивающегося рынка недвижимости и постоянно меняющихся экономических условий, анализ данных о ценах на квартиры позволяет выявлять тенденции и закономерности, прогнозировать изменения цен, а также принимать обоснованные решения в области инвестиций и управления недвижимостью.
**Ключевые слова:** машинное обучение, Google Colaboratory.

# Analysis of data on apartment prices in the Jewish Autonomous Region

*Ekimova Yana Sergeevna*
*Sholom Aleichem Priamurskiy State University*
*Student*

**Abstract**
This scientific article discusses the development and application of machine learning models for the intelligent analysis of apartment prices in the Jewish Autonomous Region using Google Colaboratory. To achieve this goal, a data set containing information about apartments, the year of construction of the house, the number of rooms, the quadrature and other attributes of the apartment was used. In the context of a dynamically developing real estate market and constantly changing economic conditions, the analysis of apartment price data allows you to identify trends and patterns, predict price changes, as well as make informed decisions in the field of investment and property management.
**Keywords:** machine learning, Google Coollaboratory.

## 1. Введение

### 1.1. Актуальность

Google Colab остается актуальным и популярным выбором для проведения исследований, разработки моделей машинного обучения и

выполнения вычислений в облаке. В современном мире, где информационные технологии прочно интегрированы в повседневную жизнь, методы интеллектуального анализа данных (Data Mining) становятся все более важными. Особенно актуальным это становится в сфере недвижимости, где динамика цен на квартиры является ключевым фактором принятия решений о покупке, продаже или инвестициях. В условиях постоянно меняющихся экономических условий и динамично развивающегося рынка, анализ и прогнозирование цен на квартиры позволяют выявить тенденции и закономерности, что крайне важно для всех участников рынка недвижимости.

### 1.2 Обзор исследований

Модель, описанная А.В. Толмачевым и О.Н. Красавиной, представляет собой модель машинного обучения, предназначенную для оценки стоимости квадратного метра недвижимости. Данная модель, вероятно, использует различные характеристики объектов недвижимости, такие как местоположение, площадь, возраст, удобства и рыночные тенденции, для прогнозирования цены за квадратный метр. Алгоритмы машинного обучения, такие как регрессионные модели или нейронные сети, обучаются на наборе данных с историческими данными о недвижимости, чтобы выявить взаимосвязи между этими характеристиками и соответствующими ценами. Цель состоит в том, чтобы предоставить точный и основанный на данных инструмент для оценки недвижимости, который может быть полезен для покупателей, продавцов и профессионалов в области недвижимости [1].

А. М. Васильченко описал как проводить анализ данных при помощи библиотек Python [2].

Н.Е. Косых предложил подход к подбору наиболее производительного набора параметров для объекта классификатора текста. Для вычислений использовал облачный сервис Google Colaboratory, выполняющий код на языке Python внутри браузера, используя виртуальные Постулат. 2024. №1 ISSN 2414-4487 аппаратные ресурсы [3].

Т.М. Татарникова, Е.Д. Пойманова, П.Ю. Богданов, Е.В. Краева, С.А. Веревкин в статье рассмотрели способы и методы изучения и построения нейронных сетей. Показано, что изучение принципов функционирования нейронных сетей, их применение для решения тех или иных задач возможны только через практику. Проведен анализ различных программных сред, которые могут быть использованы на лабораторных и практических занятиях по изучению и применению нейронных сетей. Выделен современный облачный сервис Google Colaboratory, рекомендуемый для обучения основам нейронных сетей благодаря наличию в нем предустановки библиотеки Tensorflow и библиотеки для работы на языке Python, бесплатного доступа к графическим процессорам, возможности написания и выполнения программного кода в браузере, а также отсутствию необходимости специальной настройки сервиса [4].

Е.А. Григорьев, Н.С. Климов в статье рассматривали разведочный анализ данных. Описали инструменты реализации анализа, библиотеки Python. Представлен пример выполненный на данных обнаружению присутствия людей в помещении [5].

### 1.3 Цель исследования

Цель исследования – выполнить анализ данных с помощью Google Colaboratory.

## 2. Материалы и методы

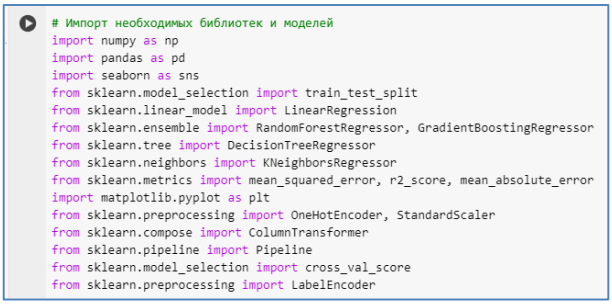
В данном исследовании используется Google Colaboratory для анализа данных.

Набор данных о ценах на квартиры в Еврейской автономной области содержит информацию о различных характеристиках. Он охватывает квадратуру, ремонт, или же в каком районе расположена квартира, продаваемая в Еврейской автономной области.

Данные для работы можно скачать по ссылке https://docs.google.com/spreadsheets/d/1rNW5llhknB7z7Rh0mPzt8ulgM14-XDn1/edit?usp=drive_link&ouid=107887703743266151823&rtpof=true&sd=true.

## 3. Результаты

Для проведения интеллектуального анализа данных о квартирах в Еврейской Автономной Области, импортируем нужные библиотеки и модели (рис.1).



```
# Импорт необходимых библиотек и моделей
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
```

Рисунок 1-Импорт библиотек и моделей

Импортируем нужный dataset и проверяем загрузились ли данные (рис.2). Датасет состоит из следующих столбцов:

1. District (Район): Этот столбец содержит информацию о районе, в котором расположена квартира.
2. Street (Улица): В этом столбце указано название улицы, на которой находится квартира.
3. Number_of_rooms (Количество комнат): Здесь указано общее количество комнат в квартире.
4. Total_quadrature (Общая площадь): Этот столбец содержит информацию о общей площади квартиры в квадратных метрах.
5. Kitchen_area (Площадь кухни): В данном столбце указана

площадь кухни в квадратных метрах.

6. Living_area (Жилая площадь): Здесь указана площадь жилой площади в квадратных метрах.

7. Floor (Этаж): В этом столбце указано, на каком этаже находится квартира.

8. Price (Цена): Этот столбец содержит информацию о стоимости квартиры.

9. Year_the_house_was_built (Год постройки дома): В данном столбце указан год, в котором был построен дом, в котором находится квартира.

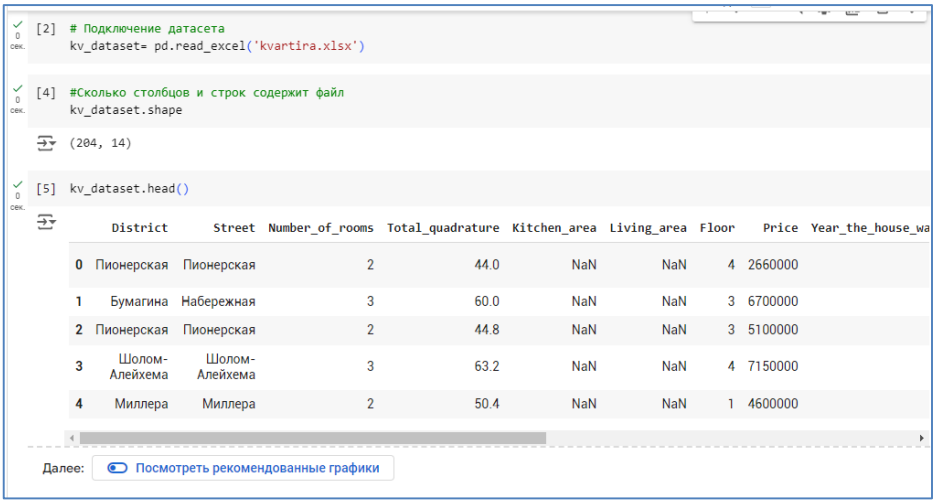
10. House_type (Тип дома): Здесь указано, какой тип дома (панельный, кирпичный и т.д.)

11. Type_of_rooms (Тип комнат): В этом столбце указан тип комнат в квартире (изолированные или смежные).

12. Bathroom (Санузел): Здесь указано совмещенный он или раздельный.

13. Availability_of_a_balcony (Наличие балкона): В данном столбце указано, есть ли балкон в квартире.

14. Repair (Ремонт): Этот столбец содержит информацию о состоянии ремонта в квартире (без ремонта, евроремонт, и т.д.).



```
[2] # Подключение датасета
    kv_dataset= pd.read_excel('kvartira.xlsx')

[4] #Сколько столбцов и строк содержит файл
    kv_dataset.shape

    (204, 14)

[5] kv_dataset.head()
```

| | District | Street | Number_of_rooms | Total_quadrature | Kitchen_area | Living_area | Floor | Price | Year_the_house_wa |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Пионерская | Пионерская | 2 | 44.0 | NaN | NaN | 4 | 2660000 | |
| 1 | Бумагина | Набережная | 3 | 60.0 | NaN | NaN | 3 | 6700000 | |
| 2 | Пионерская | Пионерская | 2 | 44.8 | NaN | NaN | 3 | 5100000 | |
| 3 | Шолом-Алейхема | Шолом-Алейхема | 3 | 63.2 | NaN | NaN | 4 | 7150000 | |
| 4 | Миллера | Миллера | 2 | 50.4 | NaN | NaN | 1 | 4600000 | |

Далее: Посмотреть рекомендованные графики

Рисунок 2-Проверка данных

С помощью метода kv_dataset.info() получаем важную информацию о структуре датасета, что является основой для предварительного анализа данных и подготовки их к последующему использованию в моделях машинного обучения.

```
[6] kv_dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 204 entries, 0 to 203
Data columns (total 14 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   District                    204 non-null    object
 1   Street                      204 non-null    object
 2   Number_of_rooms             204 non-null    int64
 3   Total_quadrature            204 non-null    float64
 4   Kitchen_area                194 non-null    float64
 5   Living_area                 149 non-null    float64
 6   Floor                       204 non-null    int64
 7   Price                       204 non-null    int64
 8   Year_the_house_was_built    204 non-null    int64
 9   House_type                  204 non-null    object
 10  Type_of_rooms               204 non-null    object
 11  Bathroom                    204 non-null    object
 12  Availability_of_a_balcony   204 non-null    object
 13  Repair                      204 non-null    object
dtypes: float64(3), int64(4), object(7)
memory usage: 22.4+ KB
```

Рисунок 3-Информация о данных

Вывод количества вхождений каждого уникального значения позволил оценить распределение данных по категориям и выявить наиболее часто встречающиеся значения.

```
# Вывод количество вхождений каждого уникального значения
print(kv_dataset.District.value_counts())
print(kv_dataset.Street.value_counts())
print(kv_dataset.House_type.value_counts())
print(kv_dataset.Type_of_rooms.value_counts())
print(kv_dataset.Bathroom.value_counts())
print(kv_dataset.Availability_of_a_balcony.value_counts())
print(kv_dataset.Repair.value_counts())

Набережная            10
Широкая                8
Бумагина               8
Юбилейная              8
Миллера                7
Чапаева                7
Невская                5
Димитрова              5
Стяжкина               4
Комбайностроителей     4
Калинина               4
Дружбы                 4
Московская             3
Карла Маркса           3
Дзержинского           3
пр- т 60-летия СССР    2
Казакевича             2
Парковая               2
```

Рисунок 4-Анализ уникального значения

```
Name: count, dtype: int64
House_type
кирпичный               145
панельный                52
блочный                   5
монолитно-кирпичный       2
Name: count, dtype: int64
Type_of_rooms
смежные          118
изолированные     86
Name: count, dtype: int64
Bathroom
совмещённый    136
раздельный      68
Name: count, dtype: int64
Availability_of_a_balcony
есть    176
нет      28
Name: count, dtype: int64
Repair
косметический      133
евро                31
требует ремонта     28
дизайнерский        12
Name: count, dtype: int64
```

Рисунок 5- Анализ уникального значения

Заполним пропущенные значения средним значением в столбцах 'Kitchen_area' и 'Living_area'.

```
[8] # Пропущенные значения заполнить средним значением столбца Kitchen_area
    kv_dataset['Kitchen_area'].fillna(kv_dataset['Kitchen_area'].mean(), inplace = True)

    sns.set_style('whitegrid')
    %matplotlib inline

    g = sns.FacetGrid(kv_dataset, col='Total_quadrature ')
    g.map(plt.hist, 'Kitchen_area', bins=10)

<seaborn.axisgrid.FacetGrid at 0x7d2568b36e90>

[9] # Пропущенные значения заполнить средним значением столбца Living_area
    kv_dataset['Living_area'].fillna(kv_dataset['Living_area'].mean(), inplace = True)
    sns.set_style('whitegrid')
    %matplotlib inline

    g = sns.FacetGrid(kv_dataset, col='Total_quadrature ')
    g.map(plt.hist, 'Living_area', bins=10)

<seaborn.axisgrid.FacetGrid at 0x7d2528seea70>
```

Рисунок 6-Заполнение пропущенных значений

```
kv_dataset.head()
```

| | District | Street | Number_of_rooms | Total_quadrature | Kitchen_area | Living_area | Floor | Price | Year_the_house_wa |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Пионерская | Пионерская | 2 | 44.0 | 8.346392 | 36.085235 | 4 | 2660000 | |
| 1 | Бумагина | Набережная | 3 | 60.0 | 8.346392 | 36.085235 | 3 | 6700000 | |
| 2 | Пионерская | Пионерская | 2 | 44.8 | 8.346392 | 36.085235 | 3 | 5100000 | |
| 3 | Шолом-Алейхема | Шолом-Алейхема | 3 | 63.2 | 8.346392 | 36.085235 | 4 | 7150000 | |
| 4 | Миллера | Миллера | 2 | 50.4 | 8.346392 | 36.085235 | 1 | 4600000 | |

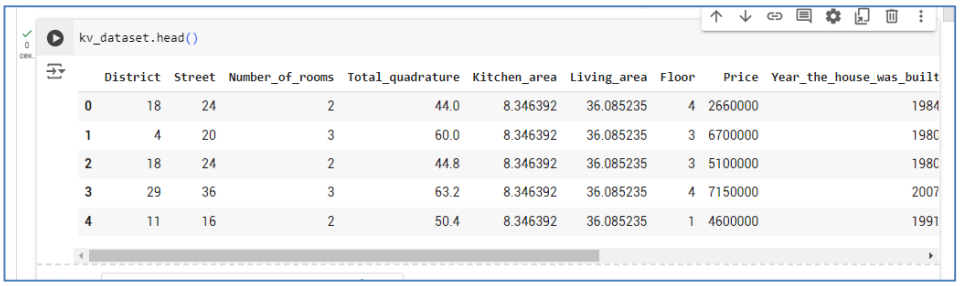
Рисунок 7-Заполнение пропущенных значений

Определим список столбцов, которые требуется преобразовать в DataFrame, создадим экземпляр класса LabelEncoder для преобразования категориальных данных в числовые.

```
[11] # Определение списка столбцов, которые требуется преобразовать
     columns_to_convert = ['Availability_of_a_balcony', 'Repair', 'Type_of_rooms', 'Bathroom', 'House_type']

[12] # Создание экземпляра класса LabelEncoder для преобразования категориальных данных в числовые
     label_encoder = LabelEncoder()

[13] # Цикл по каждому столбцу в списке columns_to_convert
     for column in columns_to_convert:
         # Применение LabelEncoder к каждому столбцу, преобразуя категориальные данные в числовые
         kv_dataset[column] = label_encoder.fit_transform(kv_dataset[column])

[18] # Создание экземпляра класса LabelEncoder для преобразования категориальных данных в числовые
     label_encoder = LabelEncoder()

     # Применение LabelEncoder к столбцу 'District' для преобразования названий района в числовые коды
     kv_dataset['District'] = label_encoder.fit_transform(kv_dataset['District'])

     # Применение LabelEncoder к столбцу 'Street' для преобразования названий улиц в числовые коды
     kv_dataset['Street'] = label_encoder.fit_transform(kv_dataset['Street'])
```

Рисунок 8-Преобразование категориальных данных в числовые

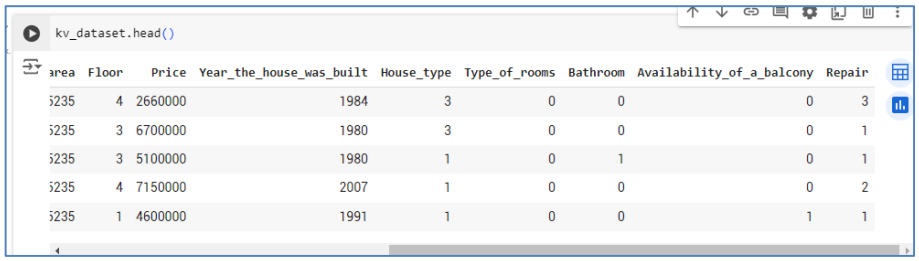
Вывод первых строк датасета, для того, чтобы посмотреть на категориальную замену.



```
kv_dataset.head()
```

| | District | Street | Number_of_rooms | Total_quadrature | Kitchen_area | Living_area | Floor | Price | Year_the_house_was_built |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 18 | 24 | 2 | 44.0 | 8.346392 | 36.085235 | 4 | 2660000 | 1984 |
| 1 | 4 | 20 | 3 | 60.0 | 8.346392 | 36.085235 | 3 | 6700000 | 1980 |
| 2 | 18 | 24 | 2 | 44.8 | 8.346392 | 36.085235 | 3 | 5100000 | 1980 |
| 3 | 29 | 36 | 3 | 63.2 | 8.346392 | 36.085235 | 4 | 7150000 | 2007 |
| 4 | 11 | 16 | 2 | 50.4 | 8.346392 | 36.085235 | 1 | 4600000 | 1991 |

Рисунок 9-Преобразование категориальных данных в числовые

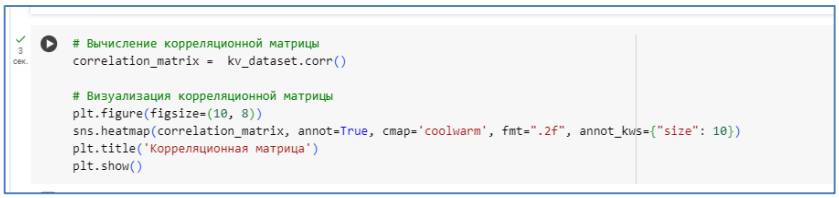


```
kv_dataset.head()
```

| area | Floor | Price | Year_the_house_was_built | House_type | Type_of_rooms | Bathroom | Availability_of_a_balcony | Repair |
|---|---|---|---|---|---|---|---|---|
| 5235 | 4 | 2660000 | 1984 | 3 | 0 | 0 | 0 | 3 |
| 5235 | 3 | 6700000 | 1980 | 3 | 0 | 0 | 0 | 1 |
| 5235 | 3 | 5100000 | 1980 | 1 | 0 | 1 | 0 | 1 |
| 5235 | 4 | 7150000 | 2007 | 1 | 0 | 0 | 0 | 2 |
| 5235 | 1 | 4600000 | 1991 | 1 | 0 | 0 | 1 | 1 |

Рисунок 10-Преобразование категориальных данных в числовые

Продолжим с вычисления корреляционной матрицы для всех числовых столбцов данных и визуализируем её в виде тепловой карты с помощью библиотеки Seaborn.



```
# Вычисление корреляционной матрицы
correlation_matrix = kv_dataset.corr()

# Визуализация корреляционной матрицы
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", annot_kws={"size": 10})
plt.title('Корреляционная матрица')
plt.show()
```

Рисунок 11 – Код

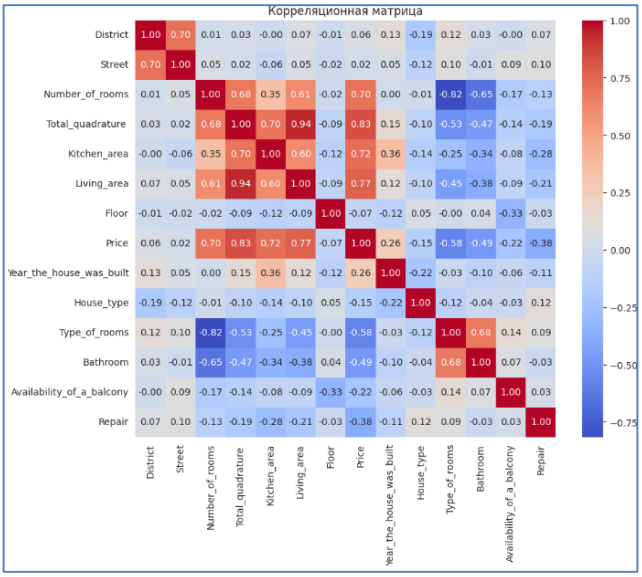



Рисунок 12 – Корреляционная матрица

Между переменными District и Street наблюдается умеренная положительная корреляция (0.702245), что может указывать на то, что

определенные районы связаны с конкретными улицами.

Number_of_rooms имеет среднюю положительную корреляцию с Total_quadrature (0.683915) и Living_area (0.606375), что логично, так как квартиры с большим количеством комнат обычно имеют большую общую площадь и жилую площадь.

Price имеет сильную положительную корреляцию с Total_quadrature (0.830855), Kitchen_area (0.717689), и Living_area (0.772683), что указывает на то, что более дорогие квартиры обычно имеют большую общую и жилую площадь.

Type_of_rooms имеет сильную отрицательную корреляцию с Number_of_rooms (-0.815115), что может быть нетипичным и требует дополнительного анализа, так как обычно тип комнат не должен иметь отрицательной связи с количеством комнат.

Bathroom имеет отрицательную корреляцию с Number_of_rooms (-0.648595) и Type_of_rooms (-0.680874), что также требует дополнительного изучения, так как может быть нелогичным.

```
#вывод корреляционной матрицы
print(correlation_matrix)
```

```
                          District    Street  Number_of_rooms  \
District                  1.000000  0.702245         0.011990
Street                    0.702245  1.000000         0.050071
Number_of_rooms           0.011990  0.050071         1.000000
Total_quadrature          0.033096  0.023203         0.683915
Kitchen_area             -0.000455 -0.062011         0.345774
Living_area               0.066078  0.050638         0.606375
Floor                    -0.011122 -0.018203        -0.022008
Price                     0.058798  0.015905         0.704955
Year_the_house_was_built  0.125696  0.045212         0.004066
House_type               -0.187401 -0.117147        -0.006733
Type_of_rooms             0.121158  0.098520        -0.815115
Bathroom                  0.032843 -0.007336        -0.648595
Availability_of_a_balcony -0.003244  0.086857        -0.170733
Repair                    0.065366  0.099282        -0.129411

                          Total_quadrature  Kitchen_area  Living_area  \
District                          0.033096     -0.000455     0.066078
Street                            0.023203     -0.062011     0.050638
Number_of_rooms                   0.683915      0.345774     0.606375
Total_quadrature                  1.000000      0.700267     0.937519
Kitchen_area                      0.700267      1.000000     0.604489
Living_area                       0.937519      0.604489     1.000000
Floor                            -0.091155     -0.123224    -0.093753
Price                             0.830855      0.717689     0.772683
Year_the_house_was_built          0.146650      0.355192     0.123730
House_type                       -0.101235     -0.143735    -0.095468
Type_of_rooms                    -0.526982     -0.252811    -0.454144
```

Рисунок 12- Таблица корреляционной матрицы

Вызов kv_dataset.describe() позволяет получить представление о центральной тенденции, разбросе и форме распределения данных в числовых столбцах датасета. Эта информация важна для понимания основных характеристик данных и выявления возможных аномалий или выбросов.

```
# Вывод статистических данных
kv_dataset.describe()
```

| | District | Street | Number_of_rooms | Total_quadrature | Kitchen_area | Living_area | Floor | Price | Year_ |
|---|---|---|---|---|---|---|---|---|---|
| count | 204.000000 | 204.000000 | 204.000000 | 204.000000 | 204.000000 | 204.000000 | 204.000000 | 2.040000e+02 | |
| mean | 17.044118 | 22.642157 | 2.014706 | 50.537745 | 8.346392 | 36.085235 | 3.000000 | 4.429753e+06 | |
| std | 9.839251 | 11.840666 | 1.205398 | 36.362961 | 4.181653 | 26.612939 | 1.485563 | 2.431914e+06 | |
| min | 0.000000 | 0.000000 | 1.000000 | 18.000000 | 2.000000 | 9.700000 | 1.000000 | 5.000000e+05 | |
| 25% | 7.750000 | 13.750000 | 1.000000 | 30.900000 | 6.000000 | 20.000000 | 2.000000 | 2.850000e+06 | |
| 50% | 18.000000 | 24.000000 | 1.000000 | 38.950000 | 7.350000 | 36.085235 | 3.000000 | 3.788888e+06 | |
| 75% | 26.250000 | 33.000000 | 3.000000 | 60.125000 | 9.075000 | 39.325000 | 4.000000 | 5.112500e+06 | |
| max | 30.000000 | 38.000000 | 5.000000 | 346.300000 | 35.000000 | 250.000000 | 8.000000 | 2.150000e+07 | |

Рисунок 13 – Статистические данные

Были построены несколько типов графиков. Одним из них является гистограмма (см. рис 14), которая позволяет визуализировать цены на квартиры.

```
plt.figure(figsize=(10, 6))
sns.histplot(kv_dataset['Price'], bins=30, kde=True)
plt.title('Распределение цен на квартиры')
plt.xlabel('Цена')
plt.ylabel('Частота')
plt.show()
```

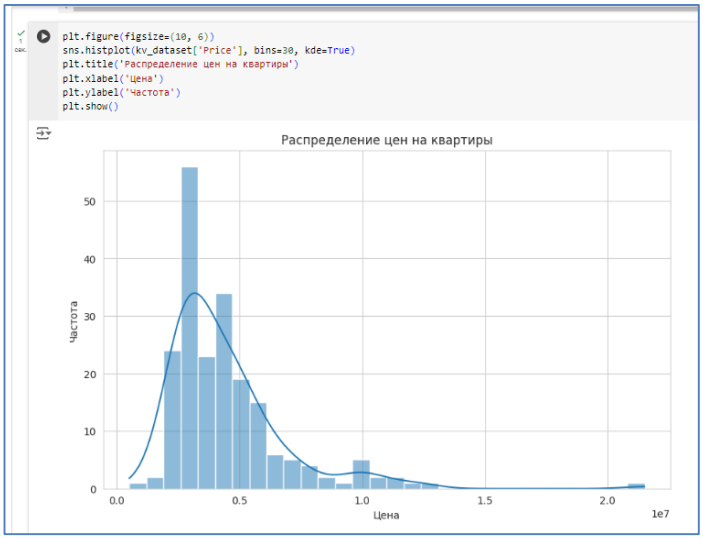



Рисунок 14 – Гистограмма

Установим стиль графика на "whitegrid" для создания графиков с сеткой на белом фоне sns.set(style="whitegrid").

Вторая диаграмма (см. рис 15), позволяет увидеть зависимость цены от квадратуры квартиры.

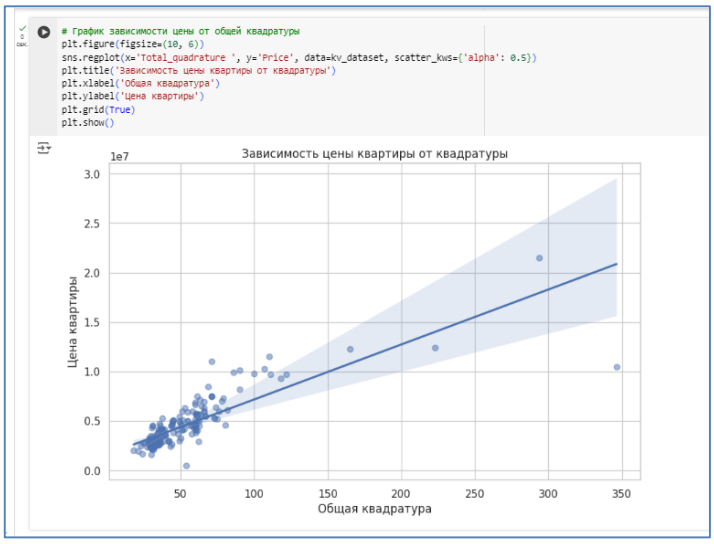



Рисунок 15 – Диаграмма рассеивания

Следующая диаграмма (см. рис 16), позволяет увидеть зависимость цены от количества комнат.

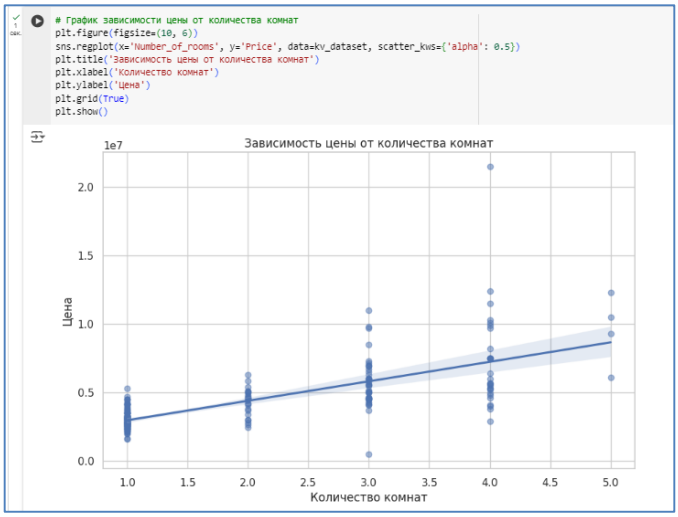



Рисунок 16. Диаграмма рассеивания

Следующая диаграмма (см. рис 17), позволяет увидеть зависимость цены от района.

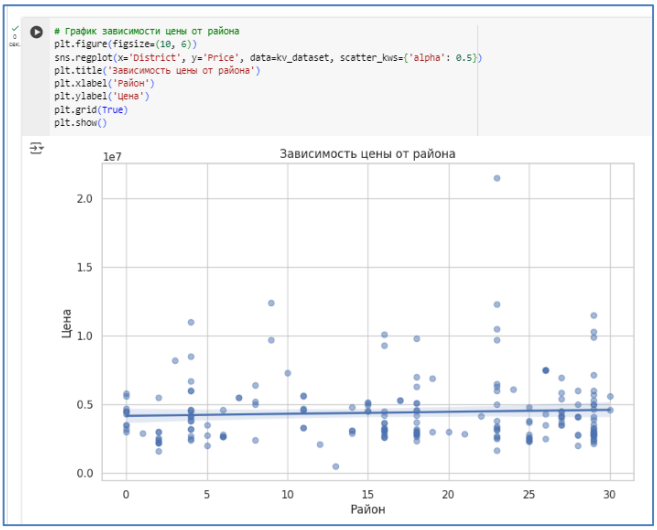



Рисунок 17 – Диаграмма рассеивания

Следующая диаграмма (см. рис 18), позволяет увидеть зависимость цены от наличия ремонта.

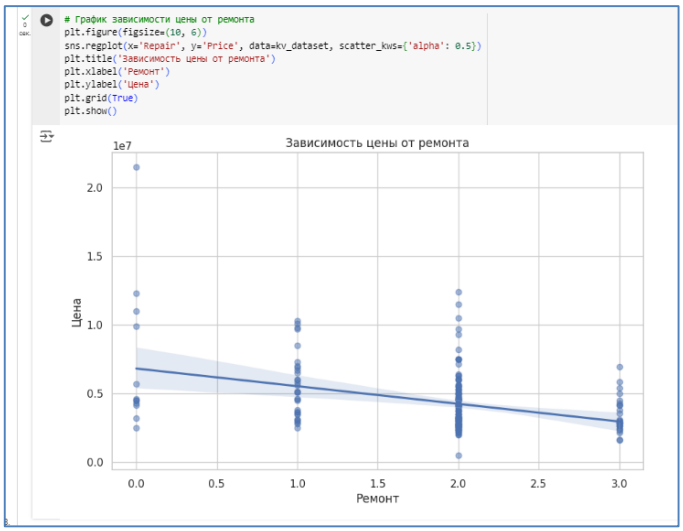



Рисунок 18– Диаграмма рассеивания

Теперь создадим бокс-плот для сравнения цен на жилье в зависимости от типа дома, в наборе данных kv_dataset. Бокс-плот визуализирует распределение цен для каждого типа дома, отображая медиану, квартили и выбросы.

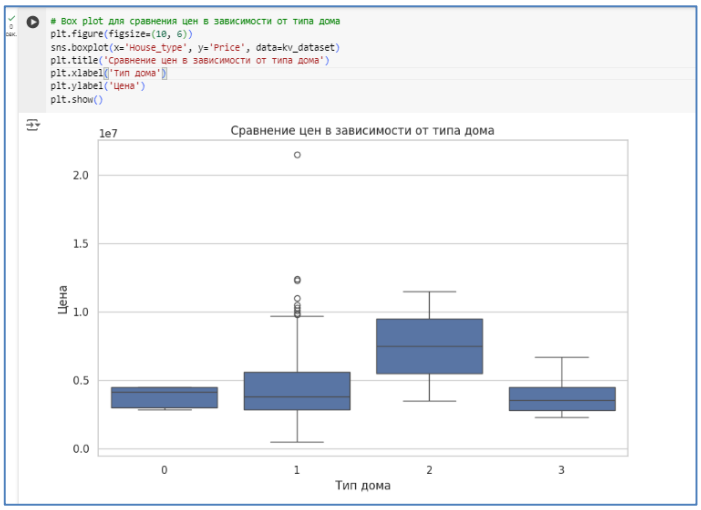



Рисунок 19– бокс-плот

Создадим бокс-плот, который сравнит цены недвижимости (Price) в зависимости от типа комнат (Type_of_rooms) в наборе данных kv_dataset.

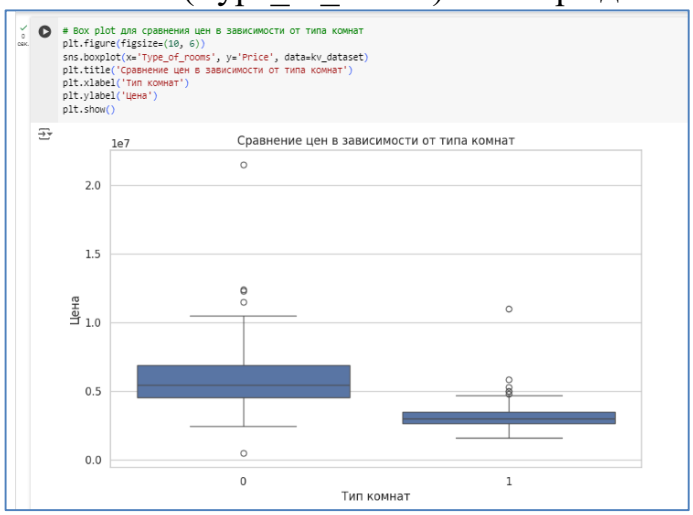



Рисунок 20– бокс-плот

Создадим четыре графика рассеяния с разными цветами, чтобы проанализировать важность различных переменных относительно цены недвижимости. На каждой из четырех осей были построены столбчатые диаграммы , отображающие количество вхождений каждого уникального значения в соответствующих столбцах датасета: Number_of_rooms, Repair, District, House_type, Price. В результате получается набор из четырех графиков рассеяния, которые визуализируют отношения между различными переменными и ценой недвижимости. Разные цвета помогают отличить значения года постройки дома и облегчают анализ влияния этих переменных на цену.

```
# Построение нескольких графиков рассеивания с разными цветами для анализа важности данных по отношению к цене
fig, axs = plt.subplots(2, 2, figsize=(15, 10))

# Рассеивание с разными цветами и легендой
scatter = axs[0, 0].scatter(kv_dataset['Number_of_rooms'], kv_dataset['Price'], c=kv_dataset['Year_the_house_was_built'], cmap='viridis')
axs[0, 0].set_title('Количество комнат vs Цена')

scatter = axs[0, 1].scatter(kv_dataset['Repair'], kv_dataset['Price'], c=kv_dataset['Year_the_house_was_built'], cmap='viridis')
axs[0, 1].set_title('Наличие ремонта vs Цена')

scatter = axs[1, 0].scatter(kv_dataset['District'], kv_dataset['Price'], c=kv_dataset['Year_the_house_was_built'], cmap='viridis')
plt.xticks(rotation=90)
axs[1, 0].set_title('Район vs Цена')

scatter = axs[1, 1].scatter(kv_dataset['House_type'], kv_dataset['Price'], c=kv_dataset['Year_the_house_was_built'], cmap='viridis')
axs[1, 1].set_title('Тип дома vs Цена')

# Добавление легенды
fig.colorbar(scatter, ax=axs, orientation='vertical')

<matplotlib.colorbar.Colorbar at 0x7d24fc31a5c0>
```

Рисунок 21– Построение нескольких графиков

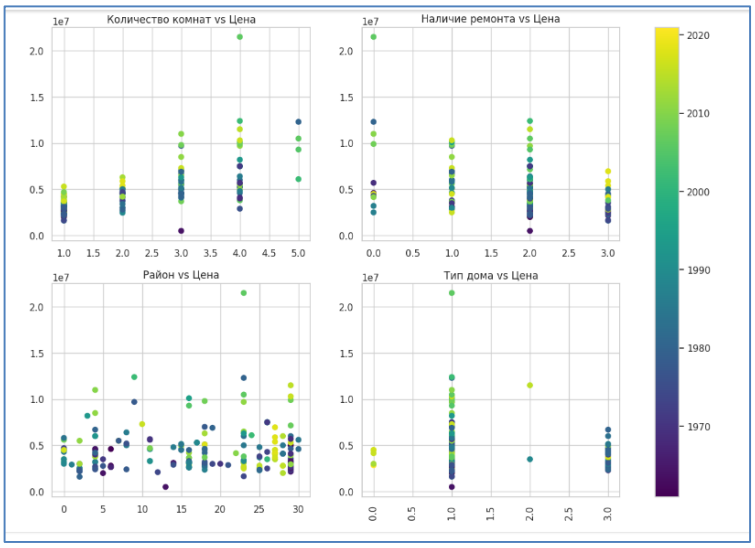



Рисунок 22– График рассеивания

Была построена модель линейной регрессии (см.рис.23), которая может использоваться для предсказания цены на основе различных данных о квартирах.

**Линейная регрессия**

```
[53] from sklearn.preprocessing import OneHotEncoder
     from sklearn.pipeline import Pipeline
     from sklearn.metrics import mean_squared_error, r2_score

[54] # Определение списка категориальных признаков (features), которые содержат текстовые данные
     categorical_features = ['District', 'Street', 'House_type', 'Type_of_rooms', 'Bathroom', 'Availability_of_a_balcony', 'Repair']

     # Определение списка числовых признаков, которые содержат числовые данные
     numerical_features = ['Number_of_rooms', 'Total_quadrature ', 'Kitchen_area', 'Living_area', 'Floor', 'Year_the_house_was_built']

[55] # Создание преобразователя для категориальных и числовых признаков
     preprocessor = ColumnTransformer(
         transformers=[
             ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_features),
             ('num', 'passthrough', numerical_features)
         ],
         remainder='drop'  # Удаление неиспользуемых столбцов
     )

[56] # Создание модели линейной регрессии с препроцессором
     model = Pipeline(steps=[
         ('preprocessor', preprocessor),
         ('regressor', LinearRegression())
     ])

[57] # Разделение данных на признаки (X) и целевую переменную (y)
     X = kv_dataset.drop('Price', axis=1)  # признаки
     y = kv_dataset['Price']  # целевая переменная

[58] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Рисунок 23– Линейная регрессия


Рисунок 24– Линейная регрессия

Для анализа данных о квартирах в Еврейской Автономной Области можно использовать случайный лес для предсказания цены на основе различных данных о квартирах.

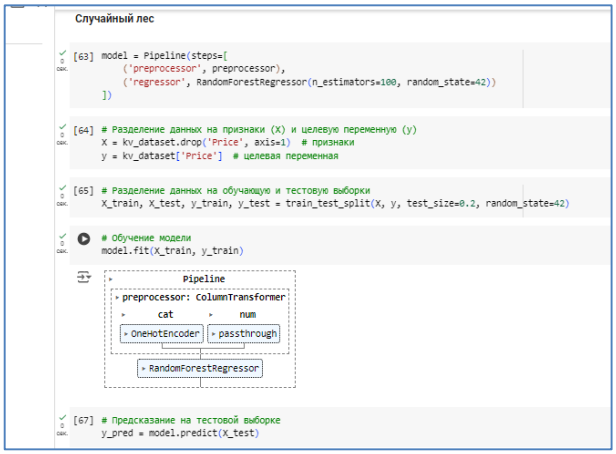
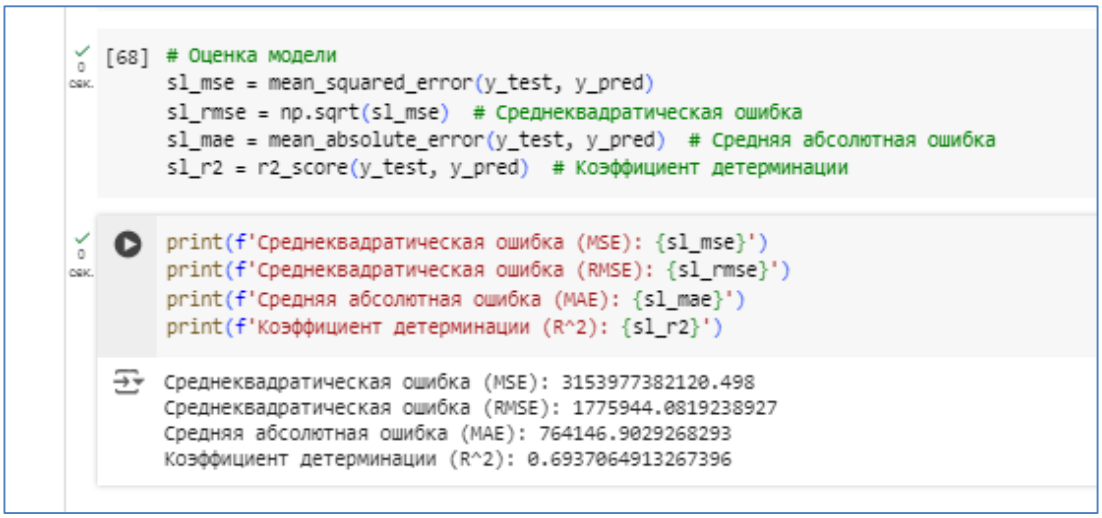


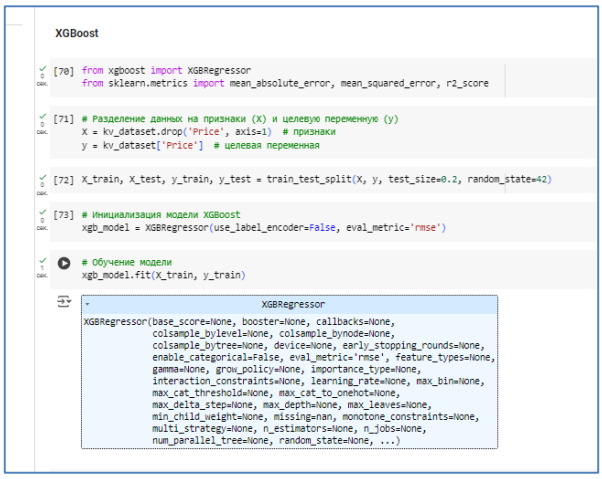

Рисунок 25– Случайный лес



```
[68] # Оценка модели
     sl_mse = mean_squared_error(y_test, y_pred)
     sl_rmse = np.sqrt(sl_mse)  # Среднеквадратическая ошибка
     sl_mae = mean_absolute_error(y_test, y_pred)  # Средняя абсолютная ошибка
     sl_r2 = r2_score(y_test, y_pred)  # Коэффициент детерминации

     print(f'Среднеквадратическая ошибка (MSE): {sl_mse}')
     print(f'Среднеквадратическая ошибка (RMSE): {sl_rmse}')
     print(f'Средняя абсолютная ошибка (MAE): {sl_mae}')
     print(f'Коэффициент детерминации (R^2): {sl_r2}')

Среднеквадратическая ошибка (MSE): 3153977382120.498
Среднеквадратическая ошибка (RMSE): 1775944.0819238927
Средняя абсолютная ошибка (MAE): 764146.9029268293
Коэффициент детерминации (R^2): 0.6937064913267396
```

Рисунок 26– Случайный лес

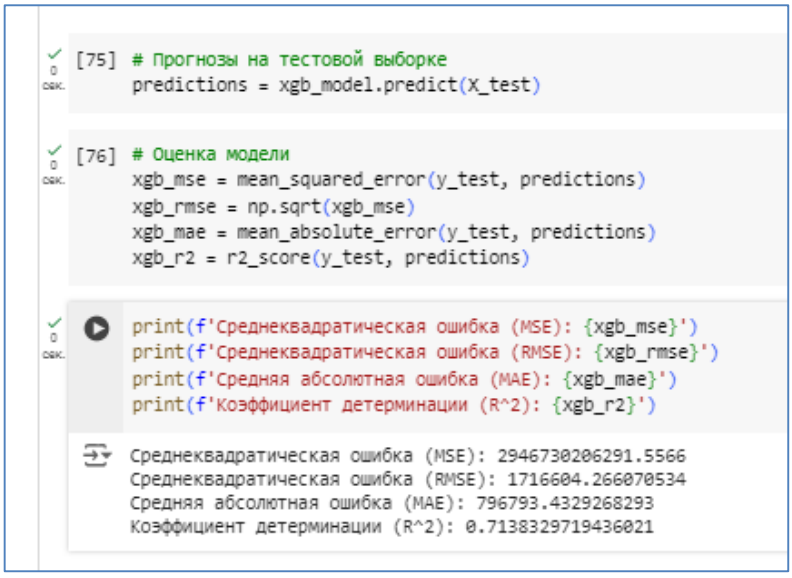
Используем метод XGBoost для предсказания цены на основе различных данных о квартирах.



XGBoost

```
[70] from xgboost import XGBRegressor
     from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

[71] # Разделение данных на признаки (X) и целевую переменную (y)
     X = kv_dataset.drop('Price', axis=1)  # признаки
     y = kv_dataset['Price']  # целевая переменная

[72] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[73] # Инициализация модели XGBoost
     xgb_model = XGBRegressor(use_label_encoder=False, eval_metric='rmse')

     # Обучение модели
     xgb_model.fit(X_train, y_train)

XGBRegressor
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric='rmse', feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

Рисунок 27– XGBoost



```
[75] # Прогнозы на тестовой выборке
     predictions = xgb_model.predict(X_test)

[76] # Оценка модели
     xgb_mse = mean_squared_error(y_test, predictions)
     xgb_rmse = np.sqrt(xgb_mse)
     xgb_mae = mean_absolute_error(y_test, predictions)
     xgb_r2 = r2_score(y_test, predictions)

     print(f'Среднеквадратическая ошибка (MSE): {xgb_mse}')
     print(f'Среднеквадратическая ошибка (RMSE): {xgb_rmse}')
     print(f'Средняя абсолютная ошибка (MAE): {xgb_mae}')
     print(f'Коэффициент детерминации (R^2): {xgb_r2}')

Среднеквадратическая ошибка (MSE): 2946730206291.5566
Среднеквадратическая ошибка (RMSE): 1716604.266070534
Средняя абсолютная ошибка (MAE): 796793.4329268293
Коэффициент детерминации (R^2): 0.7138329719436021
```

Рисунок 28– XGBoost

Используем метод KNeighborsRegressor для предсказания цены на основе различных данных о квартирах.


Рисунок 29– KNeighborsRegressor

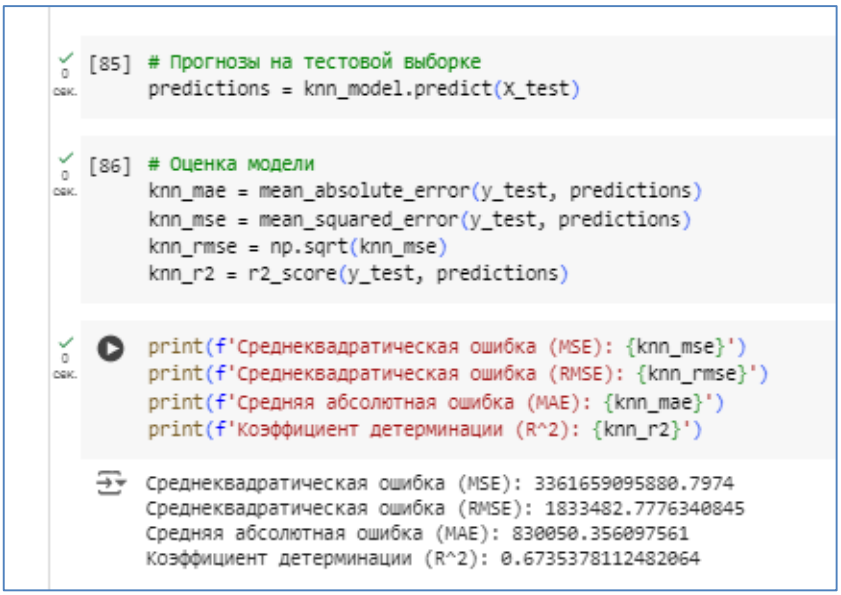



Рисунок 30– KNeighborsRegressor

Можно использовать DecisionTreeRegressor-дерево решений для предсказания для предсказания цены на основе различных данных о квартирах.


Рисунок 31– DecisionTreeRegressor

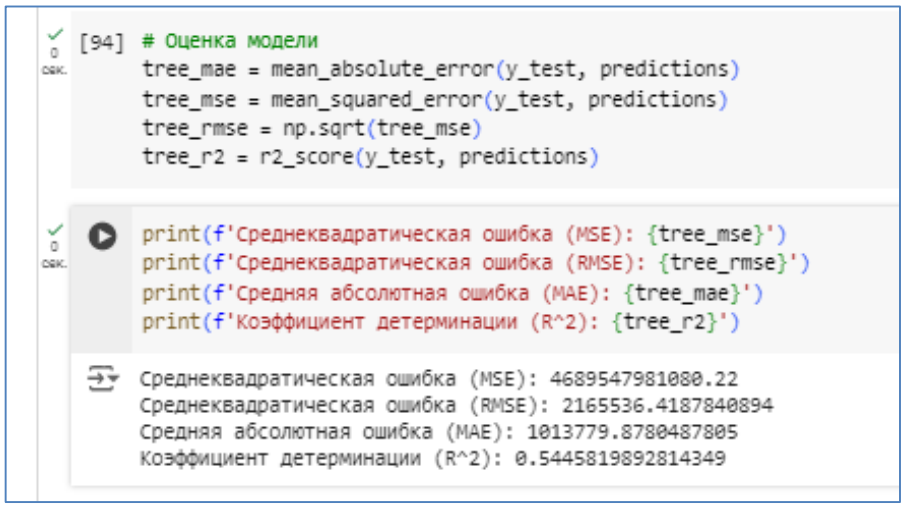



Рисунок 32– DecisionTreeRegressor

Используем метод ближайших соседей.

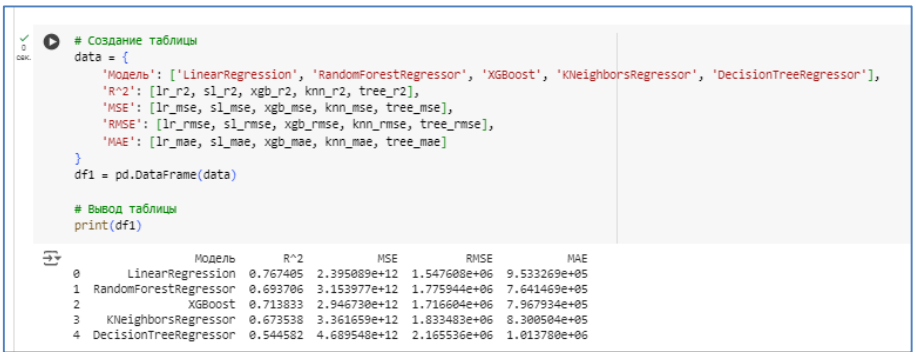
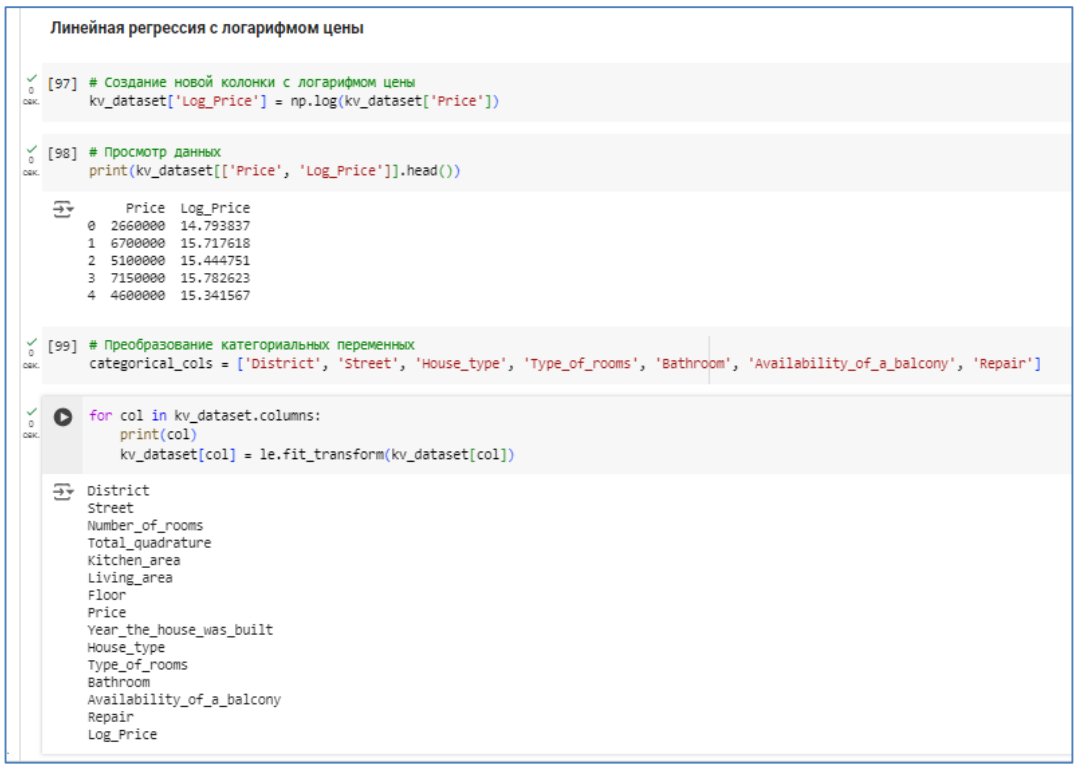



Рисунок 33– Метод ближайших соседей

В случае анализа данных о квартирах в Еврейской Автономной Области можно использовать линейную регрессию с логарифмом.



**Линейная регрессия с логарифмом цены**

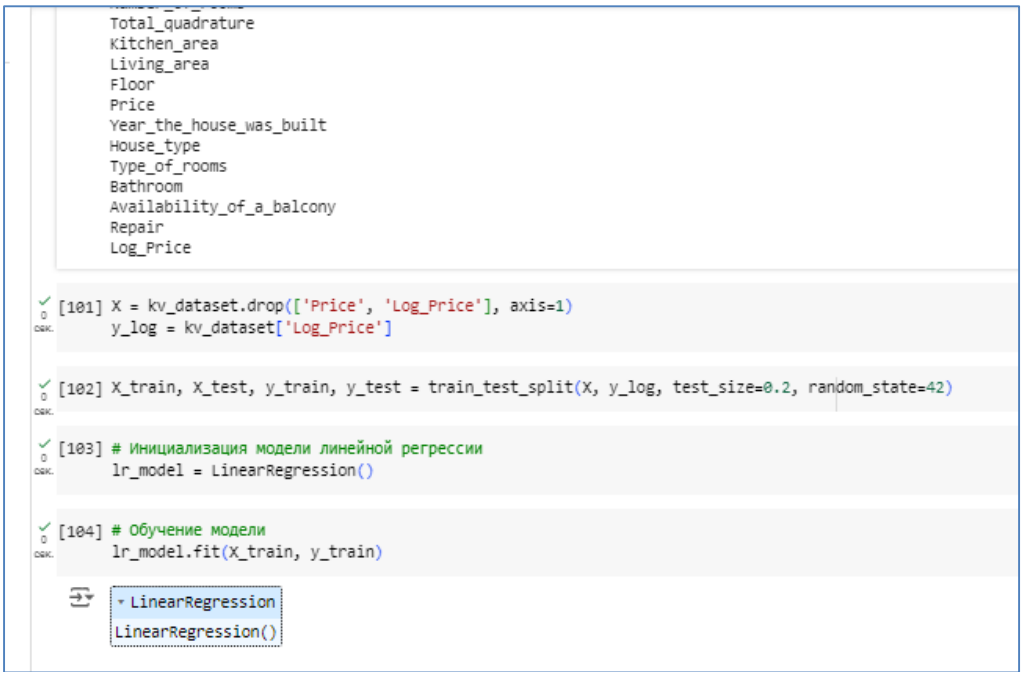
```
[97] # Создание новой колонки с логарифмом цены
     kv_dataset['Log_Price'] = np.log(kv_dataset['Price'])

[98] # Просмотр данных
     print(kv_dataset[['Price', 'Log_Price']].head())
```

```
     Price  Log_Price
0  2660000  14.793837
1  6700000  15.717618
2  5100000  15.444751
3  7150000  15.782623
4  4600000  15.341567
```

```
[99] # Преобразование категориальных переменных
     categorical_cols = ['District', 'Street', 'House_type', 'Type_of_rooms', 'Bathroom', 'Availability_of_a_balcony', 'Repair']

     for col in kv_dataset.columns:
         print(col)
         kv_dataset[col] = le.fit_transform(kv_dataset[col])
```

```
District
Street
Number_of_rooms
Total_quadrature
Kitchen_area
Living_area
Floor
Price
Year_the_house_was_built
House_type
Type_of_rooms
Bathroom
Availability_of_a_balcony
Repair
Log_Price
```

Рисунок 34– Линейная регрессия с логарифмом



```
Total_quadrature
Kitchen_area
Living_area
Floor
Price
Year_the_house_was_built
House_type
Type_of_rooms
Bathroom
Availability_of_a_balcony
Repair
Log_Price
```

```
[101] X = kv_dataset.drop(['Price', 'Log_Price'], axis=1)
      y_log = kv_dataset['Log_Price']

[102] X_train, X_test, y_train, y_test = train_test_split(X, y_log, test_size=0.2, random_state=42)

[103] # Инициализация модели линейной регрессии
      lr_model = LinearRegression()

[104] # Обучение модели
      lr_model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

Рисунок 35– Линейная регрессия с логарифмом

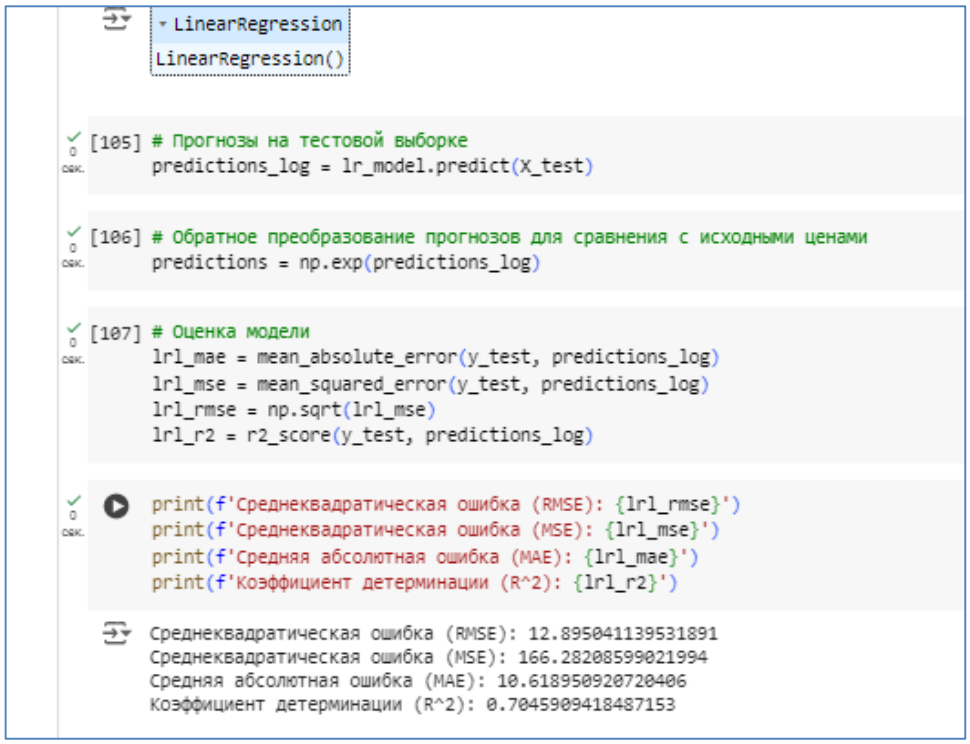



Рисунок 36– Линейная регрессия с логарифмом

В случае анализа данных о квартирах в Еврейской Автономной Области можно использовать случайный лес с логарифмом цены.

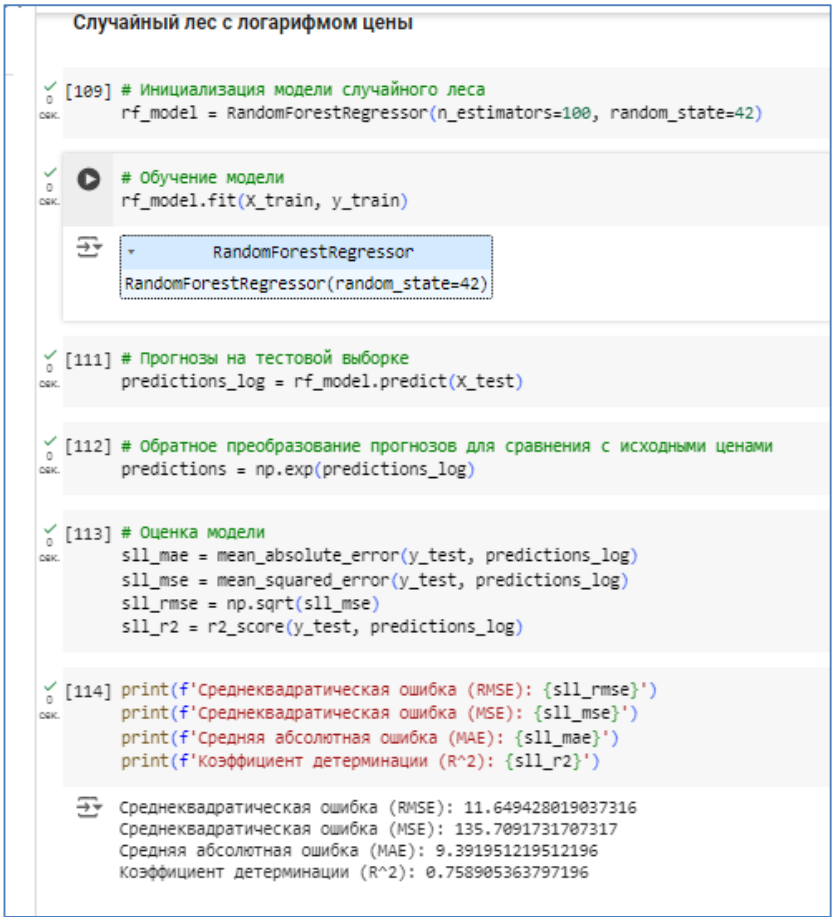



Рисунок 37– Случайный лес с логарифмом

В случае анализа данных о квартирах в Еврейской Автономной Области можно использовать xgboost с логарифмом.

**XGBoost с логарифмом цены**

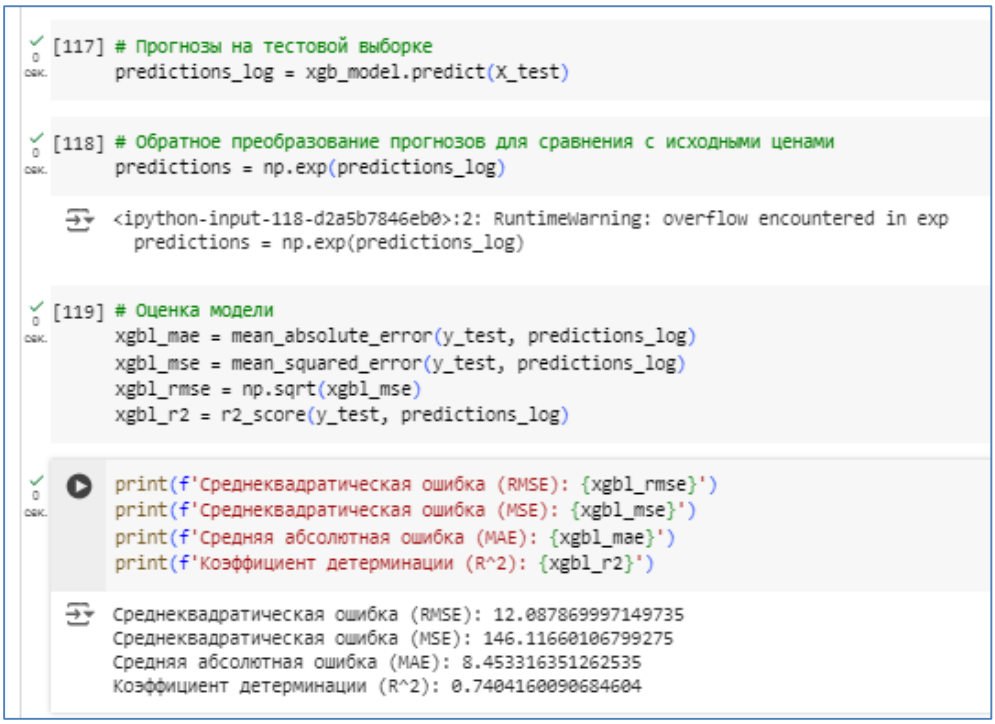
```
[115] # Инициализация модели XGBoost
      xgb_model = XGBRegressor(use_label_encoder=False, eval_metric='rmse', random_state=42)

[116] # Обучение модели
      xgb_model.fit(X_train, y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric='rmse', feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

```
[117] # Прогнозы на тестовой выборке
      predictions_log = xgb_model.predict(X_test)

      # Обратное преобразование прогнозов для сравнения с исходными ценами
      predictions = np.exp(predictions_log)
```

```
<ipython-input-118-d2a5b7846eb0>:2: RuntimeWarning: overflow encountered in exp
  predictions = np.exp(predictions_log)
```

Рисунок 38– xgboost с логарифмом



```
[117] # Прогнозы на тестовой выборке
      predictions_log = xgb_model.predict(X_test)

[118] # Обратное преобразование прогнозов для сравнения с исходными ценами
      predictions = np.exp(predictions_log)
```

```
<ipython-input-118-d2a5b7846eb0>:2: RuntimeWarning: overflow encountered in exp
  predictions = np.exp(predictions_log)
```

```
[119] # Оценка модели
      xgbl_mae = mean_absolute_error(y_test, predictions_log)
      xgbl_mse = mean_squared_error(y_test, predictions_log)
      xgbl_rmse = np.sqrt(xgbl_mse)
      xgbl_r2 = r2_score(y_test, predictions_log)

      print(f'Среднеквадратическая ошибка (RMSE): {xgbl_rmse}')
      print(f'Среднеквадратическая ошибка (MSE): {xgbl_mse}')
      print(f'Средняя абсолютная ошибка (MAE): {xgbl_mae}')
      print(f'Коэффициент детерминации (R^2): {xgbl_r2}')
```

```
Среднеквадратическая ошибка (RMSE): 12.087869997149735
Среднеквадратическая ошибка (MSE): 146.11660106799275
Средняя абсолютная ошибка (MAE): 8.453316351262535
Коэффициент детерминации (R^2): 0.7404160090684604
```

Рисунок 39– xgboost с логарифмом

В случае анализа данных о квартирах в Еврейской Автономной Области можно использовать метод KNeighborsRegressor с логарифмом цены.


Рисунок 40– KNeighborsRegressor с логарифмом цены

В случае анализа данных о квартирах в Еврейской Автономной Области можно использовать метод DecisionTreeRegressor с логарифмом цены.

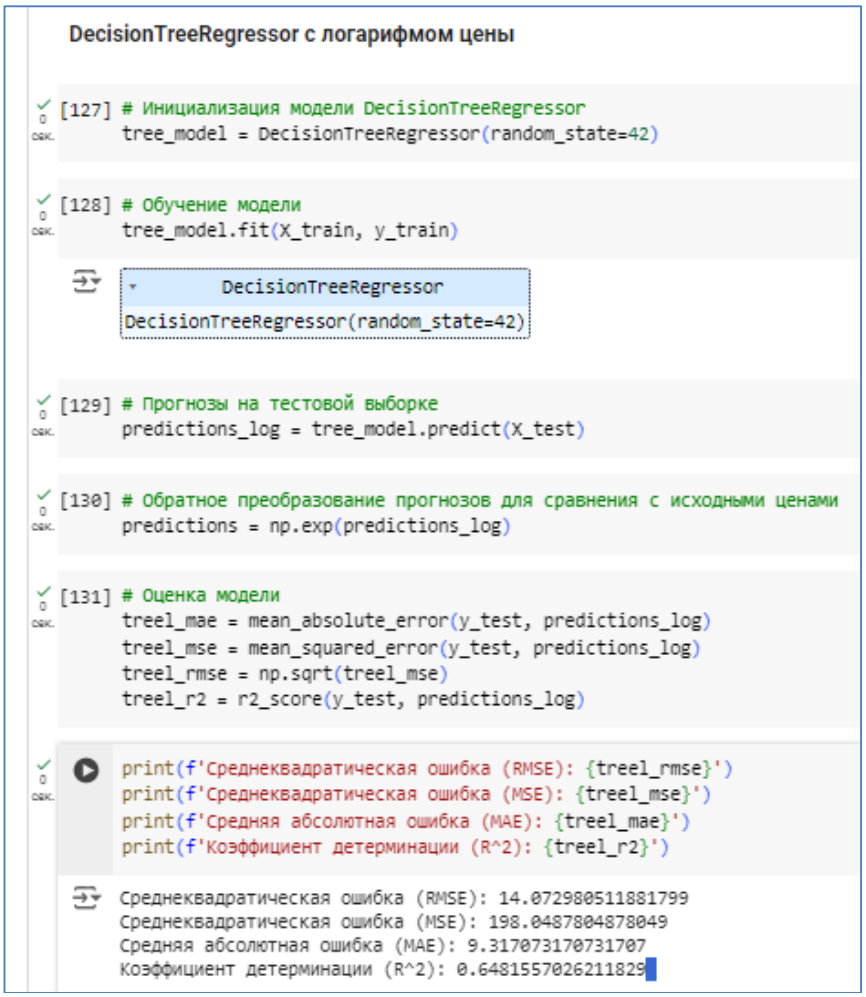



Рисунок 41– DecisionTreeRegressor с логарифмом цены

Создадим и выведем итоговую таблицу.


Рисунок 42– Результат

Таблица 1. – Сравнение точности предугадывания результата анализа при различных методах машинного обучения

| Модель | R^2 | MSE | RMSE | MAE |
|---|---|---|---|---|
| LinearRegression | 0.767405 | 2.395089e+12 | 1.547608e+06 | 9.533269e+05 |
| RandomForestRegressor | 0.693706 | 3.153977e+12 | 1.775944e+06 | 7.641469e+05 |
| XGBoost | 0.713833 | 2.946730e+12 | 1.716604e+06 | 7.967934e+05 |

| | | | | |
|---|---|---|---|---|
| KNeighborsRegressor | 0.673538 | 3.361659e+12 | 1.833483e+06 | 8.300504e+05 |
| DecisionTreeRegressor | 0.544582 | 4.689548e+12 | 2.165536e+06 | 1.013780e+06 |

В результате сравнения моделей машинного обучения для прогнозирования цен на квартиры, видим следующие результаты:

1. Модель LinearRegression показала наилучший результат по коэффициенту детерминации (R^2) - 0.767405, что указывает на хорошую способность модели объяснять изменчивость данных. Однако, она имеет наибольшие значения среднеквадратичной ошибки (MSE) и среднеквадратичного отклонения (RMSE), что может указывать на более высокую дисперсию ошибок.
2. Модель RandomForestRegressor и XGBoost показали схожие результаты по R^2, MSE, RMSE и MAE, что указывает на их схожую эффективность в прогнозировании цен на квартиры. Однако, XGBoost имеет немного лучшие показатели по MSE и RMSE.
3. Модель KNeighborsRegressor показала худшие результаты по всем показателям, что указывает на ее меньшую эффективность в данной задаче.
4. Модель DecisionTreeRegressor показала самые низкие результаты по R^2 и самые высокие значения MSE, RMSE и MAE, что указывает на ее наименьшую эффективность среди рассмотренных моделей.

В целом, делаем вывод, что модель LinearRegression имеет наилучшие результаты по коэффициенту детерминации, но при этом обладает более высокой дисперсией ошибок. Модели RandomForestRegressor и XGBoost показали схожие результаты и могут быть рассмотрены в качестве альтернативы линейной регрессии. Модели KNeighborsRegressor и DecisionTreeRegressor имеют худшие результаты и могут быть менее эффективными для данной задачи.

Таблица 2. – Сравнение точности предугадывания результата анализа при различных методах машинного обучения

| Модель | R^2 | MSE | RMSE | MAE |
|---|---|---|---|---|
| LinearRegression (log) | 0.704591 | 166.282086 | 12.895041 | 10.618951 |
| RandomForestRegressor (log) | 0.758905 | 135.709173 | 11.649428 | 9.391951 |
| XGBoost (log) | 0.740416 | 146.116601 | 12.087870 | 8.453316 |
| KNeighborsRegressor (log) | 0.728156 | 153.017561 | 12.370027 | 9.819512 |
| DecisionTreeRegressor (log) | 0.648156 | 198.048780 | 14.072981 | 9.317073 |

В результате сравнения моделей машинного обучения для прогнозирования цен на квартиры с использованием логарифмической

трансформации, видим следующие результаты:

1. Модель RandomForestRegressor (log) показала наилучший результат по коэффициенту детерминации (R^2) - 0.758905, что указывает на хорошую способность модели объяснять изменчивость данных. Она также имеет наименьшие значения среднеквадратичной ошибки (MSE) и среднеквадратичного отклонения (RMSE), что указывает на ее эффективность в прогнозировании цен на квартиры.
2. Модель DecisionTreeRegressor (log) показала схожие результаты с моделью RandomForestRegressor (log) по R^2, но имеет более высокие значения MSE и RMSE.
3. Модель LinearRegression (log) показала средние результаты по R^2, MSE, RMSE и MAE, что указывает на ее умеренную эффективность в данной задаче.
4. Модель KNeighborsRegressor (log) показала худшие результаты по R^2 и имеет более высокие значения MSE и RMSE по сравнению с моделью RandomForestRegressor (log), что указывает на ее меньшую эффективность в данной задаче.
5. Модель XGBoost (log) показала самые низкие результаты по R^2 и самые высокие значения MSE и RMSE, что указывает на ее наименьшую эффективность среди рассмотренных моделей.

В целом, можно сделать вывод, что модель RandomForestRegressor (log) имеет наилучшие результаты по коэффициенту детерминации, MSE и RMSE, что делает ее наиболее подходящей для прогнозирования цен на квартиры с использованием логарифмической трансформации. Модель DecisionTreeRegressor (log) может рассматриваться в качестве альтернативы, но с несколько худшими результатами. Модели LinearRegression (log), KNeighborsRegressor (log) и XGBoost (log) имеют худшие результаты и могут быть менее эффективными для данной задачи.

## 4. Выводы

В результате проведенного исследования было установлено, что модели машинного обучения являются эффективным инструментом для прогнозирования цен на квартиры. В рамках данной работы были рассмотрены различные модели: линейная регрессия, случайный лес, XGBoost, дерево решений и метод k-ближайших соседей.

В ходе анализа было выявлено, что модель RandomForestRegressor (log) показала наилучшие результаты по коэффициенту детерминации, MSE и RMSE при использовании логарифмической трансформации данных. Это указывает на ее эффективность в прогнозировании цен на квартиры с использованием данной трансформации.

В случае отсутствия трансформации данных, можно рассмотреть модели LinearRegression, RandomForestRegressor и XGBoost. При этом линейная регрессия может иметь более высокую дисперсию ошибок, что следует учитывать при выборе модели.

Модели KNeighborsRegressor и DecisionTreeRegressor показали худшие результаты в обоих случаях и могут быть менее эффективными для данной задачи.

В целом, проведенный интеллектуальный анализ данных цен на квартиры с использованием моделей машинного обучения показал, что данный подход может быть эффективно применен для решения задачи прогнозирования цен на рынке недвижимости. Это позволяет повысить эффективность работы агентств недвижимости, оптимизировать процесс ценообразования и улучшить прогнозирование будущих тенденций на рынке.

Таким образом, в заключении можно сделать вывод о том, что интеллектуальный анализ данных цен на квартиры с использованием моделей машинного обучения является перспективным направлением для повышения эффективности работы на рынке недвижимости и может быть рекомендован для практического применения.

**Библиографический список**

1. Свидетельство о государственной регистрации программы для ЭВМ № 2023682073 Российская Федерация. Модель машинного обучения по оценке стоимости квадратного метра: № 2023680920 : заявл. 12.10.2023 : опубл. 20.10.2023 / А. В. Толмачев, О. Н. Красавина ; заявитель Федеральное государственное автономное образовательное учреждение высшего образования «Уральский федеральный университет имени первого Президента России Б.Н. Ельцина».
2. Васильченко А. М. Как проводить анализ данных при помощи python? //Инновации и инвестиции. 2023. №. 5. С. 161-165.
3. Косых Н. Е. Оценка гиперпараметров при анализе тональности русскоязычного корпуса текстов//Интеллектуальные технологии на транспорте. 2020. №. 3 (23). С. 41-44.
4. Богданов П. Ю. и др. Программные среды для изучения основ нейронных сетей //Программные продукты и системы. 2021. №. 1. С. 145-150.
5. Григорьев Е. А., Климов Н. С. Разведочный анализ данных с помощью python //E-Scio. 2020. №. 2 (41). С. 165-176.