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 №4**

по дисциплине «Технологии обработки, анализа и визуализации данных»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ИИТм-23-1 |  |  |  | Солопов Д.Д. |
|  |  | шифр группы |  | подпись |  | И.О. Фамилия |
| Проверил |  |  |  |  |  | Григорьев С.В. |
|  |  |  |  | подпись |  | И.О. Фамилия |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Иркутск 2025 г

**Содержание**

[Введение 3](#_Toc190692664)

[1. Анализ данных методом многомерной линейной регрессии 4](#_Toc190692665)

[2. Анализ данных методом выявления аномалий 17](#_Toc190692666)

[3. Анализ данных с помощью метода иерархической кластеризации 20](#_Toc190692667)

[4. Анализ данных методом наивного байесовского классификатора 23](#_Toc190692668)

[Выводы 27](#_Toc190692669)

[Список использованных источников 29](#_Toc190692670)

[Приложение А 30](#_Toc190692671)

Введение

**Цель работы**: изучить способы применения методов многомерного анализа данных из заданного набора, а также известных алгоритмов на их основе, для решения предсказательных задач, в том числе за счёт построения и использования моделей.

**Задачи**:

1. Сформулировать задачу для анализа данных из данного набора и решить его методом многомерной линейной регрессии;
2. Сформулировать задачу для анализа данных из данного набора и решить её методом выявления аномалий;
3. Сформулировать задачу для анализа данных из данного набора и решить её методом иерархической кластеризации;
4. Сформулировать задачу для анализа данных из данного набора и решить её с помощью наивного байесовского классификатора.

1. Анализ данных методом многомерной линейной регрессии

Многомерная линейная регрессионная модель – это статистическая модель, которая предсказывает значение зависимой переменной на основе нескольких независимых переменных. Она основана на предположении, что зависимая переменная линейно зависит от нескольких независимых переменных.

Простейшая многомерная линейная регрессионная модель выглядит следующим образом:

Где:

1. Y – зависимая переменная, которую мы хотим предсказать;
2. x1, x2, …, xn – независимые переменные, которые мы используем для предсказания;
3. b0, b1, b2, … , bn – коэффициенты модели, которые мы используем для предсказания.

**Сформулированная задача для анализа данных**: предсказать цену недвижимости за единицу площади в зависимости от расстояния до ближайшей станции метро и количества расположенных поблизости круглосуточных магазинов с помощью многомерной 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()

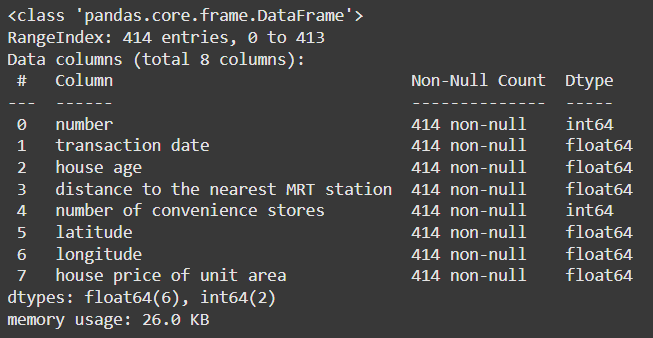


Рисунок 1 – Основная информация о наборе данных

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

1. **distance to the nearest MRT station** – расстояние до ближайшей станции метро;
2. **number of convenience stores** – количество расположенных по близости круглосуточных магазинов;
3. **house price of unit area** – цена недвижимости за единицу площади.

Поскольку исходя из сформулированной задачи мы будем предсказывать цену за единицу площади, то необходимо заранее определится что данный столбец представляет из себя значения Y (то, что будет предсказываться). Поскольку регрессия должна быть многомерной, то значений X будет несколько (они представляют из себя вектор), а именно: расстояние до ближайшей станции метро и количество расположенных поблизости круглосуточных магазинов.

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.

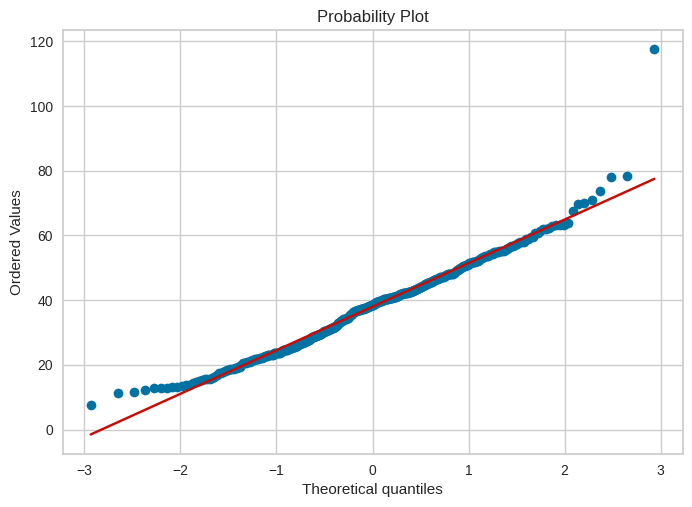


Рисунок 2 – Результат построения квантиль-квантиль графика по цене на недвижимость за единицу площади

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))

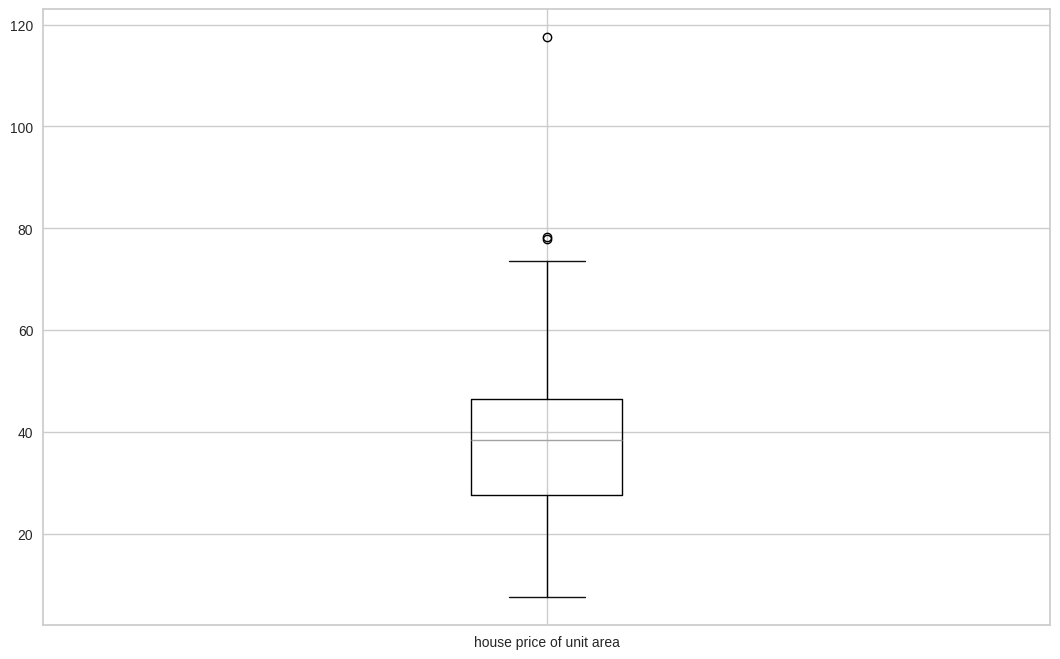


Рисунок 3 – Диаграмма boxplot для цены на недвижимость за единицу площади

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 медианной 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)]

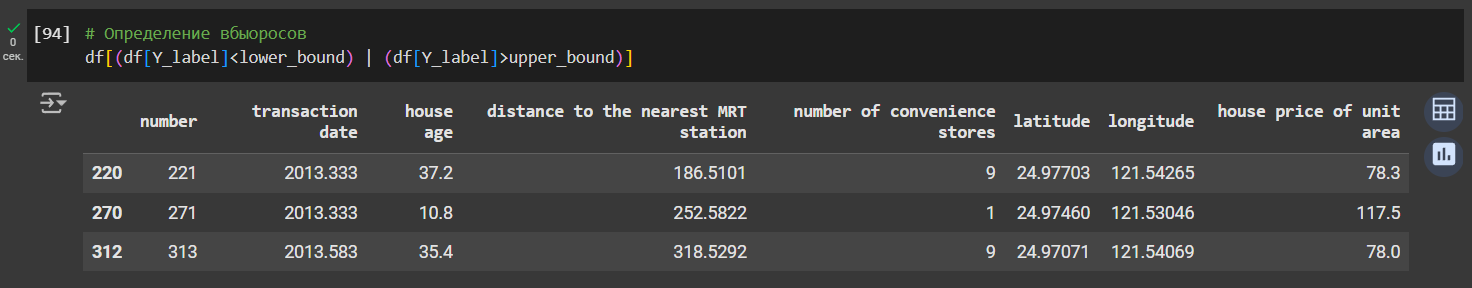


Рисунок 4 – Вывод выбросов в виде значений из датафрейма

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.

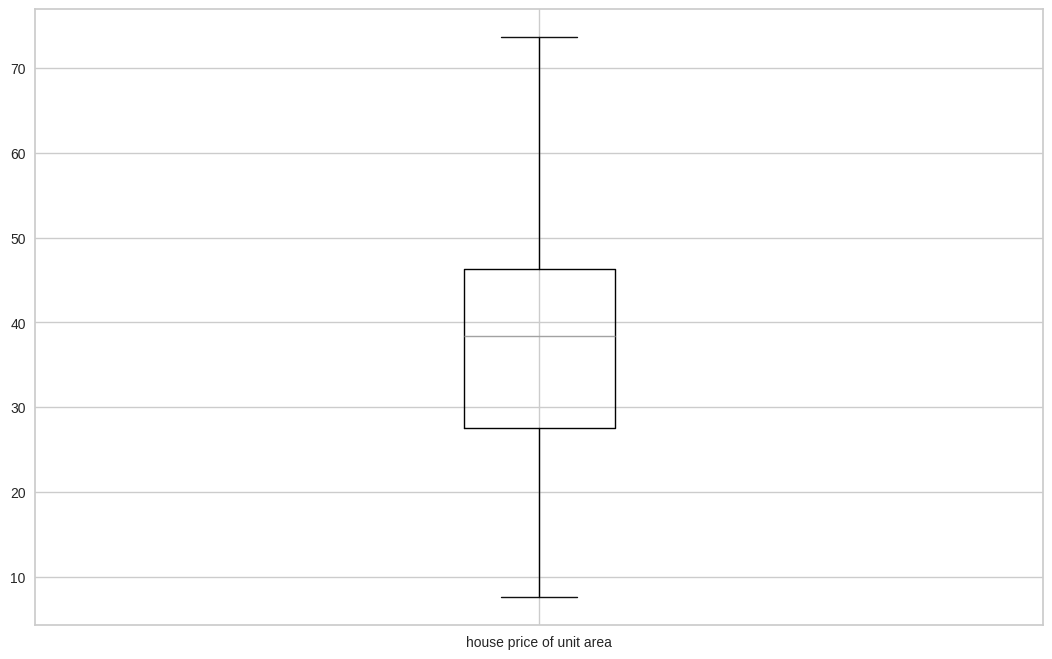


Рисунок 5 - Диаграмма boxplot для цены на недвижимость за единицу площади после удаления выбросов

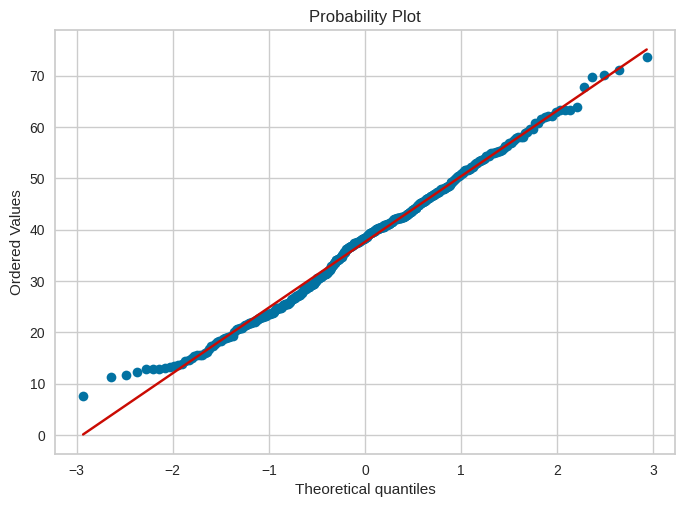


Рисунок 6 - Результат построения квантиль-квантиль графика по цене на недвижимость за единицу площади

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

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

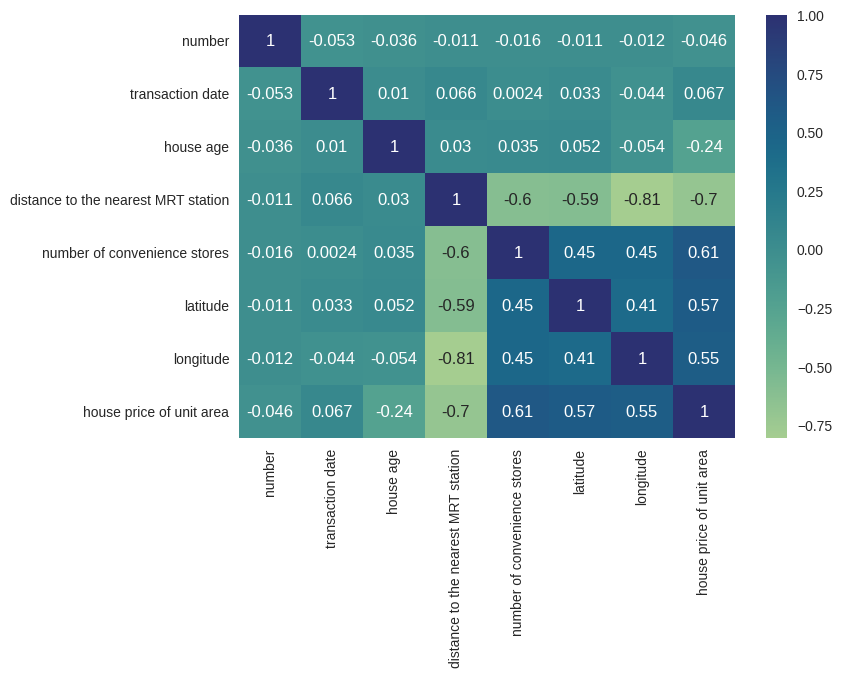


Рисунок 7 – Тепловая карта коэффициентов корреляции

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 сильновыражена, а поэтому для более лучшего предсказания регрессионной модели цены её целесообразно использовать в качестве элемента вектора X.

Также, можно отметить положительную сильновыраженную корреляцию между столбцом number of convenience stores (X) и house price of unit area (Y), коэффициент которой равен 0.61. Данную корреляцию можно трактовать следующим образом: чем больше круглосуточных магазинов рядом с недвижимостью, тем цена за единицу её площади будет больше. Таким образом данный столбец также целесообразно использовать в качестве элемента вектора X.

Теперь построим модель многомерной 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 вектор X и переменную Y:

Y\_label = "house price of unit area"

X\_label = ["distance to the nearest MRT station", "number of convenience stores"]

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

)

y\_train = y\_train.to\_numpy()

y\_test = y\_test.to\_numpy()

Теперь создаём модель многомерной линейной регрессии и обучим 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

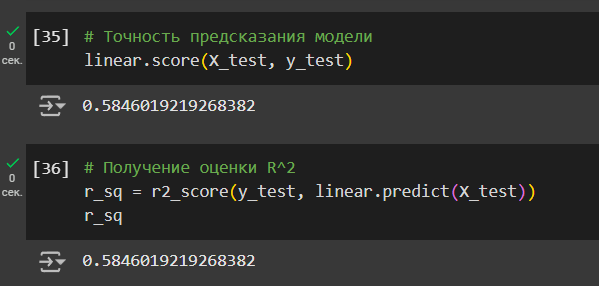


Рисунок 8 – Результат оценки точности модели

Точность предсказания модели на тестовом наборе данных составляет 0.584, что является довольно неплохим результатом учитывая, что весь набор данных состоит из 411-ой записи.

Теперь выведим оптимальные значения весов b0 и b1, которые были вычислены моделью:

linear.intercept\_

linear.coef\_

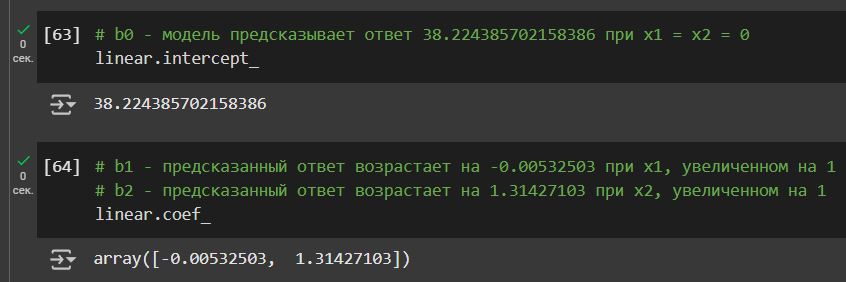


Рисунок 9 – Вывод значения предсказанных коэффициентов b0, b1 и b2 модели многомерной линейной регрессии

Исходя из вычисленных весов получаем, что модель предсказывает значение b0 равным 38.224385702158386 при значениях x1 и x2 равных нулю.

Коэффициент b1 будет возрастать на -0.00532503, при x1 увеличенном на единицу, а коэффициент b2 будет возрастать на 1.31427103, при увеличении на единицу значения x2 (см. рис. 10).

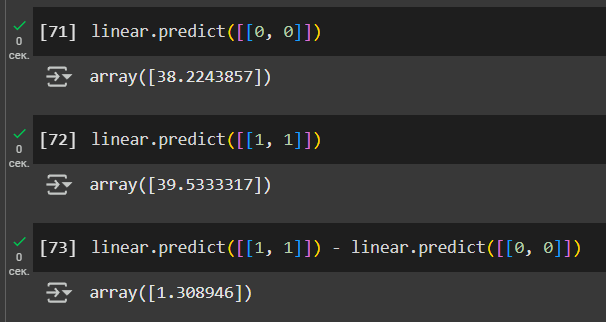


Рисунок 10 – Вывод предсказаний модели для x (0, 0) и x (1, 1), а также разности этих предсказаний

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)

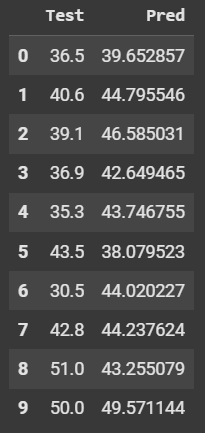


Рисунок 11 – Первые 10 предсказаний по тестовым данным

Как видно из рисунка 11 предсказанные значения модели имеют большую ошибку в результатах. Однако для 7-го и 9-го теста получилось хорошее предсказанице цены за единицу площади недвижимости в зависимости от неизвестных.

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()

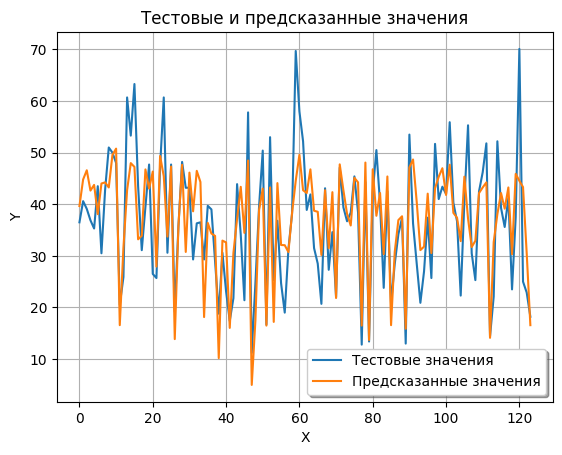


Рисунок 12 – График тестовых и предсказанных значений

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 неизвестных X были выбраны столбцы, которые наиболее коррелируют с y, с помощью многомерной линейной регрессии удалось добиться точности только в 0.584.

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:

Для вычислеия MSE используется следующая формула:

Для вычисления RMSE используется следующая формула:

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'])

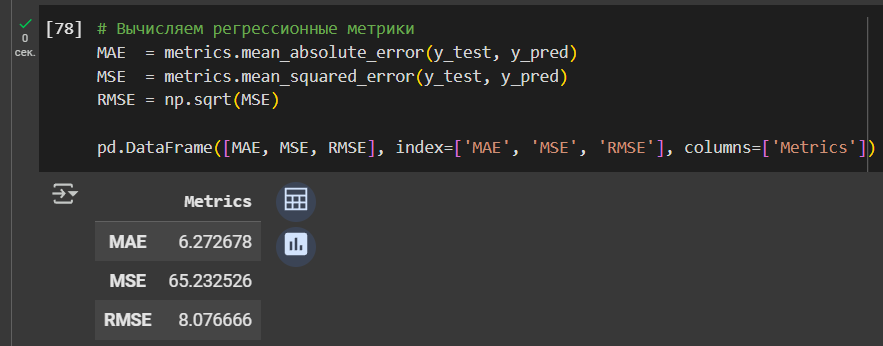


Рисунок 13 – Результат вычисления метрик

После вычисления получаем, что MAE равно 6.272678, MSE равно 65.232526, а RMSE равно 8.076666.

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 -1.5, и уменьшается с промежутка от -1.5 до 32.

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

2. Анализ данных методом выявления аномалий

**Сформулированная задача для анализа данных**: определить аномалии в двух столбцах из исходного набора данных, а именно: расстояние до ближайшей станции метро и цена за единицу площади недвижимости. Для обнаружения аномалий необходимо использовать метод PCA (анализ главных компонент), который выявляет аномалии путём выявления выбросов на основе главных компонент и интерпретировать полученные результаты

Для начала импортируем необходимые библиотеки для обнаружения аномалий в наборе данных:

from pyod.utils.data import generate\_data, get\_outliers\_inliers

from pyod.models.pca import PCA

from pyod.utils.data import evaluate\_print

from pyod.utils.example import visualize

Теперь подготовим данные, аномалии в которых будем анализировать:

Y\_label = "house price of unit area"

X\_label = ["distance to the nearest MRT station", "number of convenience stores"]

X = df[X\_label]

y = df[Y\_label]

df\_train = pd.DataFrame(X)

df\_train['y'] = y

Теперь построим график зависимости цены за единицу недвижимости от расстояния до ближайшей станции метро:

sns.scatterplot(x="distance to the nearest MRT station", y="y", hue='y', data=df\_train, palette="hls")

plt.title('Зависимость цены от расстояния до станции')

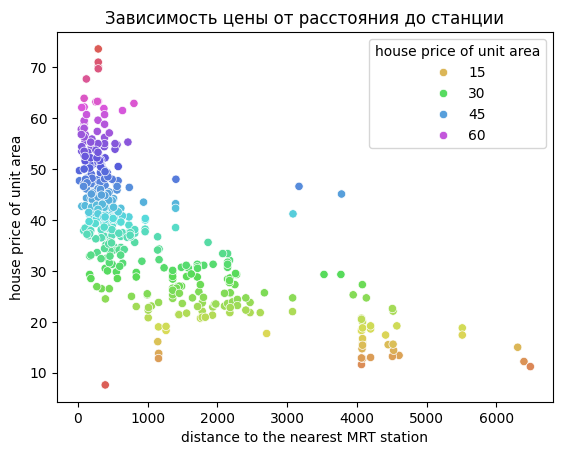


Рисунок 15 – График зависимости цены от расстояния до метро

Теперь создадим и обучим модель PCA на исходном наборе данных:

# Создание модели PCA (метод главных компонент)

clf = PCA()

X\_train = X.to\_numpy().reshape(-1, 1)

# Тренировка PCA модели

clf.fit(X\_train)

После того, как тренировка PCA модели завершена, мы можем получить предсказанные значения и оценку аномалий, которая определила модель после обучения:

# Предсказанные значения и оценки аномалий методом главных компонент

y\_train\_pred = clf.labels\_

y\_train\_scores = clf.decision\_scores\_

Для визуализации оценок аномалий нужно использовать следующий программный код:

ax = sns.scatterplot(x=X\_label, y=Y\_label, hue=y\_train\_scores, data=df\_train, palette="RdBu\_r")

legend\_labels = [f"{score:.2f}" for score in np.unique(y\_train\_scores)]

ax.legend(title="Оценка аномалии", labels=legend\_labels)

plt.title('Оценки аномалий по методу PCA')

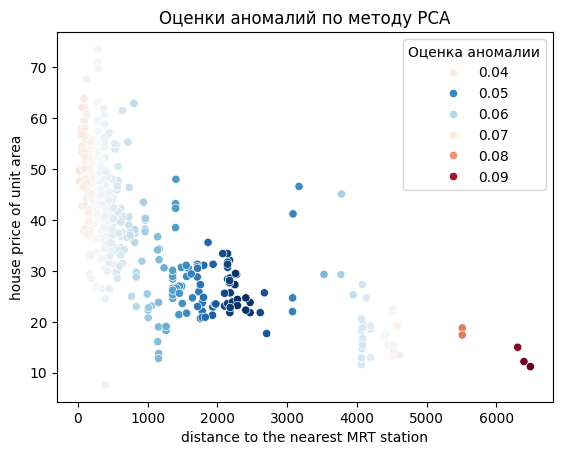


Рисунок 16 – График оценки аномалий полученный методом PCA

**Интерпретация результатов**:

**Плотные кластеры**: плотные кластеры точек в центре с более низкими показателями аномалий (0.04-0.06) указывают на нормальные точки данных, которые соответствуют ожидаемому шаблону.

**Рассеянные точки**: рассеянные точки с более высокими показателями аномалий (красные и темнее) указывают на потенциальные аномалии или выбросы, которые значительно отклоняются от нормального шаблона (расстояние до ближайшей станции метро, которых больше 5000).

Данная визуализация помогла определить точки аномалий, которые значительно удалены от плотных кластеров и имеют высокие оценки аномалий.

3. Анализ данных с помощью метода иерархической кластеризации

Иерархическая кластеризация – это метод кластерного анализа, целью которого является построение иерархии кластеров. Он особенно полезен, когда количество кластеров заранее неизвестно.

Основная идея такой кластеризации заключается в создании древовидной структуры (дендограммы), представляющей вложенную группировку точек данных.

Агломеративная кластеризация – это подход “снизу-вверх”. Данная кластеризация начинается с каждой точки данных как отдельного кластера и итеративно объединяет ближайшие пары кластеров до тех пор, пока все точки не окажутся в одном кластере или не будет выполнен критерий остановки.

**Сформулированная задача для анализа данных**: определить различные группы (кластеры) недвижимости относительно географического расположения в пространстве используя метод агломеративной кластеризации.

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

Для начала импортируем все необходимые библиотеки:

import numpy as np

import matplotlib.pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage

from sklearn.cluster import AgglomerativeClustering

Теперь предварительно подготовим данные:

Y\_label = "longitude"

X\_label = "latitude"

X = df[X\_label].to\_numpy()

y = df[Y\_label].to\_numpy()

data = list(zip(X, y))

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

# Визуализация местоположения недвижимости

plt.scatter(x=X, y=y, marker = "x", color = 'purple', s = 60)

plt.xlabel('latitude')

plt.ylabel('longitude')

plt.show()

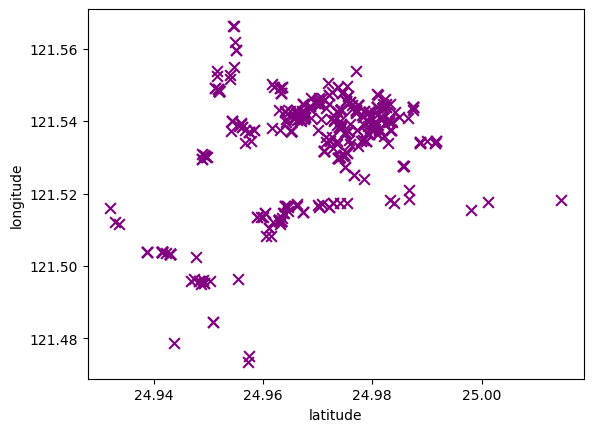


Рисунок 17 – График положения недвижимости в пространстве по координатам

Теперь визуализируем дендограмму, на основе исходных данных:

linkage\_data = linkage(data, method='ward', metric='euclidean')

dendrogram(linkage\_data)

plt.show()

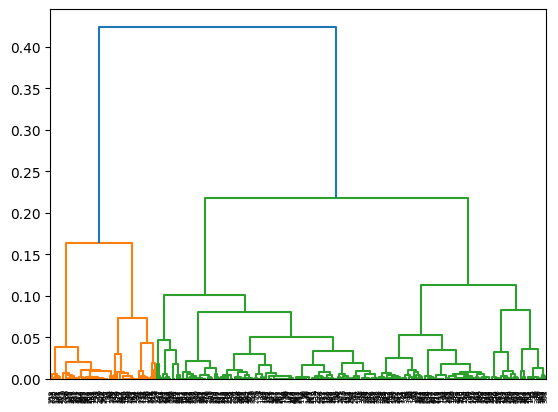


Рисунок 18 – Дендограмма

Из рисунка 18 можно сделать предположение о том, что из существующих данных можно сформировать как минимум 4 различных кластера.

Исходя из данного предположения используем агломеративный метод из библиотеки scipy и визуализируем полученные кластеры:

hierarchical\_cluster = AgglomerativeClustering(n\_clusters=4, linkage='ward')

labels = hierarchical\_cluster.fit\_predict(data)

plt.scatter(X, y, c=labels, marker = "x")

plt.xlabel('latitude')

plt.ylabel('longitude')

plt.show()

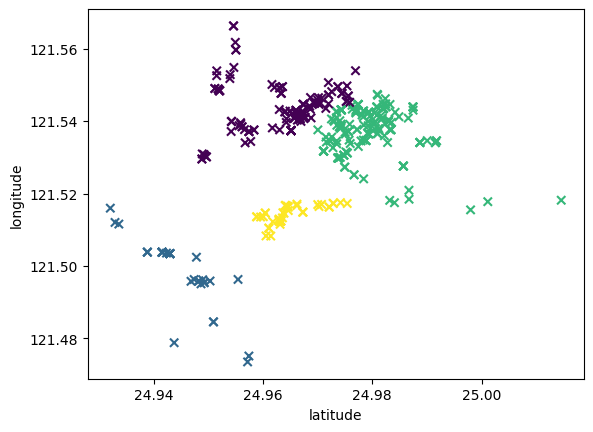


Рисунок 19 – Визуализация полученных кластеров с помощью агломеративного метода

Таким образом расположение недвижимости по координатам latitude и longtitude разделены на 4 кластера с помощью агломеративного метода (иерархической кластеризации).

4. Анализ данных методом наивного байесовского классификатора

Использование наивного байесовского классификатора предполагает наличие такого набора обучающих данных, в которых классы либо уже выделены, либо их можно выделить предварительно для использования классификации с помощью байесовского классификатора. В рамках данного задания таким набором данных будут выступать результаты выполнения предыдущей задачи, в которой уже были определены классы (их 4) и принадлежность координат недвижимости определённому классу. Такая мера необходима, поскольку в наборе данных нет значений, которые бы характеризовали какой-либо класс. Байесовский классификатор будет решать задачу соответствия определённых координат недвижимости определённому классу (расположению в пространстве).

**Сформулированная задача для анализа данных**: используя модифицированный набор данных, полученный при решении задачи методом иерархической кластеризации, обучить наивный байесовский классификатор определять класс недвижимости, расположенной по определённым координатам.

Для начала, подготовим обучающую и тестовую выборки данных на основе уже существующих данных (местоположение и классы):

location = df.iloc[:, [3, 4]].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

location,

labels,

test\_size=0.3,

random\_state=101

)

На рисунке 20 представлен формат используемых данных.

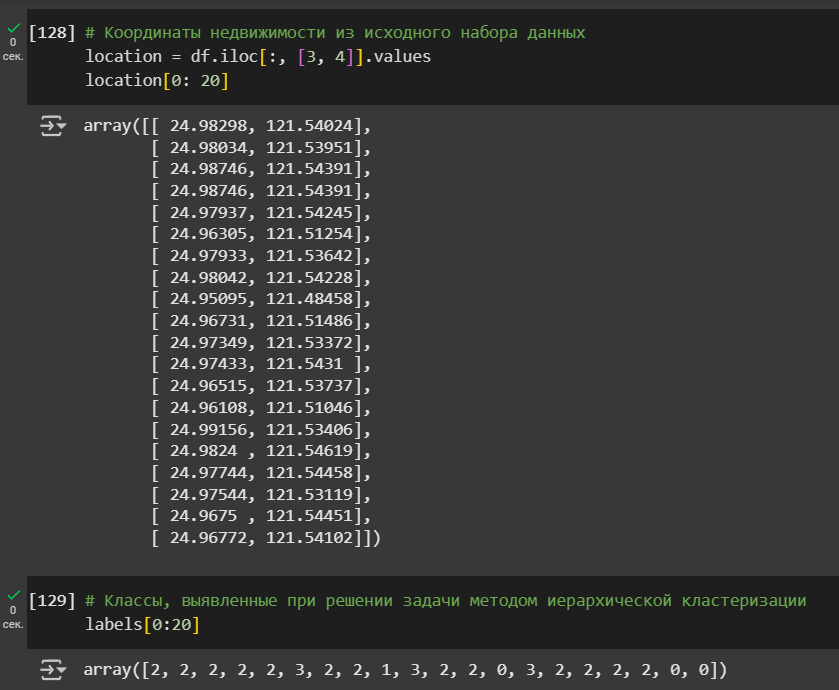


Рисунок 20 – Формат данных в массиве местоположений и метках

Теперь импортируем необходимые библиотеки для выполнения классификации с помощью наивного байесовского классификатора:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import load\_iris

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

from mlxtend.plotting import plot\_decision\_regions

Затем, разделим данные на обучающую и тестовую выборки:

location = df.iloc[:, [3, 4]].values

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

location,

labels,

test\_size=0.3,

random\_state=101

)

Теперь создадим модель наивного байесовского классификатора и обучим её на обучающей выборке:

# Определение модели байесовского классификатора

sk\_nb\_clf = GaussianNB()

# Обучение байесовского классификатора на тренировочной выборке

sk\_nb\_clf.fit(X\_train, y\_train)

И получим оценку точности предсказания модели на тестовых данных:

# Предсказание данных по тестовому набору

sk\_nb\_clf\_pred\_res = sk\_nb\_clf.predict(X\_test)

# Оценка точности классификации

sk\_nb\_clf\_accuracy = accuracy\_score(y\_test, sk\_nb\_clf\_pred\_res)

print(f'Оценка точности наивного байесовского классификатора: {sk\_nb\_clf\_accuracy}')

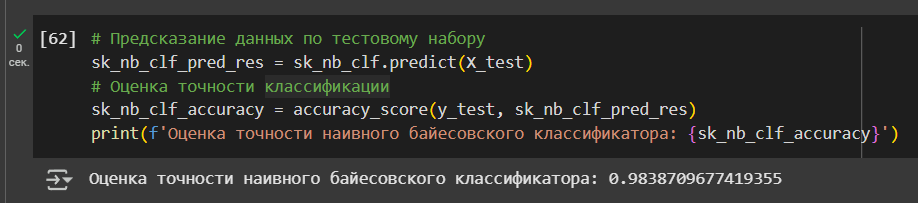


Рисунок 21 – Оценка обучения модели наивного байесовского классификатора

В результате была получена точность в 0.9838709677419355.

Данная точность свидетельствует о хорошей классификации с помощью метода байесовского классификатора. Таким образом можно определить класс недвижимости по её координатам.

Дополнительно получим значения метрик:

# Вычисляем ошибку

MAE = metrics.mean\_absolute\_error(y\_test, sk\_nb\_clf\_pred\_res)

MSE = metrics.mean\_squared\_error(y\_test, sk\_nb\_clf\_pred\_res)

RMSE = np.sqrt(MSE)

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

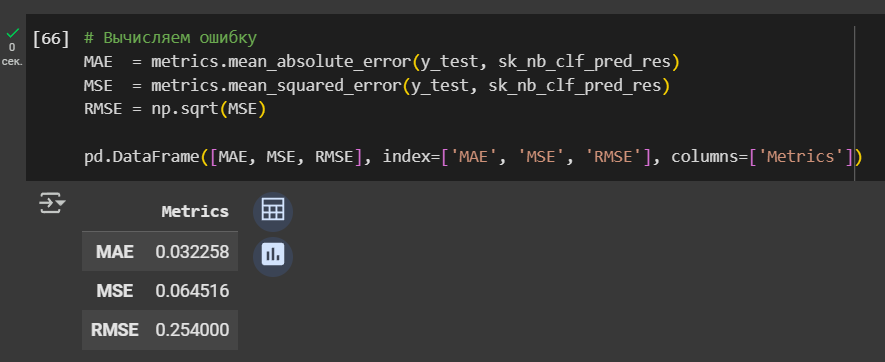


Рисунок 22 – Результат вычисления метрик

Исходя из результатов, представленных на рисунке 22, можно дополнительно судить о том, что результаты классификации тренировочного набора данных незначительно отличаются от тестового набора, поскольку метрики минимальны.

Выводы

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

Для анализа данных с помощью метода многомерной линейной регрессии была сформулирована задача: предсказать цену недвижимости за единицу площади в зависимости от расстояния до ближайшей станции метро и количества расположенных поблизости круглосуточных магазинов. При решении данной задачи была построена модель многомерной линейной регрессии, которая была обучена на предварительно подготовленном наборе обучающих данных. Несмотря на то, что в качестве неизвестных X были выбраны столбцы, которые наиболее коррелируют с y, с помощью многомерной линейной регрессии удалось добиться точности в 0.584, что является довольно неплохим результатом в рамках данной задачи.

Для анализа данных с помощью метода поиска аномалий была сформулирована следующая задача: определить аномалии в двух столбцах из исходного набора данных (расстояние до ближайшей станции метро и цена за единицу площади недвижимости). Для обнаружения аномалий необходимо использовать метод PCA (анализ главных компонент), который выявляет аномалии путём выявления выбросов на основе главных компонент и интерпретировать полученные результаты. В результате была построена модель для анализа главных компонент и выявлены плотные кластеры точек в центре с более низкими показателями аномалий (0.04-0.06), которые указывают на нормальные точки данных соответствующие ожидающему шаблону. Однако, были также выявлены и рассеянные точки с более высокими показателями аномалий, указывающие на потенциальные аномалии или выборы. С помощью визуализации были определены точки аномалий, которые значительно удалены от плотных кластеров и имеют высокие оценки аномалий.

Для анализа данных с помощью метода иерархической кластеризации была сформулирована следующая задача: определить различные группы (кластеры) недвижимости относительно географического расположения в пространстве используя метод агломеративной (иерархической) кластеризации. В результате анализа данных была построена дендограмма и выделены отдельные кластеры, которые будут использованы также и в другой схожей задаче – с использованием метода наимного байесовского классификатора для классификации недвижимости к отдельному классу.

Для анализа данных с помощью метода наивного байесовского классификатора была сформулирована следующая задача: используя модифицированный набор данных, полученный при решении задачи методом иерархической кластеризации, обучить наивный байесовский классификатор определять класс недвижимости, расположенной по определённым координатам. В результате применения данного метода была получена точность классификации различной недвижимости по её географическому местоположению равная 0.9834, что является весьма неплохим результатом для применения данного метода.

В рамках данной лабораторной работы были выполнены все поставленные задачи и достигнута её цель.

Список использованных источников

1. Нисчал Н. Python — это просто. Пошаговое руководство по программированию и анализу данных: Пер. с англ. — СПб.: БХВ Петербург, 2022;
2. Погружение в аналитику данных. Маунт Джордж: Пер. с англ. — СПб.: БХВ-Петербург, 2023;
3. Пример решения задачи множественной регрессии с помощью Python (URL: <https://habr.com/ru/articles/206306/> );
4. Агломеративная кластеризация и дендрограмма в Python (URL: <https://teletype.in/@dt_analytic/LtpSsL__xO2> );
5. Истина где‑то рядом — ищем аномалии с Python. Часть 1: теория (URL: <https://www.reg.ru/blog/ishchem-anomalii-s-python-chast-1/> );
6. Наивный байесовский классификатор. Основная идея, модификации и реализация с нуля на Python (URL: <https://habr.com/ru/articles/802435/> ).

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/lab4/lab4_tech_program.ipynb>