

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

Институт информационных технологий
и анализа данных

наименование института

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

по дисциплине «Технологии обработки, анализа и визуализации данных»

Выполнил студент

ИИТм-23-1

шифр группы

подпись

Солопов Д.Д.

И.О. Фамилия

Проверил

Григорьев С.В.

подпись

И.О. Фамилия

Отчет защищен с оценкой _____

Иркутск 2024 г

Содержание

Введение	3
1. Анализ данных методом одномерной линейной регрессии.....	4
2 Анализ данных методом дерева решений.....	16
3. Анализ данных с помощью метода k-ближайших соседей.....	22
4. Анализ данных методом опорных векторов	28
Выводы	32
Список использованных источников	33
Приложение А Ссылка на исходный код.....	34

Введение

Цель работы заключается в изучении способов применения методов анализа данных из заданного набора, а также известных алгоритмов на их основе, для решения предсказательных задач, в том числе за счёт построения и использования классификаторов, регрессионных или кластерных моделей.

Задачи:

1. Сформулировать задачу для анализа данных из данного набора и решить ее методом одномерной линейной регрессии;
2. Сформулировать задачу для анализа данных из данного набора и решить ее методом дерева решений;
3. Сформулировать задачу для анализа данных из данного набора и решить ее методом k-ближайших соседей или k-средних;
4. Сформулировать задачу для анализа данных из данного набора и решить ее методом опорных векторов.

1. Анализ данных методом одномерной линейной регрессии

Сформулированная задача для анализа данных: предсказать цену недвижимости за единицу площади в зависимости от расстояния до ближайшей станции метро с помощью одномерной линейной регрессии.

Для начала импортируем необходимые библиотеки для последующего использования в среде Google Colab:

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

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression

%matplotlib inline
```

Теперь загрузим и предварительно ознакомимся с датасетом:

```
# Загрузка набора данных
df = pd.read_csv("/content/Real Estate Dataset.csv")

# Просмотр основной информации о фрейме
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 414 entries, 0 to 413
Data columns (total 8 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   number                                     414 non-null    int64
1   transaction date                           414 non-null    float64
2   house age                                  414 non-null    float64
3   distance to the nearest MRT station        414 non-null    float64
4   number of convenience stores               414 non-null    int64
5   latitude                                    414 non-null    float64
6   longitude                                   414 non-null    float64
7   house price of unit area                   414 non-null    float64
dtypes: float64(6), int64(2)
memory usage: 26.0 KB
```

Рисунок 1 – Основная информация о наборе данных

Набор данных содержит информацию о цене недвижимости за единицу площади, которая обладает определёнными характеристиками, влияющими на эту цену, в том числе:

1. **house age** – число лет недвижимости;
2. **distance to the nearest MRT station** – расстояние до ближайшей станции метро;

3. **number of convenience stores** – количество расположенных по близости круглосуточных магазинов

Поскольку исходя из сформулированной задачи мы будем предсказывать цену за единицу площади, то необходимо заранее определиться что данный столбец представляет из себя значения Y (то, что будет предсказываться), а расстояние до ближайшей станции метро – X .

В наборе данных находятся только числовые значения, поэтому никаких дополнительных преобразований не требуется. Количественные значения определены с помощью типа данных `int64`, а вещественные – с помощью `float64`.

Определим выбросы в исследуемом наборе данных с помощью диаграммы “ящик с усами” (boxplot), предварительно построив график QQ-Plot (квантиль-квантиль график) с помощью следующего программного кода:

```
import statsmodels.api as sm
import pylab as py
import scipy.stats as stats
Y_label = 'house price of unit area'

#QQ-Plot
stats.probplot(df[Y_label], dist='norm', plot=py)
py.show()
```

QQ-plot (квантиль-квантиль график) — это инструмент, который используется для оценки сходства распределения одной числовой переменной с нормальным распределением или между распределениями двух числовых переменных.

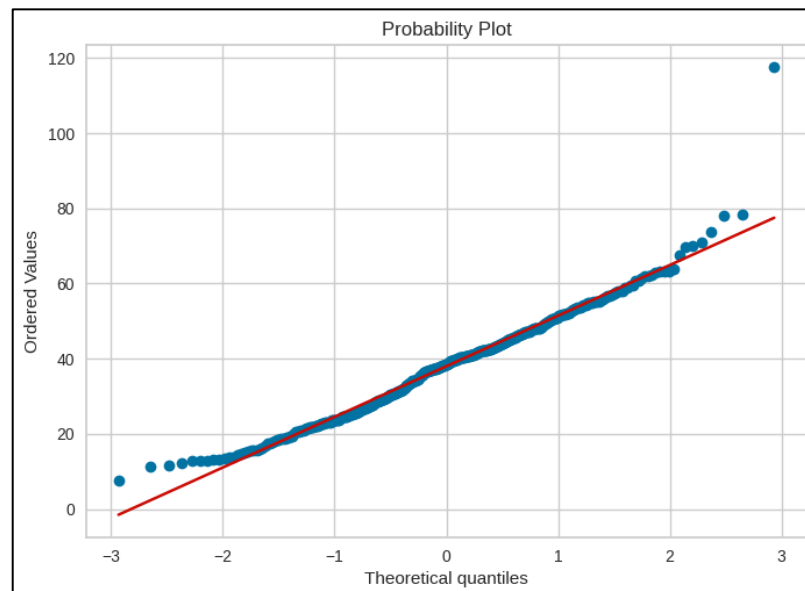


Рисунок 2 – Результат построения квантиль-квантиль графика по цене на недвижимость за единицу площади

Как видно из квантиль-квантиль графика имеются значения со слишком большим отклонением от линии лучшего соответствия (обозначена красным цветом).

Теперь построим диаграмму `boxplot`, с помощью следующего программного кода:

```
df[[Y_label]].boxplot(figsize=(13, 8))
```

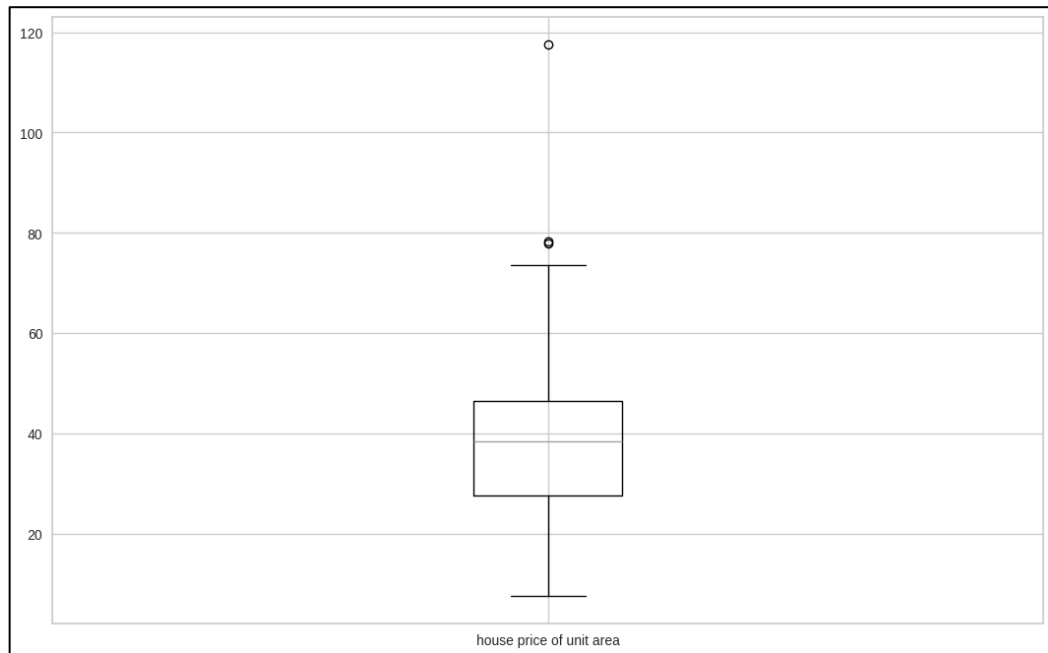


Рисунок 3 – Диаграмма `boxplot` для цены на недвижимость за единицу площади

Как видно из диаграммы `boxplot` по цене за единицу площади имеются выбросы, которые могут в дальнейшем повлиять на предсказание цены. Необходимо удалить записи с такими выбросами.

Для удаления выбросов будет использован метод межквартильного размаха (IQR). Вычислим границы IQR и отсеим выбросы на основе этих границ.

определяется как среднее число между наименьшим числом и медианой набора данных

Для начала определим среднее число между наименьшим числом и медианой набора данных (Q1) и среднее значение между медианой и самым высоким значением в наборе данных (Q2):

```
Q1 =df[Y_label].quantile(0.25)
Q3 =df[Y_label].quantile(0.75)
```

При вычислении получаем, что $Q1 = 27.7$, а $Q2 = 46.6$

Теперь вычислим значение IQR, который представляет собой разницу между третьим и первым квартилем:

```
IQR = Q3 - Q1
IQR
```

Значение IQR будет равно 18.9.

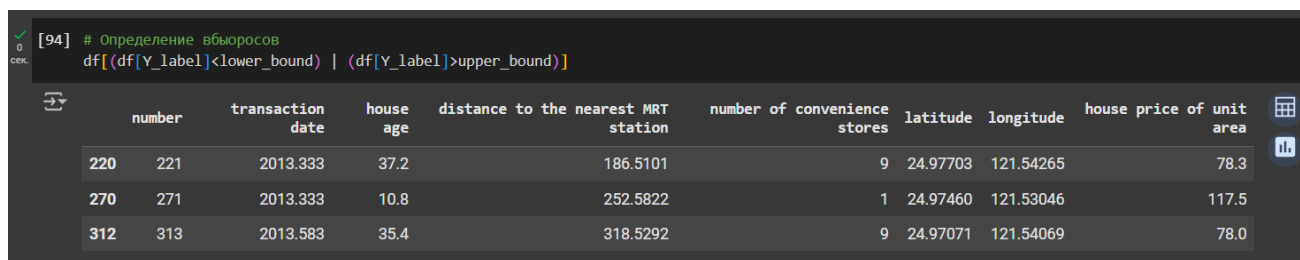
Теперь находим нижнюю и верхнюю границу для отсеивания выбросов в исходном наборе данных:

```
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

После вычисления lower_bound будет равен -0.65, а upper_bound равен 74.95.

Теперь можно рассмотреть конкретные значения выбросов из датафрейма Pandas с помощью выполнения операции фильтра:

```
df[(df[Y_label]<lower_bound) | (df[Y_label]>upper_bound)]
```



```
[94] # Определение выбросов
df[(df[Y_label]<lower_bound) | (df[Y_label]>upper_bound)]
```

	number	transaction date	house age	distance to the nearest MRT station	number of convenience stores	latitude	longitude	house price	unit area
220	221	2013.333	37.2	186.5101	9	24.97703	121.54265	78.3	
270	271	2013.333	10.8	252.5822	1	24.97460	121.53046	117.5	
312	313	2013.583	35.4	318.5292	9	24.97071	121.54069	78.0	

Рисунок 4 – Вывод выбросов в виде значений из датафрейма

Как видно из рисунка 4 выбросов в наборе данных всего 3. Данные выбросы необходимо удалить с помощью исключения их из всех значений набора данных следующим образом:

```
df = df[~((df[Y_label] < (lower_bound)) | (df[Y_label] > (upper_bound)))]
```

После этого на диаграмме “ящик с усами” и квантиль-квантиль графике никаких больших отклонений не будет (см. рис. 5), а из набора данных будет на 3 записи меньше – вместо 414 записей будет 411.

Теперь можно продолжать работу с обработанным набором данных.

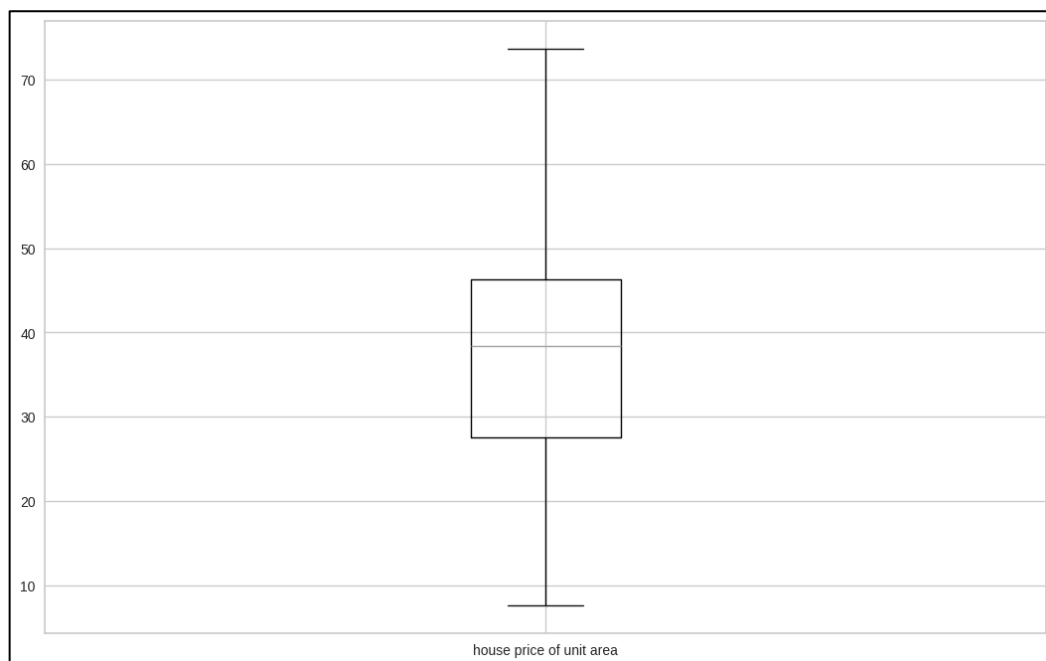


Рисунок 5 - Диаграмма boxplot для цены на недвижимость за единицу площади после удаления выбросов

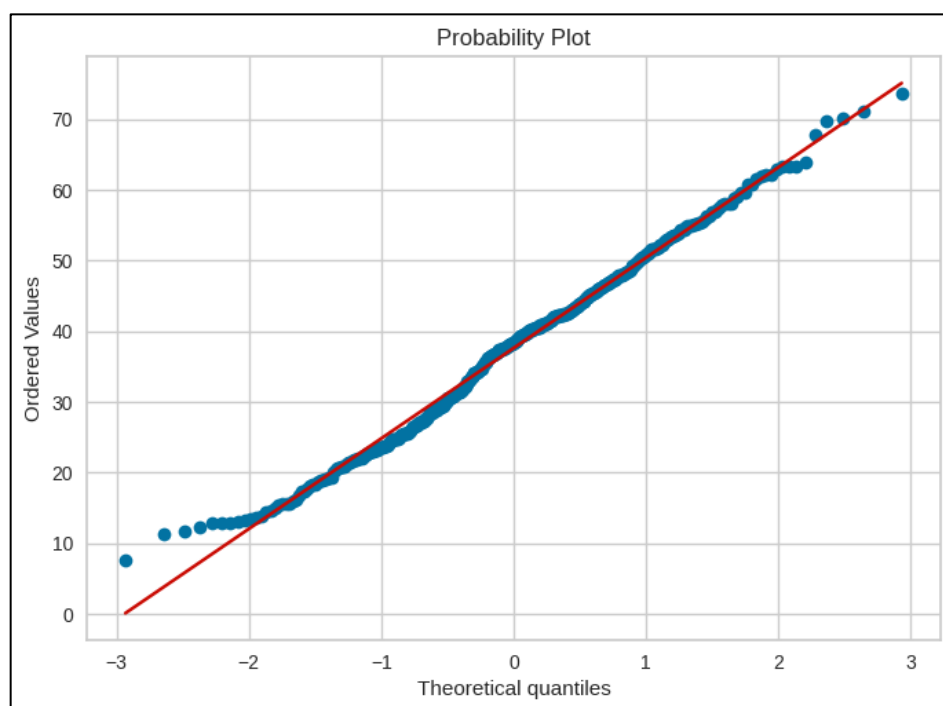


Рисунок 6 - Результат построения квантиль-квантиль графика по цене на недвижимость за единицу площади

Начнём с вывода тепловой карты для определения корреляции между разными столбцами из набора данных:

```
sns.heatmap(df.corr(), annot=True, cmap='crest')
```

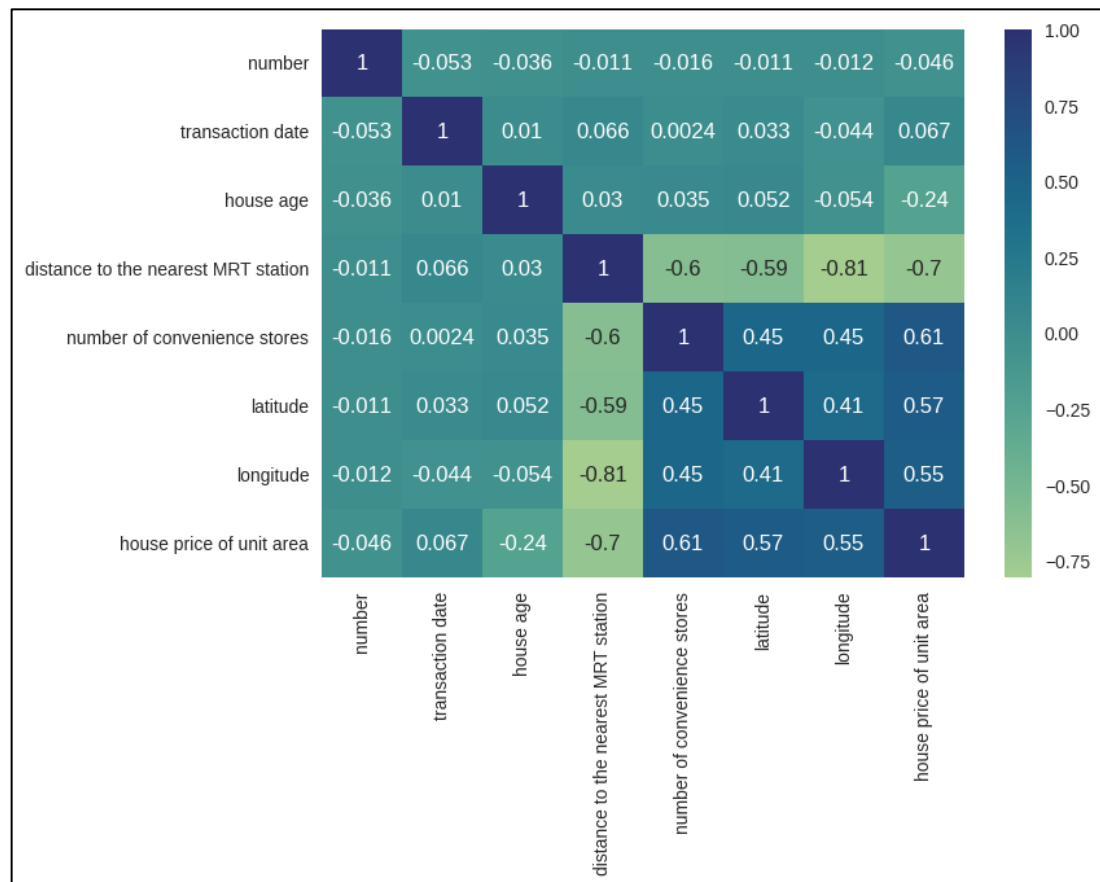


Рисунок 7 – Тепловая карта коэффициентов корреляции

Из тепловой карты (см. рис. 7) можно отметить, что столбец distance to the nearest MRT station (X) и house price of unit area (Y) имеют отрицательный коэффициент корреляции (-0.7), что означает, что расстояние до ближайшей станции метро отрицательно связано с размером цены за единицу площади жилья. Иными словами, чем ближе жилой район находится к станции метро, тем выше цена на дом, и наоборот. Данная корреляция сильно выражена, а поэтому для более лучшего предсказания регрессионной модели цены целесообразно использовать именно такую комбинацию X и Y.

Теперь построим модель одномерной линейной регрессии, но перед этим шагом необходимо определить тренировочный и тестовый набор данных для прогона через модель линейной регрессии.

Для начала отдельно выделим данные из столбцов в переменные X и y:

```
Y_label = "house price of unit area"
X_label = "distance to the nearest MRT station"

X = df[X_label]
y = df[Y_label]
```

Теперь с помощью функции `train_test_split` из пакета `sklearn.model_selection` разделим данные на обучающий и тестовый набор из `X` и `y`:

```
# Делим датасет на обучающий и тестовый наборы
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.3,
    random_state=101
)

X_train = X_train.to_numpy().reshape(-1, 1)
X_test = X_test.to_numpy().reshape(-1, 1)

y_train = y_train.to_numpy()
y_test = y_test.to_numpy()
```

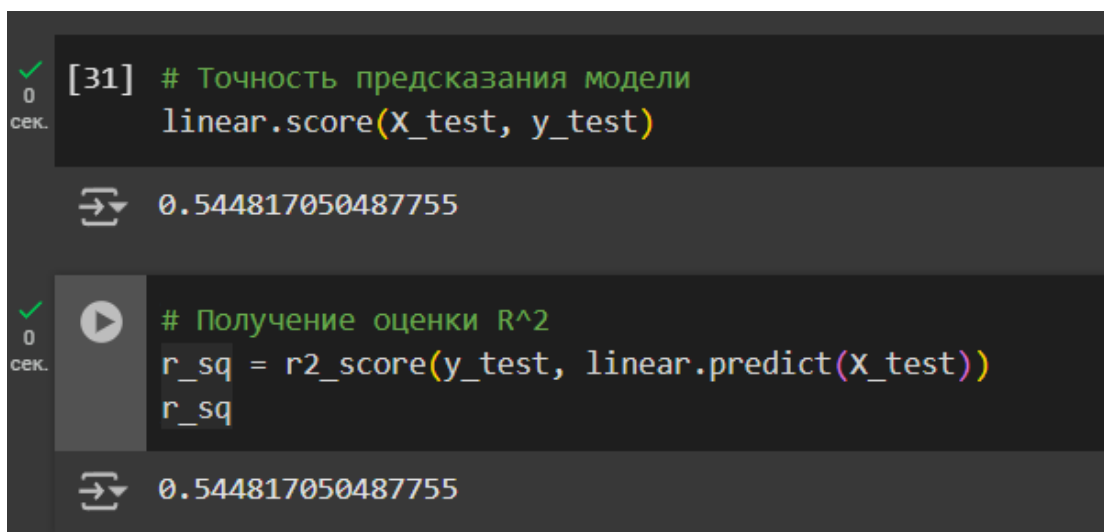
Теперь создаём модель одномерной линейной регрессии и обучаем её на обучающем наборе данных:

```
# Создание модели линейной регрессии
linear = LinearRegression()

# Обучение модели
linear.fit(X_train, y_train)
```

Теперь оценим результат предсказания модели на тестовом наборе данных и получим оценку R^2 :

```
# Получение оценки  $R^2$ 
r_sq = r2_score(y_test, linear.predict(X_test))
r_sq
```



```
[31] # Точность предсказания модели
linear.score(X_test, y_test)

0.544817050487755

# Получение оценки  $R^2$ 
r_sq = r2_score(y_test, linear.predict(X_test))
r_sq

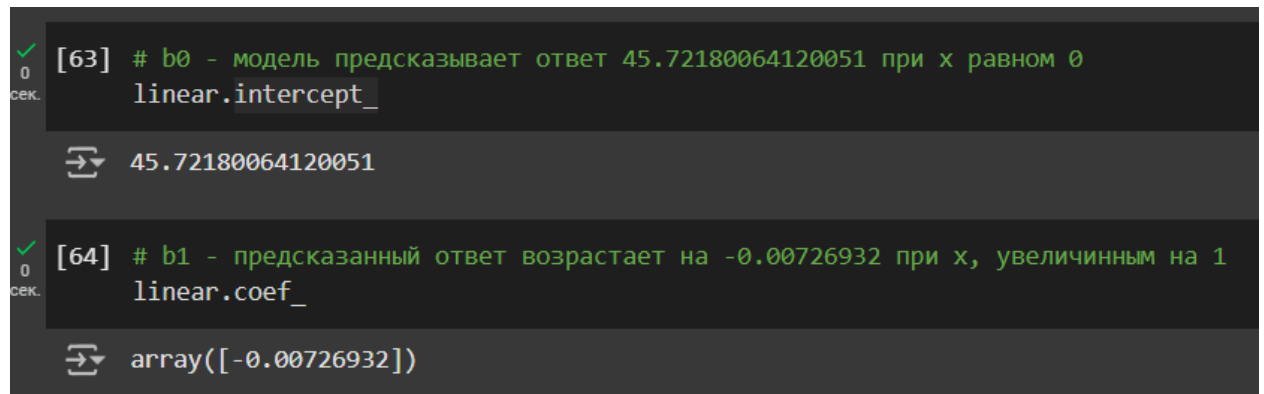
0.544817050487755
```

Рисунок 8 – Результат оценки точности модели

Точность предсказания модели на тестовом наборе данных составляет 0.544, что является довольно неплохим результатом учитывая, что весь набор данных состоит из 411-ой записи.

Теперь выведем оптимальные значения весов b_0 и b_1 , которые были вычислены моделью:

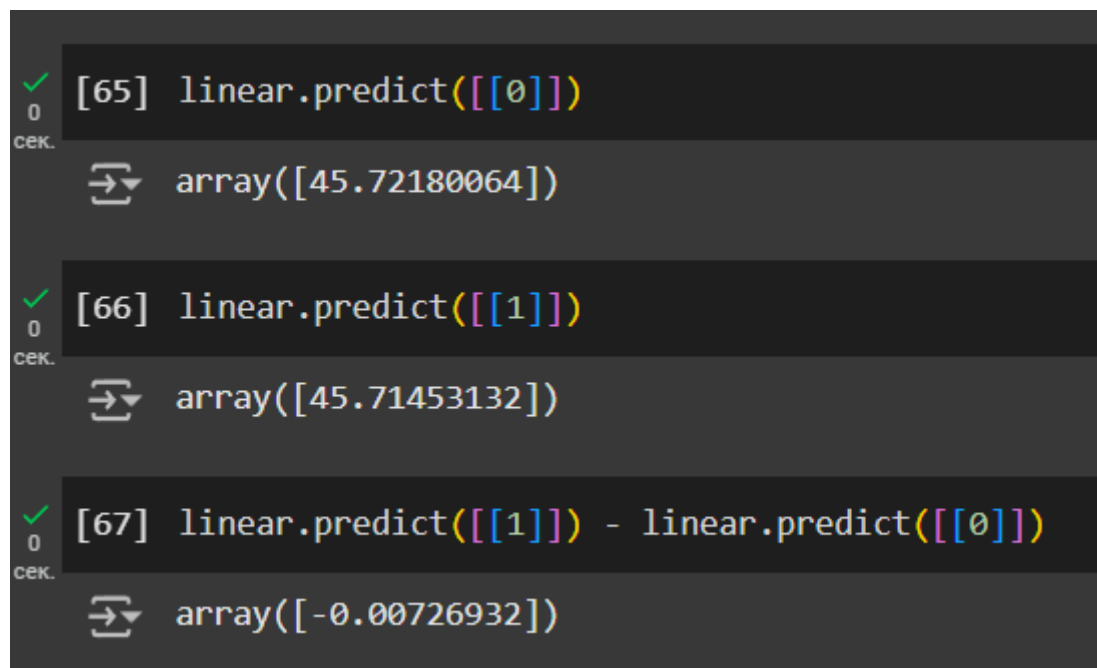
```
linear.intercept_  
linear.coef_
```



```
[63] # b0 - модель предсказывает ответ 45.72180064120051 при x равном 0  
linear.intercept_  
⇒ 45.72180064120051  
  
[64] # b1 - предсказанный ответ возрастает на -0.00726932 при x, увеличинным на 1  
linear.coef_  
⇒ array([-0.00726932])
```

Рисунок 9 – Вывод значения весов b_0 и b_1

Исходя из вычисленного веса b_0 определяем, что модель предсказывает ответ 45.7218 при x равном 0. При увеличении x на 1, предсказанный ответ будет возрастать на -0.00726932 (см. рис. 10).



```
[65] linear.predict([[0]])  
⇒ array([45.72180064])  
  
[66] linear.predict([[1]])  
⇒ array([45.71453132])  
  
[67] linear.predict([[1]]) - linear.predict([[0]])  
⇒ array([-0.00726932])
```

Рисунок 10 – Вывод предсказаний модели для $x = 0$ и $x = 1$, а также разности этих предсказаний

Выведем первые 10 результаты предсказаний на тестовых данных:

```
# Предсказание цены на недвижимость по числу лет недвижимости  
y_pred = linear.predict(X_test)
```

```
# Вывод 10-ти предсказанных цен
pd.DataFrame({'Test': y_test, 'Pred': y_pred }).head(10)
```

	Test	Pred
0	36.5	42.289411
1	40.6	42.133248
2	39.1	44.576117
3	36.9	44.586011
4	35.3	42.495660
5	43.5	43.729901
6	30.5	42.868984
7	42.8	44.959902
8	51.0	43.618605
9	50.0	45.064246

Рисунок 11 – Первые 10 предсказаний по тестовым данным

Как видно из рисунка 10 предсказанные значения модели имеют большую ошибку в результатах. Лишь для 5-го случая удалось по сравнению с остальными 9-ю случаями хорошо предсказать цену единицы площади недвижимости (43.5 – тестовая выборка, 43.7299 – результат предсказания).

Для визуального сравнения предсказанных значений с тестовыми данными сформируем график:

```
x_ax = range(len(y_test))
plt.plot(x_ax, y_test, label="Тестовые значения")
plt.plot(x_ax, y_pred, label="Предсказанные значения")
plt.title("Тестовые и предсказанные значения")

plt.xlabel('X')
plt.ylabel('Y')

plt.legend(loc='best', fancybox=True, shadow=True)
plt.grid(True)
plt.show()
```



Рисунок 12 – График тестовых и предсказанных значений

На рисунке 12 видно, что предсказанные значения на разных участках частично или полностью совпадают с тестовыми значениями, однако есть промежутки с большими разрывами. Несмотря на то, что в качестве X был выбран столбец, который наиболее коррелирует с y (расстояние до ближайшей станции метро) с помощью одномерной линейной регрессии удалось добиться точности только в 0.544.

Теперь наиболее распространённые оценки для регрессионных задач:

1. Средняя абсолютная погрешность (MAE) - это метрика, которая измеряет среднее абсолютное отклонение между фактическими и прогнозируемыми значениями в наборе данных;
2. Среднеквадратичная ошибка (MSE) - это среднее значение квадратов ошибок;
3. Корень из среднеквадратичной ошибки (RMSE) - это квадратный корень из среднего значения квадратной ошибки.

Для вычисления MAE используется следующая формула:

$$\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Для вычисления MSE используется следующая формула:

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Для вычисления RMSE используется следующая формула:

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Для вычисления регрессионных метрик используется следующий программный код:

```
# Вычисляем регрессионные метрики
MAE = metrics.mean_absolute_error(y_test, y_pred)
MSE = metrics.mean_squared_error(y_test, y_pred)
RMSE = np.sqrt(MSE)

pd.DataFrame([MAE, MSE, RMSE], index=['MAE', 'MSE', 'RMSE'],
columns=['Metrics'])
```

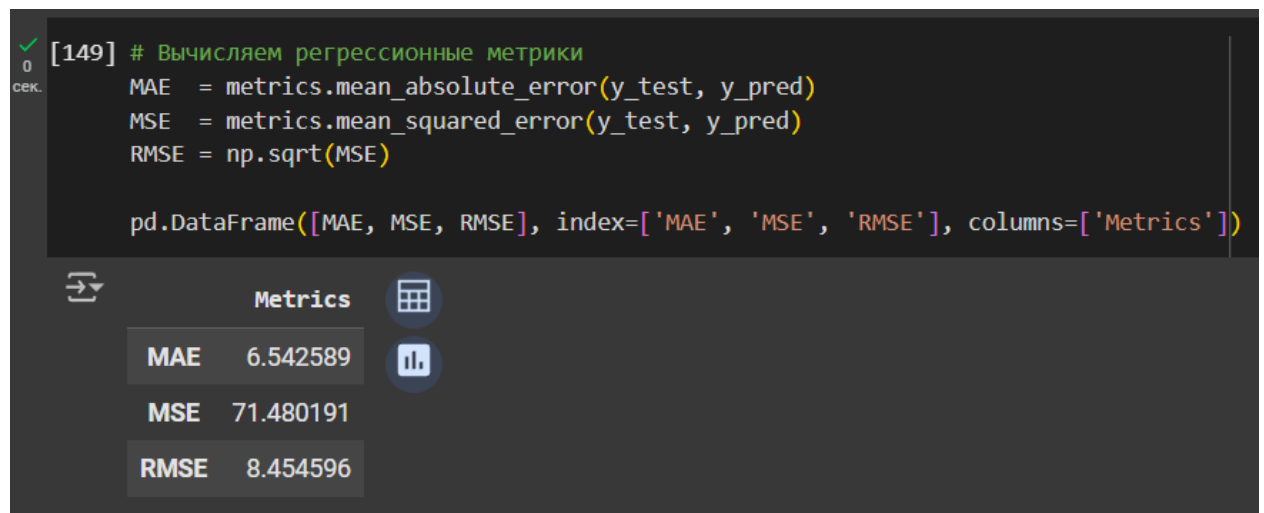


Рисунок 13 – Результат вычисления метрик

После вычисления получаем, что MAE равно 6.542589, MSE равно 71.480191, а RMSE равно 8.454596.

Построим диаграмму ошибок, чтобы визуально оценить их объём, с помощью следующего программного кода:

```
# Вычисляем разницу между тестовыми значениями и
предсказанными
test_diff = (y_test - y_pred)

# Строим диаграмму ошибок
pd.DataFrame({'Error Values': (test_diff)}).hvplot.kde()
```

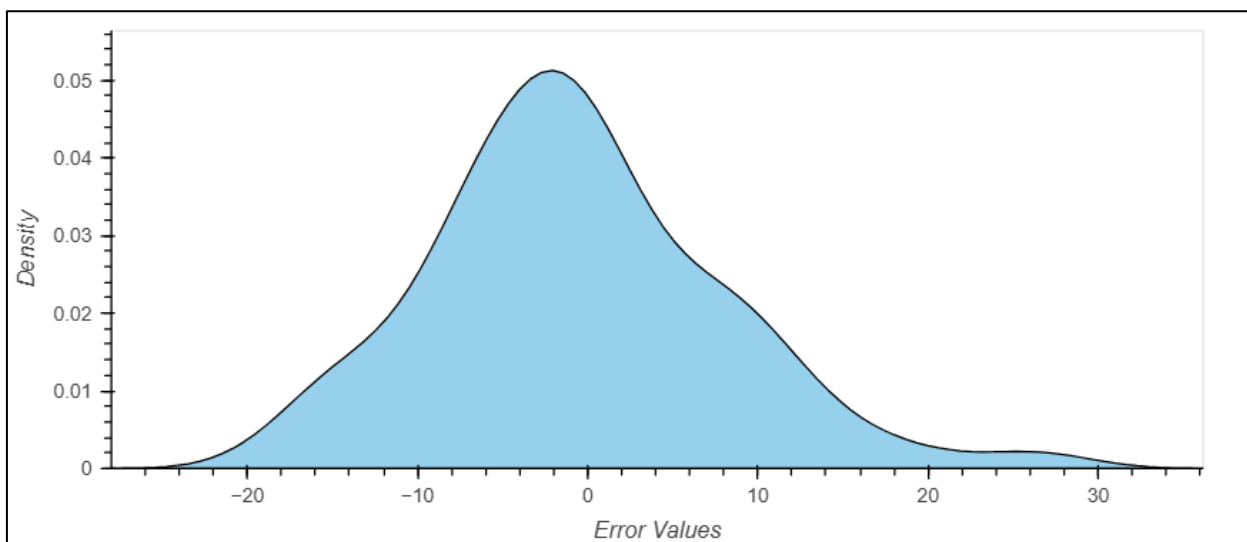


Рисунок 14 – Диаграмма разницы между тестовыми данными и предсказанными

Как видно из диаграммы на рисунке 14 плотность ошибок возрастает в промежутке от 0 до -2, и уменьшается с промежутка от -2 до 32.

Исходя из диаграммы и вычисленных числовых метрик регрессионной модели можно заключить, что для более точных предсказываний цены по переменной X необходим больший набор данных, т.к. на данный момент предсказания не очень точны, несмотря на очевидную корреляцию между X и y , выявленная по тепловой карте.

2 Анализ данных методом дерева решений

В рамках данного метода формулировка задачи будет эквивалентна предыдущей, а именно: предсказать цену недвижимости за единицу площади в зависимости от расстояния до ближайшей станции метро с помощью одномерной линейной регрессии.

После решения предыдущей задачи с новым методом можно оценить эффективность каждого отдельного метода. В данном случае сравним насколько лучше модель дерева решений справится с предсказанием цены на единицу площади недвижимости, чем справился с этой задачей метод одномерной линейной регрессии.

Для начала импортируем библиотеку для создания модели дерева решений:

```
from sklearn.tree import DecisionTreeRegressor
```

Также определим обучающий и тестовый набор данных:

```
X_label_tree = "distance to the nearest MRT station"

X_tree = df[X_label_tree]
y_tree = df[Y_label]

# Делим датасет на обучающий и тестовый наборы
X_train_tree, X_test_tree, y_train_tree, y_test_tree =
train_test_split(
    X_tree,
    y_tree,
    test_size=0.3,
    random_state=101
)

X_train_tree = X_train_tree.to_numpy().reshape(-1, 1)
X_test_tree = X_test_tree.to_numpy().reshape(-1, 1)

y_train_tree = y_train_tree.to_numpy()
y_test_tree = y_test_tree.to_numpy()
```

Теперь определим модель на основе регрессора DecisionTreeRegressor и обучим данную модель на обучающей выборке:

```
# Создание модели регрессора на основе дерева решений
decisionTree = DecisionTreeRegressor(random_state=1)

# Обучение модели на тренировочном наборе данных
decisionTree.fit(X_train_tree, y_train_tree)
```

И оценим точность предсказания модели на тестовом наборе данных:

```
decisionTree.score(X_test_tree, y_test_tree)
```



```
✓ [167] # Оценка точности предсказаний модели
0 сек. decisionTree.score(X_test_tree, y_test_tree)
⇒ 0.5996171921054466
```

Рисунок 15 – Оценка точности предсказания модели

Как видно из рисунка 15 точность предсказания модели равняется 0.599, что уже лучше результата, который был получен при обучении модели методом одномерной линейной регрессии.

Выведем 10-ть первых предсказанных новой моделью значений:

```
# Предсказание цены на недвижимость по числу лет недвижимости
y_pred_tree = decisionTree.predict(X_test_tree)

# Вывод 10-ти предсказанных цен
pd.DataFrame({'Test': y_test_tree, 'Pred': y_pred_tree
}).head(10)
```

```
✓ # Предсказание цены на недвижимость по числу лет недвижимости
0 сек. y_pred_tree = decisionTree.predict(X_test_tree)

# Вывод 10-ти предсказанных цен
pd.DataFrame({'Test': y_test_tree, 'Pred': y_pred_tree }).head(10)
```

	Test	Pred
0	36.5	36.200000
1	40.6	41.180000
2	39.1	41.500000
3	36.9	41.500000
4	35.3	42.600000
5	43.5	45.400000
6	30.5	7.600000
7	42.8	52.680000
8	51.0	50.744444
9	50.0	58.144444

Рисунок 16 – Первые 10-ть предсказанных моделью значений

Общая картина теперь выглядит немного лучше. Есть как минимум 4 значения, которые очень близки друг к другу (их номера: 0, 1, 2, 8). Это уже лучше тех результатов, которые были получены для предсказания первых 10-ти значений с линейным регрессором.

Теперь вычислим метрики:

```
y_pred_tree = decisionTree.predict(X_test_tree)
```

```

# Вычисляем ошибку
MAE = metrics.mean_absolute_error(y_test_tree, y_pred_tree)
MSE = metrics.mean_squared_error(y_test_tree, y_pred_tree)
RMSE = np.sqrt(MSE)

pd.DataFrame([MAE, MSE, RMSE], index=['MAE', 'MSE', 'RMSE'],
columns=['Metrics'])

```

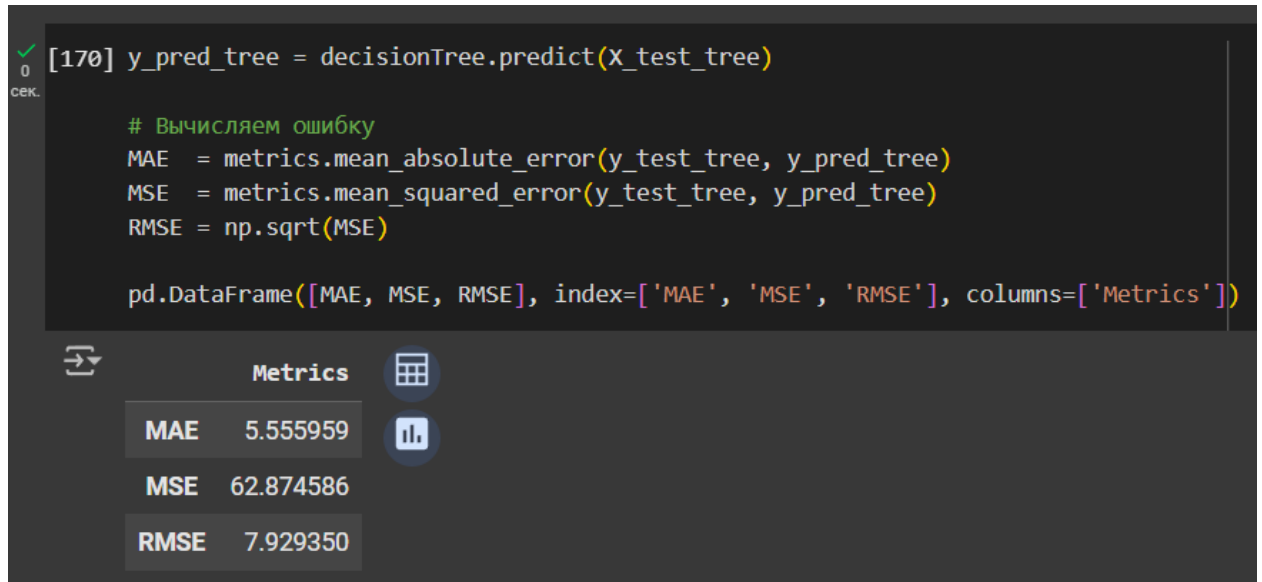


Рисунок 17 – Результат вычисления метрик

MAE, MSE и RMSE уменьшились благодаря регрессору дерева решений, что свидетельствует о лучшей предсказательной способности модели.

Выведем график предсказанных и тестовых значений:

```

x_ax = range(len(y_test_tree))
plt.plot(x_ax, y_test_tree, label="Тестовые значения")
plt.plot(x_ax, y_pred_tree, label="Предсказанные значения")
plt.title("Тестовые и предсказанные значения")

plt.xlabel('X')
plt.ylabel('Y')

plt.legend(loc='best', fancybox=True, shadow=True)
plt.grid(True)
plt.show()

```

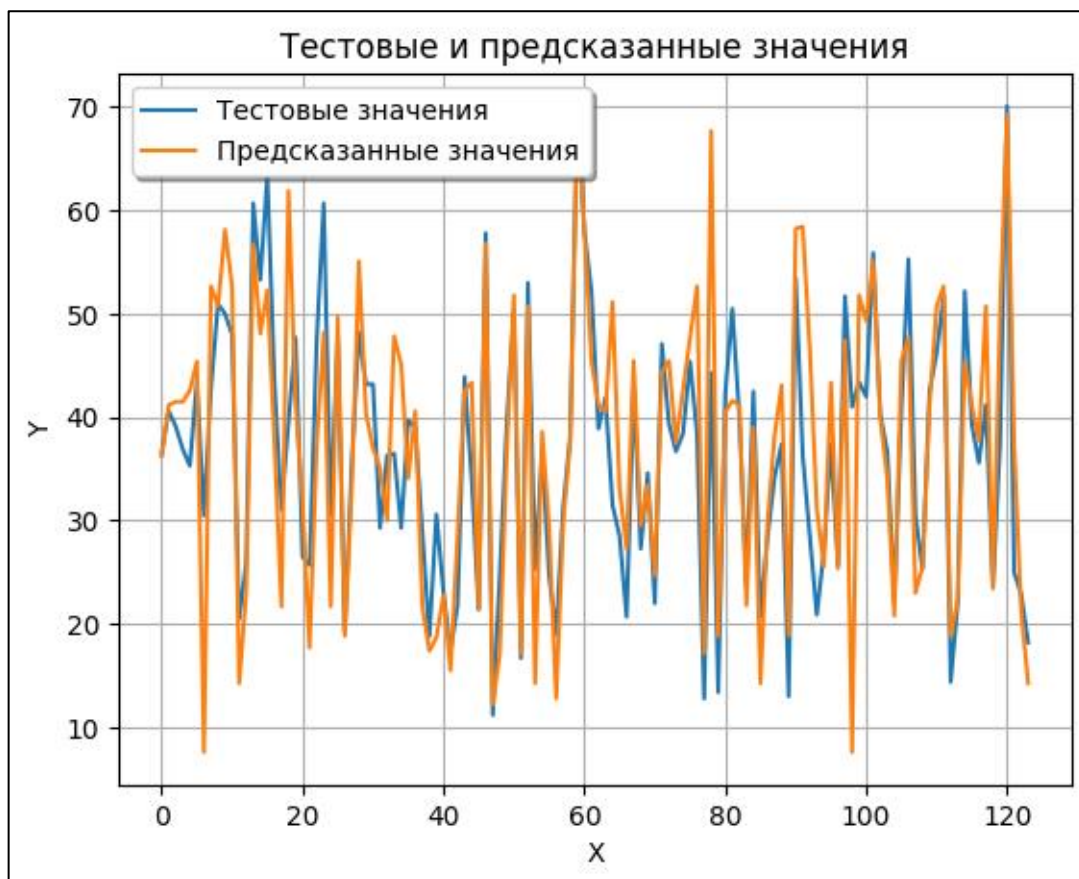


Рисунок 18 – Тестовые и предсказанные значения (метод дерева решений)

Как видно из диаграммы, представленной на рисунке 18, совпадений между предсказанными и тестовыми данными стало намного больше, по сравнению с результатами предсказания модели линейного регрессора.

Выведем график соотношения расстояния к стоимости единицы площади недвижимости:

```
X_grid = np.arange(min(X_tree), max(X_tree), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))

plt.scatter(X, y, color = 'red')

plt.plot(X_grid, decisionTree.predict(X_grid), color =
'blue')

plt.title('Соотношения расстояния к стоимости недвижимости
(Decision Tree Regression)')
plt.xlabel('Расстояние до станции метро')
plt.ylabel('Стоимость недвижимости')

plt.show()
```

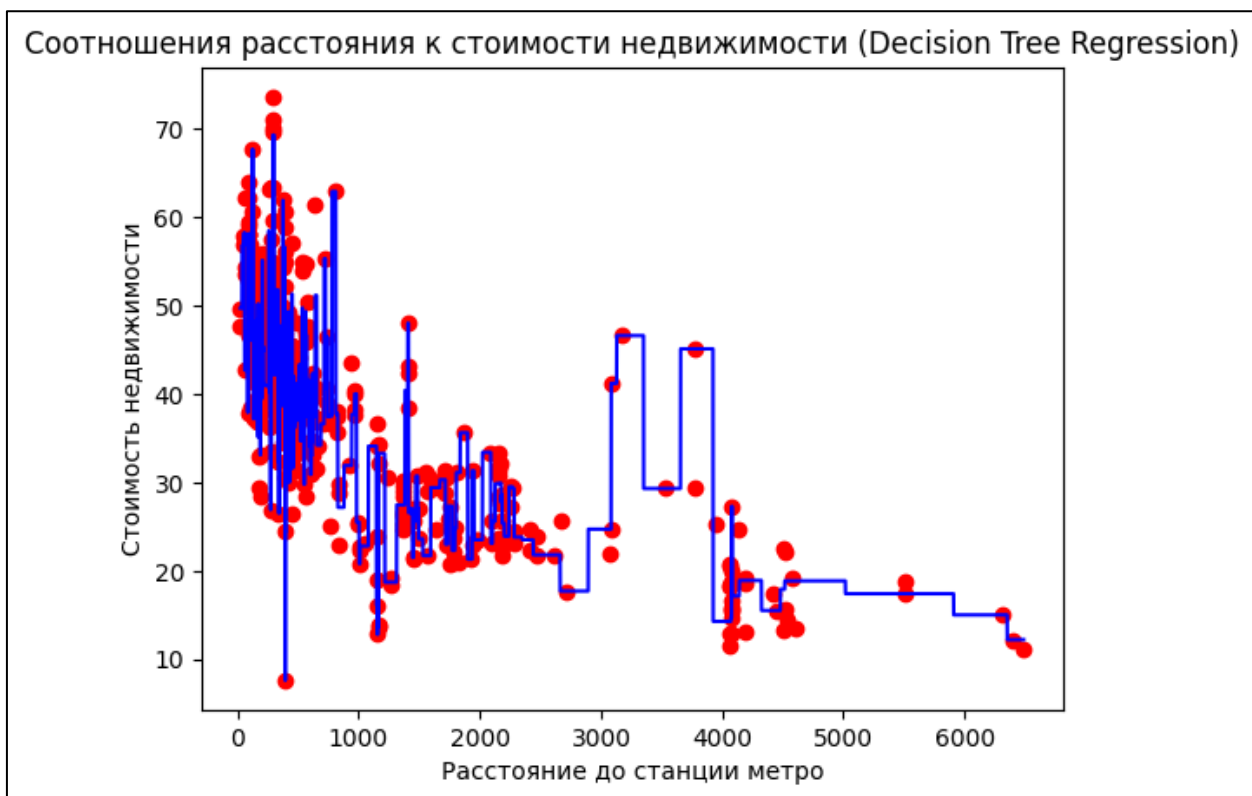


Рисунок 19 – График соотношения расстояния – стоимости

Также с помощью пакета `graphviz` можно вывести схему построенного дерева решений на основе регрессора `DecisionTreeRegressor`:

```
from IPython.display import SVG
from graphviz import Source
from sklearn import tree

graph = Source(tree.export_graphviz(decisionTree,
out_file=None, feature_names=['House prices']))
graph.format = 'png'
graph.render('dtree_render', view=True)
```

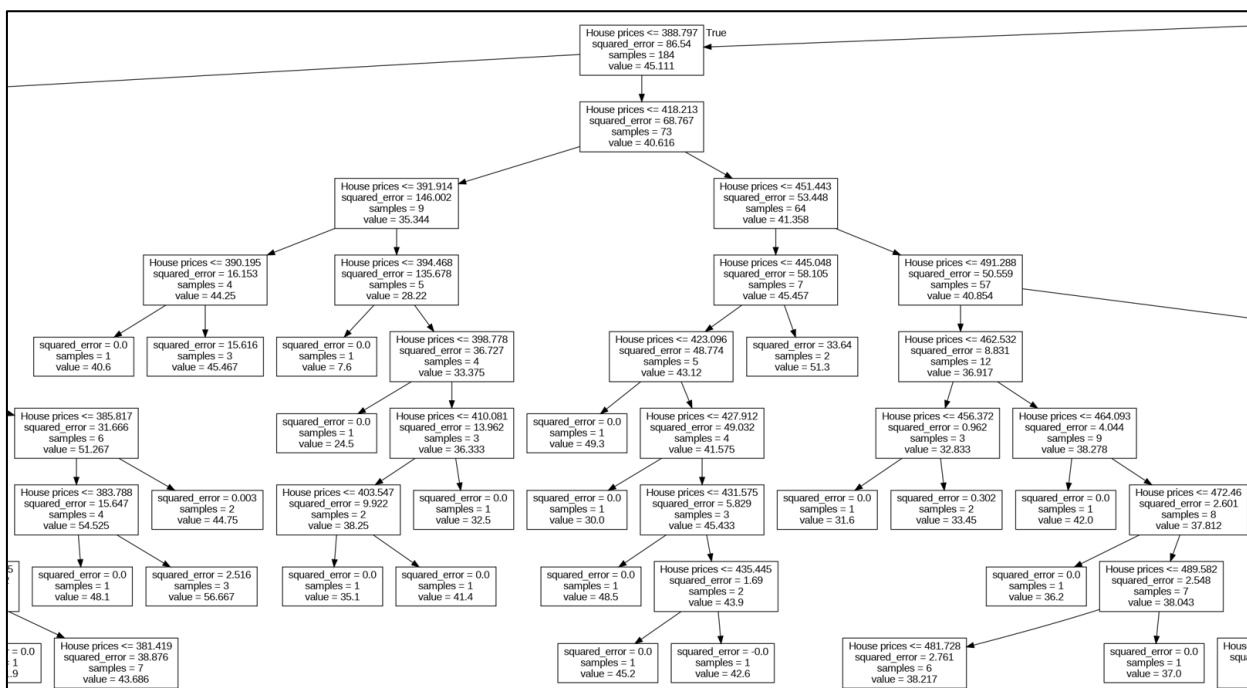


Рисунок 20 – Часть схемы дерева решений

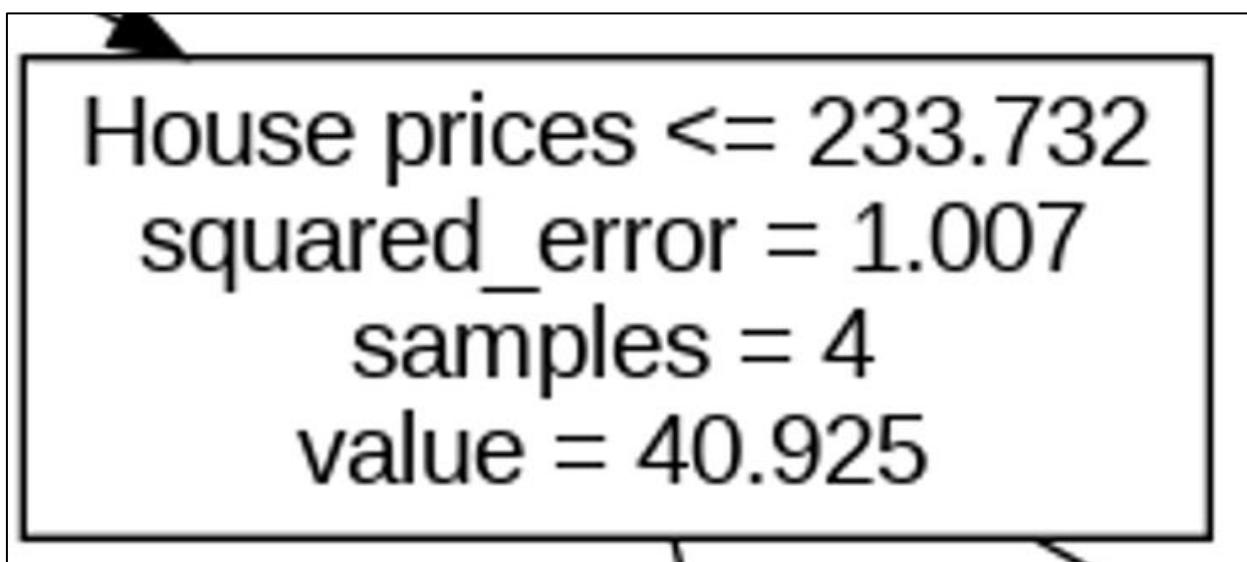


Рисунок 21 – Один из элементов схемы дерева решений

Вся схема дерева решений (см. рис. 20) состоит из элементов (см. рис. 21), которые содержат цену, квадратическую ошибку, число образцов и значение. Чем дальше схема распространяется (в ширину и вглубь) тем меньше становится квадратичная ошибка и цена.

Таким образом можно заключить, что метод дерева решений лучше справился с поставленной задачей, чем метод одномерной линейной регрессии.

3. Анализ данных с помощью метода k-ближайших соседей

Сформулированная задача для анализа данных: определить различные группы (кластеры) недвижимости относительно географического расположения в пространстве используя метод k-ближайших соседей.

Поскольку в исследуемом наборе данных присутствуют географические координаты `latitude` и `longitude`, то сформулированная задача для анализа обоснована.

Для начала определим новую переменную `location`, в которой будет содержаться массив массивов координат недвижимости:

```
location = df.iloc[:, [5, 6]].values
```

Теперь визуализируем первоначальное положение недвижимости в пространстве по координатам, назначив оси X значения `latitude`, а оси Y - `longitude`:

```
# Визуализация местоположения недвижимости
plt.scatter(location[:,0], location[:,1], marker = "x", color
= 'purple', s = 60)

plt.xlabel('latitude')
plt.ylabel('longitude')

plt.show()
```

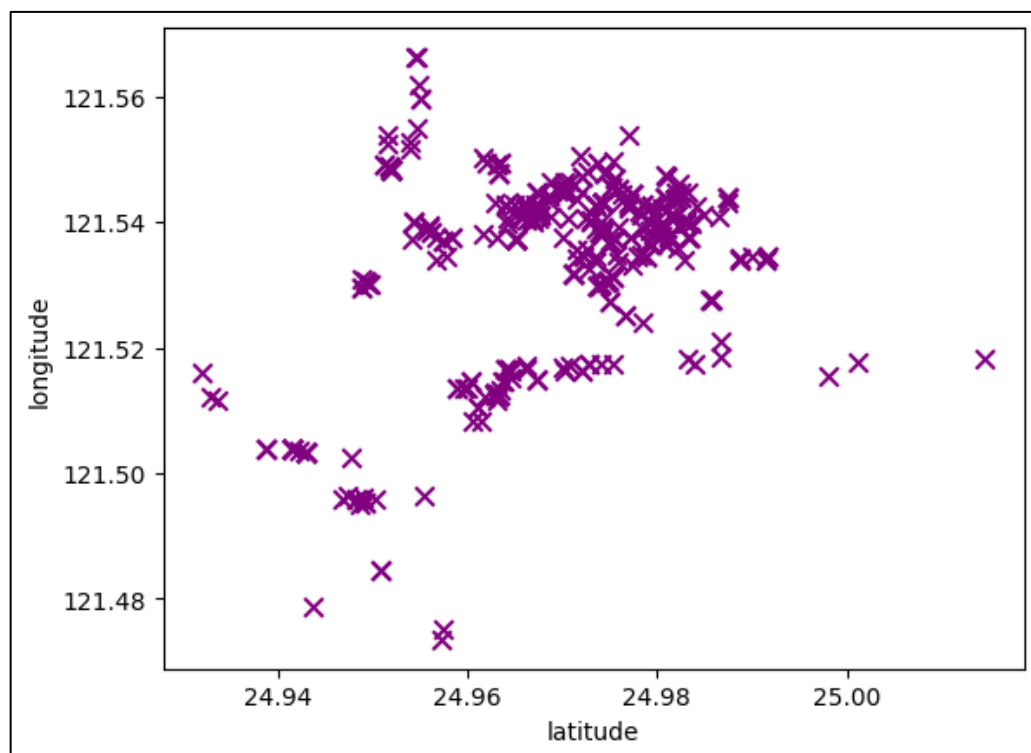


Рисунок 22 – График положения недвижимости в пространстве по координатам

Теперь с помощью локтевого метода определим оптимальное число кластеров:

```
# Поиск оптимального числа кластеров с помощью локтевого
метода
from sklearn.cluster import KMeans

wcss = []
target_range = range(1, 11)

for i in target_range:
    kmeans = KMeans(n_clusters = i, init = 'k-means++',
random_state=0)
    kmeans.fit(location)
    wcss.append(kmeans.inertia_)

plt.plot(target_range, wcss)
plt.title('Локтевой метод')

plt.xlabel('Номер кластера')
plt.ylabel('WCSS')

plt.savefig('number_of_clusters.png')
plt.show()
```

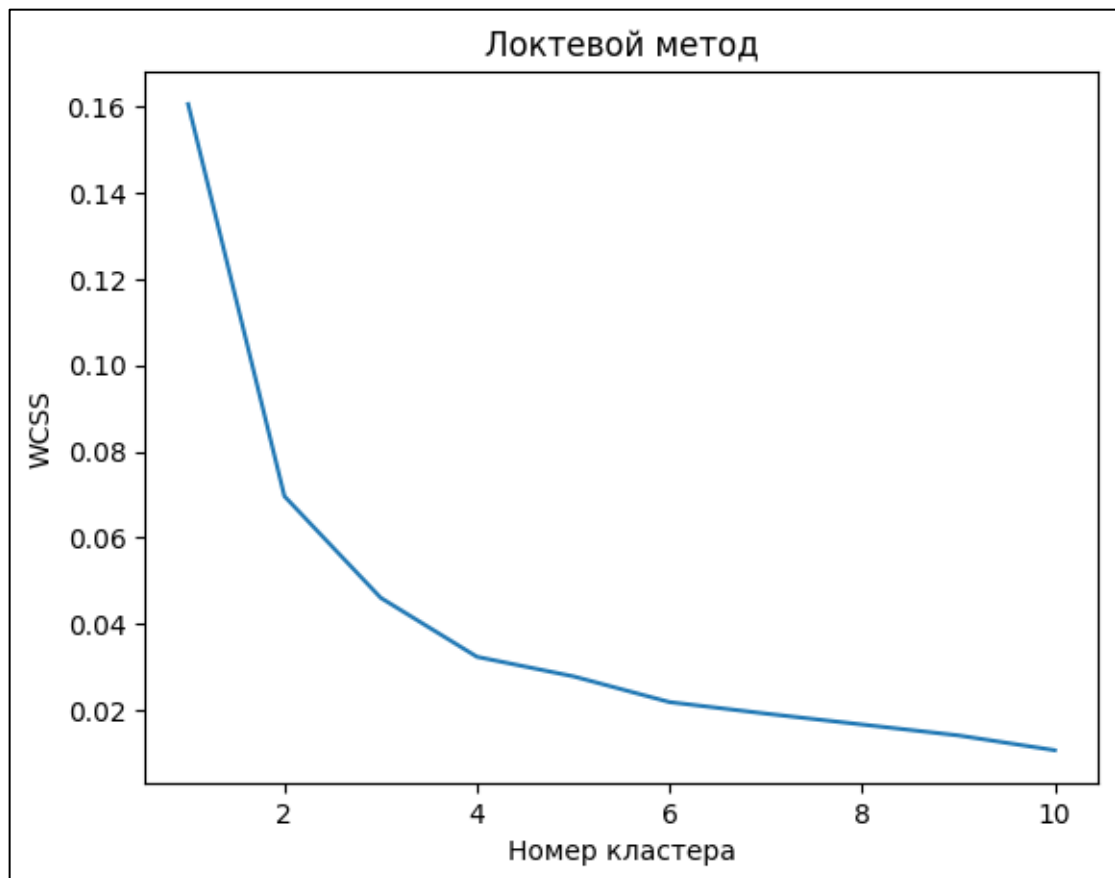


Рисунок 23 – График зависимости WCSS от номера кластера

В качестве оптимального числа кластеров возьмём число 4 и исходя из этого создадим новую модель KMeans и сразу прогоним все координаты через эту модель для определения их кластеров:

```
# Создадим модель для KMeans
kmeans = KMeans(n_clusters = 4, init = 'k-means++',
random_state = 0)

# Получение предсказаний того, к какому кластеру относится
отдельная пара координат (lat; lng)
y_kmeans = kmeans.fit_predict(location)
```

```
array([2, 2, 2, 2, 2, 3, 2, 2, 0, 3, 2, 2, 1, 3, 2, 2, 2, 2, 1, 1, 3, 2,
1, 2, 2, 3, 2, 1, 2, 2, 0, 2, 2, 2, 2, 2, 3, 1, 2, 2, 0, 0, 1, 2,
2, 1, 1, 2, 0, 0, 2, 1, 3, 1, 2, 1, 2, 1, 0, 1, 3, 2, 3, 2, 1, 2,
1, 1, 2, 2, 2, 2, 2, 0, 2, 1, 2, 3, 2, 1, 2, 1, 1, 3, 1, 1, 1, 0,
2, 0, 2, 2, 3, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1, 2, 2, 1, 2, 3,
1, 2, 3, 1, 2, 2, 0, 0, 3, 2, 1, 1, 2, 2, 2, 1, 2, 2, 1, 2, 2, 3,
2, 2, 1, 1, 2, 1, 2, 1, 2, 3, 1, 1, 1, 2, 2, 1, 0, 2, 1, 2, 1, 1,
0, 0, 1, 2, 2, 2, 1, 2, 0, 1, 2, 2, 2, 2, 1, 1, 0, 2, 2, 2, 2, 2,
0, 2, 2, 2, 0, 2, 3, 0, 1, 3, 3, 1, 2, 0, 2, 2, 1, 2, 0, 1, 2, 1,
1, 1, 1, 2, 2, 2, 1, 3, 2, 2, 1, 2, 2, 2, 2, 2, 3, 2, 2, 2, 1, 2,
2, 2, 2, 2, 1, 0, 2, 2, 1, 3, 0, 0, 2, 3, 1, 2, 2, 2, 1, 3, 1, 3,
2, 3, 2, 1, 1, 1, 0, 2, 1, 2, 3, 2, 0, 2, 2, 2, 1, 3, 3, 2, 3, 1,
2, 3, 2, 2, 1, 1, 1, 2, 2, 1, 1, 3, 1, 3, 1, 2, 1, 3, 1, 2, 2, 1,
2, 2, 2, 1, 1, 1, 2, 3, 2, 1, 0, 2, 1, 2, 3, 2, 3, 2, 2, 1, 2, 1,
1, 2, 1, 2, 3, 1, 2, 2, 1, 0, 2, 2, 2, 3, 1, 1, 2, 3, 0, 1, 0, 2,
1, 2, 1, 3, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 0, 2, 1, 1, 3, 3, 3, 1,
2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 3, 1, 3, 2, 3, 2, 1, 1, 2, 2, 1, 3,
1, 1, 2, 2, 2, 0, 2, 0, 2, 2, 3, 3, 1, 2, 1, 2, 2, 0, 2, 1, 2, 3,
2, 2, 3, 2, 2, 2, 1, 2, 1, 3, 0, 2, 2, 1, 2], dtype=int32)
```

Рисунок 24 – Результат соответствия номера кластера определённой координате (latitude; longitude)

Теперь выведем расположение недвижимости с выделением каждой определённым цветом, который зависит от номера кластера, присвоенному каждой записи из набора данных:

```
from matplotlib.colors import ListedColormap

raw_colors = ("red", "green", "blue", "orange")
colors = ListedColormap(raw_colors)

for i in range(5):
    plt.scatter(location[y_kmeans == i, 0], location[y_kmeans
== i, 1], s= 60, c = colors(i), marker = "x")

X_clusters = kmeans.cluster_centers_[ :, 0]
Y_clusters = kmeans.cluster_centers_[ :, 1]

plt.scatter(X_clusters, Y_clusters, s= 60, c= "black")
```



```
plt.title('Кластеры областей')
plt.xlabel('latitude')
plt.ylabel('longitude')
plt.show()
```

Недвижимость, отнесённая к 0-ому кластеру, будет выделена красным, к 1-ому – зелёным, к 2-ому – синим, и к 3-ему – оранжевым. При этом центры кластеров выделены чёрным кругом.

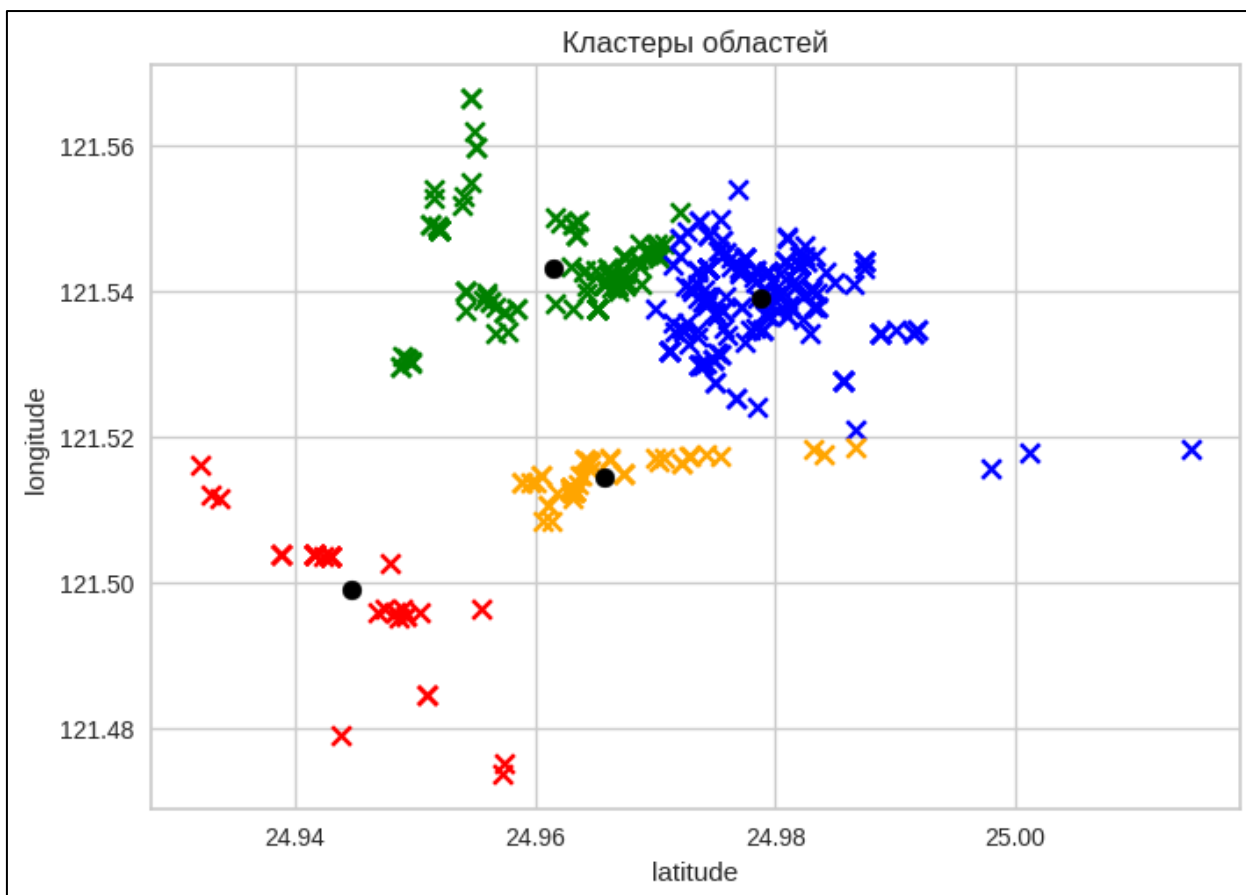


Рисунок 25 – Результат выделения групп, принадлежащих отдельным кластерам с помощью метода k-means

С помощью пакета `folium` можно также вывести сгруппированные недвижимости на карте:

```
area = y_kmeans
df_new = df.copy()

df_new["unit area"] = area
df_new

location_map = df_new[['latitude', 'longitude', 'unit area']]

def getColorByArea(area):
    if area == 0:
        return "red"
```

```

        elif area == 1:
            return "green"
        elif area == 2:
            return "blue"
        else:
            return "orange"

    location_map['color'] = location_map['unit
area'].apply(getColorByArea)
    location_map['size'] = location_map['unit
area'].apply(lambda area: 6)

import folium

# Рисование карты
map = folium.Map(location_map = [24.968, 121.53], zoom_start
= 13)

zip_res = zip(location_map['latitude'],
location_map['longitude'], location_map['unit area'],
location_map['color'], location_map['size'])

# Рисование маркеров, соответствующих элементам недвижимости
распределённым по группам
for lat,lon,price,color,size in zip_res:
    folium.CircleMarker([lat, lon],
                        popup=price,
                        radius=size,
                        color='b',
                        fill=True,
                        fill_opacity=0.7,
                        fill_color=color,

    ).add_to(map)

map

```

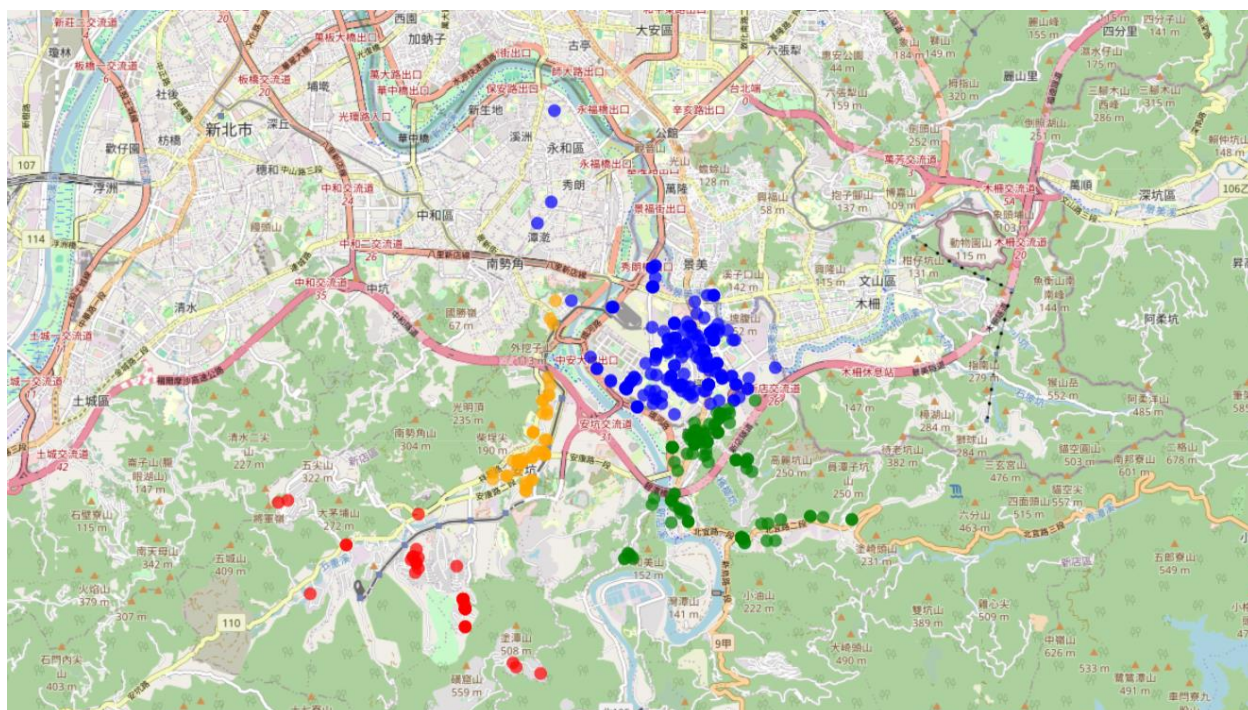


Рисунок 26 – Расположение недвижимости на карте, исходя из их координат (latitude; longitude)

На рисунке 26 показано, что недвижимость действительно была сгруппирована на отдельные группы, элементы в которых наиболее близко расположены к своим группам. Задача успешно решена с помощью метода *k*-ближайших соседей (*kmeans*).

4. Анализ данных методом опорных векторов

Задача будет сформулирована аналогичной тем, которые были решены при методе одномерной линейной регрессии и методе дерева решений с целью оценить, каким образом будет улучшен результат предсказания модели. Это целесообразно, поскольку корреляция цены за единицу площади недвижимости сильно коррелирует именно с расстоянием до ближайшей станции метро.

Сформулированная задача для анализа данных: предсказать цену недвижимости за единицу площадь в зависимости от расстояния до ближайшей станции метро с помощью метода опорных векторов.

Для начала импортируем класс SVR из пакета sklearn.svm:

```
from sklearn.svm import SVR
```

Затем, разделим данные на обучающую и тестовую выборки:

```
# Подготовка набора данных
X_label_svr = "distance to the nearest MRT station"

X_svr = df[X_label_svr]
y_svr = df[Y_label]

# Делим датасет на обучающий и тестовый наборы
X_train_svr, X_test_svr, y_train_svr, y_test_svr =
train_test_split(
    X_svr,
    y_svr,
    test_size=0.3,
    random_state=101
)

X_train_svr = X_train_svr.to_numpy().reshape(-1, 1)
X_test_svr = X_test_svr.to_numpy().reshape(-1, 1)

y_train_svr = y_train_svr.to_numpy()
y_test_svr = y_test_svr.to_numpy()
```

Теперь создадим модель метода опорных векторов и обучим её на обучающей выборке:

```
modelSVR = SVR()
modelSVR.fit(X_train_svr, y_train_svr)
```

И получим оценку точности предсказания модели на тестовых данных:

```
modelSVR.score(X_test_svr, y_test_svr)
```

```
✓ [207] # Оценка точности предсказаний модели
0 сек. modelSVR.score(X_test_svr, y_test_svr)

⇒ 0.6298849415461867
```

Рисунок 27 – Оценка обучения модели SVR

Полученная оценка точности предсказаний модели лучше, чем у модели на основе регрессора дерева решений и линейного регрессора.

Визуализируем результат 10-ти предсказаний модели на основе SVR:

```
# Предсказание цены на недвижимость по числу лет недвижимости
y_pred_svr = modelSVR.predict(X_test_svr)

# Вывод 10-ти предсказанных цен
pd.DataFrame({'Test': y_test_svr, 'Pred': y_pred_svr
}).head(10)
```

```
✓ [208] # Предсказание цены на недвижимость по числу лет недвижимости
0 сек. y_pred_svr = modelSVR.predict(X_test_svr)

# Вывод 10-ти предсказанных цен
pd.DataFrame({'Test': y_test_svr, 'Pred': y_pred_svr }).head(10)
```

	Test	Pred
0	36.5	42.252569
1	40.6	41.930997
2	39.1	45.760129
3	36.9	45.769492
4	35.3	42.664317
5	43.5	44.759506
6	30.5	43.368509
7	42.8	46.082232
8	51.0	44.599685
9	50.0	46.155075

Рисунок 28 – Результат 10-ти предсказаний цены на недвижимость

Несмотря на то, что общая оценка точности модели выше, чем у модели на основе метода деревьев, первые 10-ть предсказаний имеют высокие отклонения и нет даже приближённо совпадающих значений между тестовыми и предсказанными.

Получим значения метрик:

```
y_pred_svr = modelSVR.predict(X_test_svr)
```

```

# Вычисляем ошибку
MAE = metrics.mean_absolute_error(y_test_svr, y_pred_svr)
MSE = metrics.mean_squared_error(y_test_svr, y_pred_svr)
RMSE = np.sqrt(MSE)

pd.DataFrame([MAE, MSE, RMSE], index=['MAE', 'MSE', 'RMSE'],
columns=['Metrics'])

```

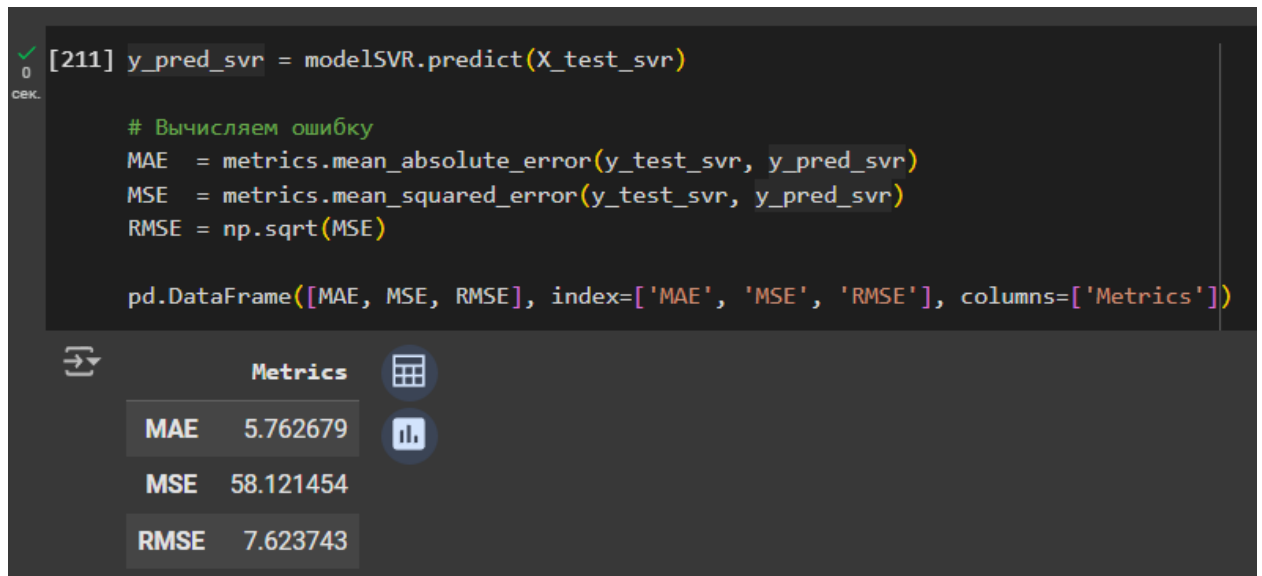


Рисунок 29 – Результат вычисления метрик

Как видно из рисунка 29 значения метрик всё-таки улучшились с использованием метода опорных векторов, по сравнению с моделями на базе линейного регрессора и регрессора дерева решений, хотя первые 10-ть предсказаний данной модели неудачны.

Для получения более подробного представления о результатах обучения выведем график тестовых и предсказанных значений:

```

x_ax_svr = range(len(y_test_svr))
plt.plot(x_ax_svr, y_test_svr, label="Тестовые значения")
plt.plot(x_ax_svr, y_pred_svr, label="Предсказанные значения")
plt.title("Тестовые и предсказанные значения")

plt.xlabel('X')
plt.ylabel('Y')

plt.legend(loc='best', fancybox=True, shadow=True)
plt.grid(True)
plt.show()

```

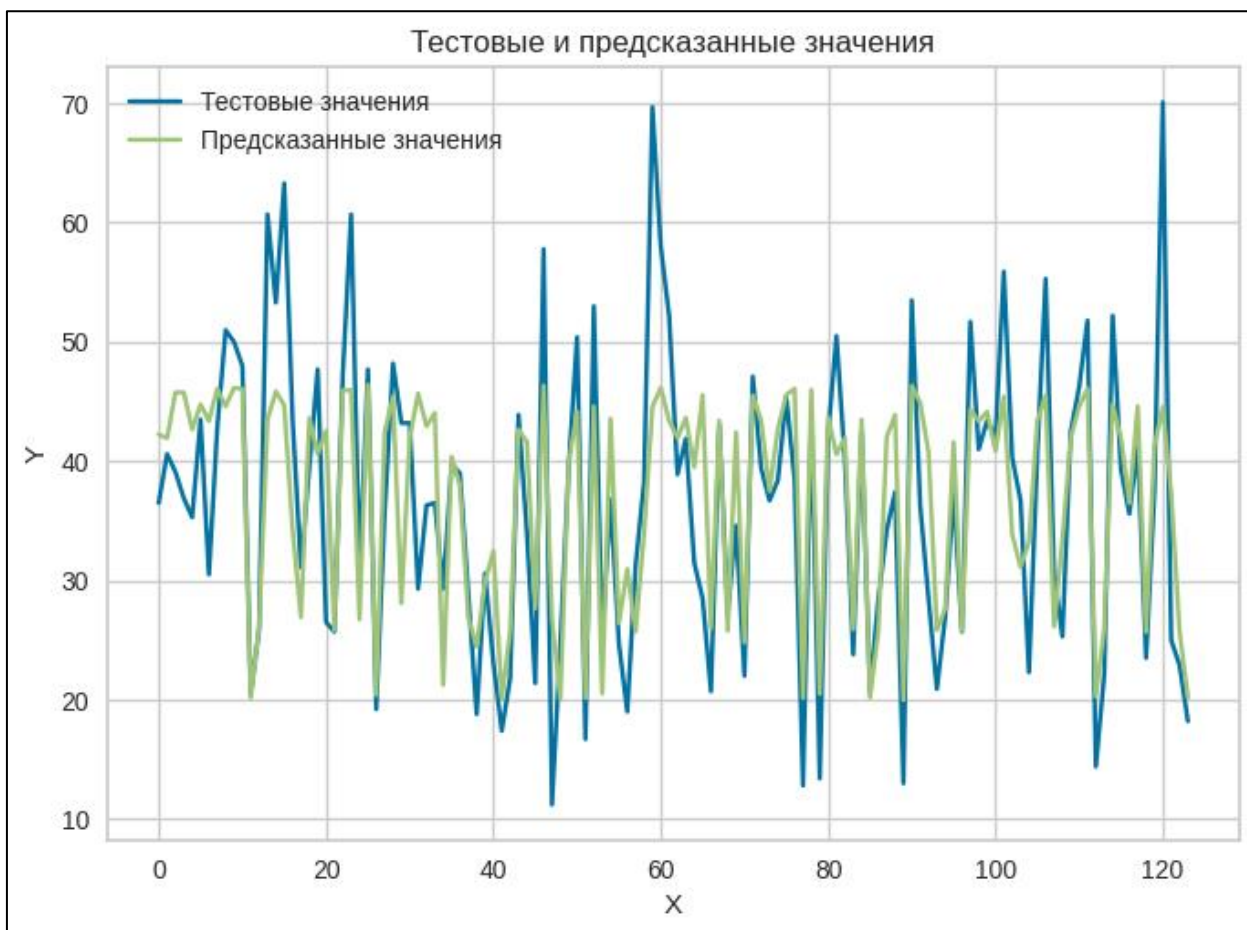


Рисунок 30 – График предсказанных и тестовых значений

На рисунке 30 видно, что модель действительно плохо предсказывает первые значения, однако последующие предсказывает довольно неплохо, а местами предсказанные значения практически полностью совпадают с тестовыми. Вероятно, за счёт отсутствия слишком большого разброса в предсказанных значениях регрессионные метрики выдают более меньшую ошибку, чем в других методах. Не смотря на начальные ошибки, модель лучше всех справляется с предсказанием последующих значений, что делает её самой лучшей при сравнении с другими методами конкретно для решения данной задачи.

Выводы

В рамках данной лабораторной работы были изучены способы применения методов анализа данных из заданного набора, а также известных алгоритмов на их основе, для решения предсказательных задач, в том числе за счёт построения и использования классификаторов, регрессионных или кластерных моделей.

Для анализа данных с помощью метода одномерной линейной регрессии, метода дерева решений и метода опорных векторов была сформулирована одна общая задача: предсказать цену недвижимости за единицу площади в зависимости от расстояния до ближайшей станции метро. Общая задача была сформулирована с целью выявления наиболее лучшего метода для решения поставленной задачи. В результате анализа корреляций между столбцами была выявлена наибольшая корреляция между столбцами “цена недвижимости за единицу площади” и “расстояние до ближайшей станции метро”, именно поэтому в трёх задачах фигурируют эти два столбца, поскольку с другими столбцами цена недвижимости за единицу площади коррелирует незначительно.

В результате решения общей задачи было выявлено, что метод опорных векторов является самым лучшим для её решения. Метод дерева решений показал себя чуть хуже по метрикам и оценке модели на тестовых данных чем метод опорных векторов, но лучше метода одномерной линейной регрессии. Метод одномерной линейной регрессии оказался самым худшим выбором для решения поставленной задачи. Оценка методов осуществлялась по полученным результатам вычисления функций ошибок (MAE, MSE, RMSE) и визуальной оценки графиков соответствия предсказанных значений тестовым.

Была также сформулирована задача для анализа данных методом k-ближайших соседей: определить отдельные группы (кластеры) недвижимости по их географическим координатам. В результате решения данной задачи был использован метод локтя, с помощью которого было получено оптимальное число кластеров для использования модели KMeans. Группы недвижимости были также визуализированы на карте, с помощью пакета folium.

В результате выполнения лабораторной работы её цель была полностью достигнута, а все поставленные задачи успешно выполнены.

Список использованных источников

1. Что такое линейная регрессия? (URL: <https://aws.amazon.com/ru/what-is/linear-regression/>)
2. Коэффициент детерминации (Coefficient of determination) (URL: <https://wiki.loginom.ru/articles/coefficient-of-determination.html>)
3. Нисчал Н. Python — это просто. Пошаговое руководство по программированию и анализу данных: Пер. с англ. — СПб.: БХВ - Петербург, 2022.
4. Погружение в аналитику данных. Маунт Джордж: Пер. с англ. — СПб.: БХВ-Петербург, 2023.
5. Основы линейной регрессии (URL: <https://habr.com/ru/articles/514818/>)
6. Что такое дерево решений и где его используют? (URL: <https://habr.com/ru/companies/productstar/articles/523044/>)
7. Метод K-Nearest Neighbors. Разбор без использования библиотек и с использованием библиотек (URL: <https://habr.com/ru/articles/680004/>)
8. Метод опорных векторов (SVM). Подходы, принцип работы и реализация с нуля на Python (URL: <https://habr.com/ru/articles/802185/>)

Приложение А

Ссылка на исходный код

1. Ссылка на исходный код проекта:
https://github.com/DanSoW/INRTU/blob/main/data-processing-analysis-and-visualization-technologies/lab3/lab3_tech_program.ipynb