

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по курсу
«Data Science Pro»

Слушатель

Раски Ирина Геннадьевна

Москва, 2024

Содержание

1. Введение.....	3
2. Аналитическая часть.....	4
2.1. Постановка задачи.....	4
2.2. Описание используемых методов	5
2.3. Разведочный анализ данных	10
3. Практическая часть	17
3.1. Предобработка данных	17
3.2. Разработка и обучение модели	21
3.3. Тестирование модели.....	23
3.4. Нейронная сеть	36
3.5. Разработка приложения	41
3.6. Создание удаленного репозитория.....	42
4. Заключение	44
5. Библиографический список	45

1 Введение

Тема выпускной квалификационной работы: «Прогнозирование конечных свойств новых материалов (композиционных материалов)».

Композиционный материал представляет собой искусственно созданный неоднородный сплошной материал, состоящий из двух или более разнородных компонентов с четкой границей раздела между ними, среди которых можно выделить армирующие элементы, обеспечивающие необходимые механические характеристики материала, и матрицу (или связующее), обеспечивающую совместную работу армирующих элементов. При этом композит обладает свойствами, которыми не обладает ни один из составляющих его материалов

Целью создания композиционного материала является объединение схожих или различных компонентов для получения материала с новыми заданными свойствами и характеристиками, отличными от свойств и характеристик исходных компонентов. С появлением такого рода материалов возникла возможность селективного выбора свойств композитов, необходимых для нужд каждой конкретной области применения.

Яркий пример композита — железобетон. Бетон прекрасно сопротивляется сжатию, но плохо растяжению. Стальная арматура внутри бетона компенсирует его неспособность сопротивляться сжатию, формируя тем самым новые, уникальные свойства. Современные композиты изготавливаются из других материалов: полимеры, керамика, стеклянные и углеродные волокна, но данный принцип сохраняется. У такого подхода есть и недостаток: даже если мы знаем характеристики исходных компонентов, определить характеристики композита, состоящего из этих компонентов, достаточно проблематично. Для решения этой проблемы есть два пути: физические испытания образцов материалов или прогнозирование характеристик. Суть прогнозирования заключается в симуляции представительного элемента объема композита на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

2 Аналитическая часть

2.1 Постановка задачи

Решаемые задачи: спрогнозировать конечные свойства получаемых композиционных материалов при помощи:

- алгоритмов машинного обучения, определяющих значения:
 - Модуль упругости при растяжении, ГПа;
 - Прочность при растяжении, МПа;
- нейронной сети, которая будет рекомендовать:
 - Соотношение матрица-наполнитель.

Кейс основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

Исходные данные:

Датасет со свойствами композитов сформирован путем объединения по индексу (тип - объединения INNER) двух документов в формате xlsx:

- Таблица «X_br.xlsx» (1024 строки и 11 столбцов);
- Таблица «X_nup.xlsx» (1041 строка и 4 столбца)

Ввод [4]:

```
# Объединение данных в датасет
dataset = dataset_1.merge(dataset_2,how="inner",on="Unnamed: 0")
dataset.drop(['Unnamed: 0'], axis=1, inplace=True)
dataset.head()
```

Out[4]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	наш
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0	0	
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0	0	
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0	0	
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0	0	
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0	0	

Рисунок 1 – объединение данных в один датасет

Объем выборки составил 1023 строки и 13 столбцов.

Входные данные о начальных свойствах компонентов композиционных материалов:

- Соотношение матрица-наполнитель,

- Плотность, кг/м³,
- модуль упругости, ГПа,
- Количество отвердителя, м.%,
- Содержание эпоксидных групп, %₂,
- Температура вспышки, С₂,
- Модуль упругости при растяжении, ГПа,
- Прочность при растяжении, МПа,
- Угол нашивки, град,
- Шаг нашивки,
- Плотность нашивки.

Общее количество параметров для анализа – 13.

2.2 Описание используемых методов

Для решения поставленной задачи использованы следующие методы:

- Линейная регрессия (LinearRegression);
- К-ближайших соседей (KNeighborsRegressor);
- Метод опорных векторов (SVR);
- Регрессия дерева решений (DecisionTreeRegressor);
- Случайный лес (RandomForestRegressor),
- Градиентный бустинг (GradientBoostingRegressor),
- Стохастический градиентный спуск (SGDRegressor),
- Нейронная сеть.

Линейная регрессия (LinearRegression) — это алгоритм машинного обучения с учителем, это простой и широко используемый алгоритм регрессии, основанный на предположении о линейной зависимости между входными переменными (предикторами) и выходной переменной (откликом). Он пытается провести прямую линию через точки данных, которые лучше всего представляют тренд данных.

Применяется в случаях, когда связь между независимой и зависимой переменной является линейной, когда нужно делать прогнозы на основе непрерывных данных, когда нужна простая и интерпретируемая модель

Преимущества: линейная регрессия проста и легко интерпретируется; скорость моделирования высока, поскольку она не требует сложных вычислений и быстро выполняет прогнозы при большом объеме данных.

Недостатки: предполагает линейную зависимость между входными и выходными переменными, что не всегда так; чувствителен к выбросам.

К-ближайших соседей (KNeighborsRegressor) — метод решения задач классификации и задач регрессии. Для регрессии алгоритм k-ближайших соседей использует значения целевой переменной его k ближайших соседей для предсказания значения для нового объекта. Регрессия на основе соседей может использоваться в случаях, когда метки данных являются непрерывными переменными. Метка, присвоенная точке запроса, вычисляется на основе среднего значения меток ее ближайших соседей.

Применяется в случаях, когда набор данных имеет небольшой объем, когда в наборе данных большое количество обучающих примеров.

Преимущества: простота реализации и понимания, не требует сложной математической модели и позволяет легко визуализировать результаты; универсальность, может использоваться для решения задач как классификации, так и регрессии, хорошо работает как с числовыми, так и с категориальными данными; адаптивность к изменяющимся данным, когда поступают новые данные, алгоритм k-ближайших соседей может легко обновить свою модель без перетренировки.

Недостатки: неэффективен для больших наборов данных, поскольку расстояние необходимо вычислять для каждой точки, закликаясь каждый раз, когда алгоритм обнаруживает новую точку данных; модель подвержена выбросам; может быть дорогостоящим в вычислительном отношении, особенно когда пространство признаков велико.

Опорная векторная регрессия (SVR) — это контролируемый алгоритм обучения, который используется для моделирования взаимосвязи между зависимой переменной и одной или несколькими независимыми переменными. Это непараметрический метод, который может обрабатывать как линейные, так и нелинейные связи между переменными. Основная идея SVR состоит в том, чтобы найти гиперплоскость, которая максимизирует разницу между прогнозируемыми значениями и фактическими значениями. Гиперплоскость выбирается таким образом, чтобы она находилась как можно дальше от точек данных. Это достигается путем минимизации ошибки между прогнозируемыми значениями и фактическими значениями. Затем гиперплоскость используется для прогнозирования значений зависимой переменной для новых точек данных.

Одним из основных преимуществ SVR является его способность обрабатывать нелинейные данные. Это связано с тем, что функция ядра позволяет алгоритму отображать данные в многомерное пространство, где их легче разделить. SVR также очень эффективен при работе с зашумленными данными, поскольку он использует погрешность вместо того, чтобы пытаться идеально подогнать данные.

Однако для больших наборов данных это может оказаться затратным в вычислительном отношении.

Регрессия дерева решений (DecisionTreeRegressor) — это тип метода регрессионного анализа в машинном обучении, в котором дерево решений используется для моделирования связи между независимыми переменными и зависимой переменной. Модели регрессии дерева решений делают прогнозы, используя древовидную структуру, основанную на серии простых решений, основанных на функциях в наборе данных. Каждый внутренний узел дерева представляет собой решение, основанное на функциях, а каждый конечный узел представляет собой прогноз.

Преимущества: простота в интерпретации, результаты можно легко понять и интерпретировать, даже без специальных знаний в области машинного обучения; обработка разнотипных данных, дерево решений способно работать с

различными типами переменных; устойчивость к выбросам, дерево решений обычно хорошо справляется с выбросами и шумом в данных, не требуя предварительной нормализации; высокая скорость работы.

Недостатки: деревья решений часто склонны к переобучению, особенно когда у них много уровней или, когда данные имеют сложные зависимости. Они также не позволяют обрабатывать пропущенные значения в данных.

Случайный лес (RandomForestRegressor) — это популярный метод ансамблевого обучения, позволяющий делать более точные прогнозы путем объединения нескольких деревьев решений. Несколько деревьев решений строятся на разных подмножествах обучающих данных в регрессии случайного леса, а окончательный прогноз делается путем усреднения прогнозов всех деревьев.

Применяется: когда есть необходимость обработать недостающие данные, когда есть необходимость обрабатывать выбросы и шум, когда есть потребность в модели, которая хорошо работает с новыми данными

Преимущества: может обрабатывать большие наборы данных с высокой размерностью; может обрабатывать пропущенные значения и поддерживать точность; не требует тщательной настройки параметров, хорошо работает «из коробки»; хорошо работает с пропущенными данными; сохраняет хорошую точность, если большая часть данных пропущена.

Недостатки: в отличие от одного дерева, результаты случайного леса сложнее интерпретировать; алгоритм работает хуже многих линейных методов, когда в выборке очень много разреженных признаков; большой размер получающихся моделей.

Градиентный бустинг (GradientBoostingRegressor) — метод ансамблевого обучения, который объединяет несколько слабых моделей для более точного прогнозирования. Регрессия с повышением градиента строит деревья решений последовательно, при этом каждое дерево пытается исправить ошибки, допущенные предыдущим деревом.

Преимущества Gradient Boosting включают в себя высокую точность прогнозов, устойчивость к переобучению и способность работать с разнородными данными. Он также позволяет эффективно решать задачи как классификации, так и регрессии.

Недостатки: реализация может быть более сложной из-за более высоких вычислительных требований; сложнее интерпретировать.

Стохастический градиентный спуск (SGDRegressor) — это алгоритм машинного обучения, который используется для минимизации функции стоимости путем повторения обновления веса на основе градиентов (обновляет параметры модели, используя только подмножество обучающих данных, называемое мини-пакетом, на каждой итерации). Регрессор SGD уменьшает расхождение между целевыми и ожидаемыми значениями за счет оптимизации параметров модели.

Преимущества: эффективность вычислений, при каждой итерации используется только мини-партия обучающих примеров, что снижает требования к памяти и позволяет выполнять параллельную обработку; подходит для больших наборов данных.

Недостатки: выбор скорости обучения, высокая скорость обучения может привести к расхождению алгоритма, а низкая скорость обучения может привести к медленной сходимости алгоритма; при большой размерности пространства признаков возможно переобучение.

Нейронная сеть — это математическая модель, который использует взаимосвязанные узлы или нейроны в слоистой структуре, напоминающей человеческий мозг. Она создает адаптивную систему, с помощью которой компьютеры учатся на своих ошибках и постоянно совершенствуются.

Основные достоинства нейронных сетей: высокая производительность; работа с неструктурированными данными; адаптивность к изменяющимся условиям, самообучаются и продолжают корректно работать.

Недостатки: вычислительные ресурсы, нейросети требуют большого объема вычислительных ресурсов, включая мощные процессоры и графические

ускорители, обучение сложных моделей может быть очень времязатратным и дорогостоящим процессом; ограниченная применимость, могут быть неэффективными или неприменимыми в некоторых задачах, особенно если данных недостаточно или задача слишком проста для них; зависимость от данных, недостаточные или неточные данные могут привести к неверным или неудовлетворительным результатам

2.3 Разведочный анализ данных

Для разведочного анализа данных использовались:

- описательная статистика: позволяет получить обобщенное представление об основных характеристиках (среднее значение, стандартное отклонение, минимальное и максимальное значение и др.);

	Количество	Среднее значение	Стандартное отклонение	Минимальное значение	25-й процентиль, 1-й квартиль	Медианное значение	75-й процентиль, 3-й квартиль	Максимальное значение
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп, %_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Рисунок 21 - описательная статистика

- проверка наличия пропусков и дубликатов;

<pre>#npnull dataset.isnull().sum()</pre>	
Соотношение матрица-наполнитель	0
Плотность, кг/м3	0
модуль упругости, ГПа	0
Количество отвердителя, м.%	0
Содержание эпоксидных групп,%_2	0
Температура вспышки, C_2	0
Поверхностная плотность, г/м2	0
Модуль упругости при растяжении, ГПа	0
Прочность при растяжении, МПа	0
Потребление смолы, г/м2	0
Угол нашивки, град	0
Шаг нашивки	0
Плотность нашивки	0
dtype: int64	
<pre># Наличие дубликатов dataset.duplicated().sum()</pre>	
0	

Рисунок 32 – дубликаты и пропуски

Дубликаты и пропуски отсутствуют.

- анализ выбросов (применялись визуализация данных и статистический анализ (метод стандартных отклонений, метод межквартильных расстояний));

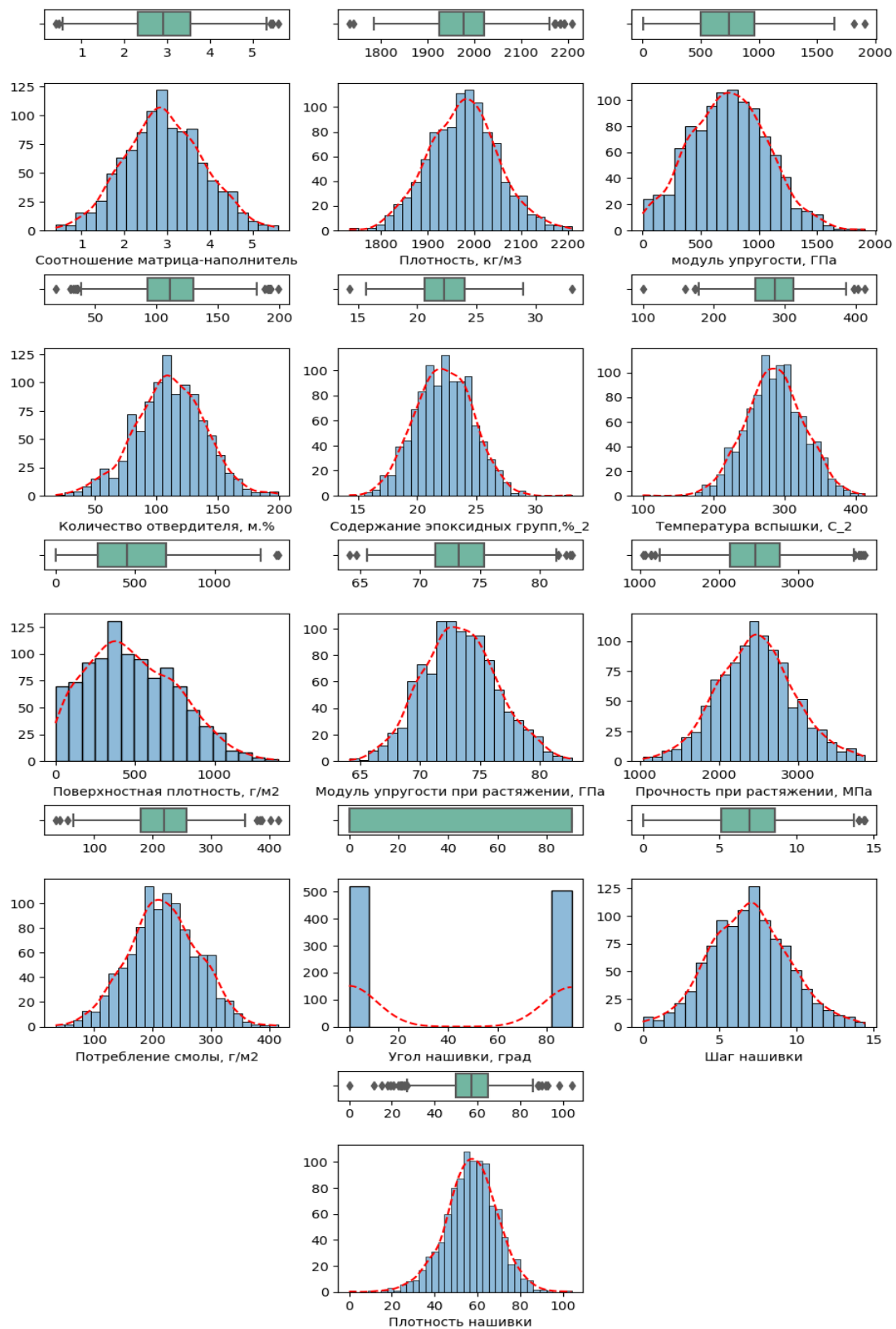


Рисунок 43 – гистограммы распределения каждой из переменной, диаграммы ящика с усами

Практически все параметры датасета имеют нормальное распределение.

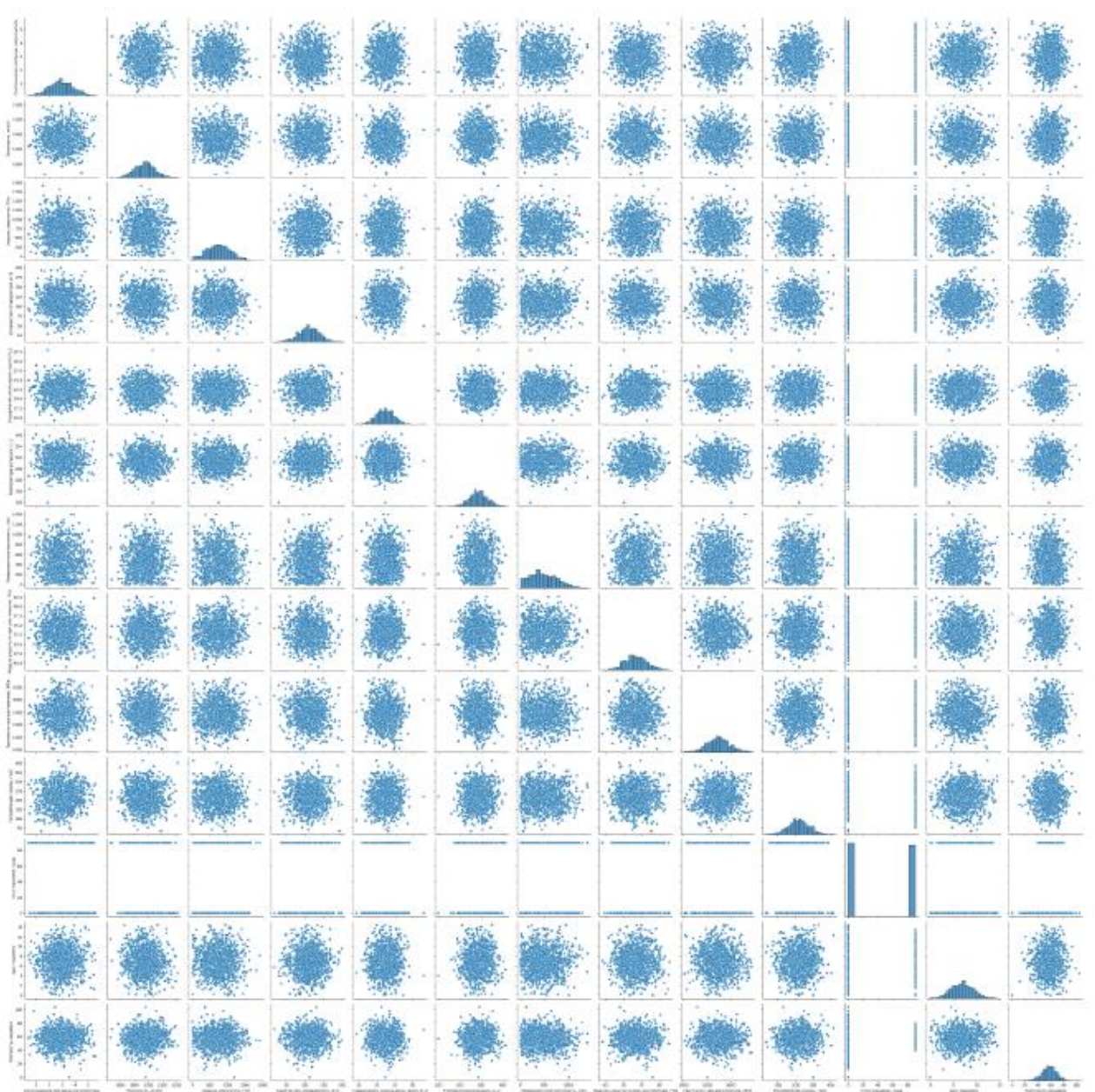


Рисунок 5 – Попарные графиков рассеяния точек

– преобразование данных (нормализация): в качестве нормализатора применялся `MinMaxScaler`, позволяющая привести числовых переменные к диапазону от 0 до 1;

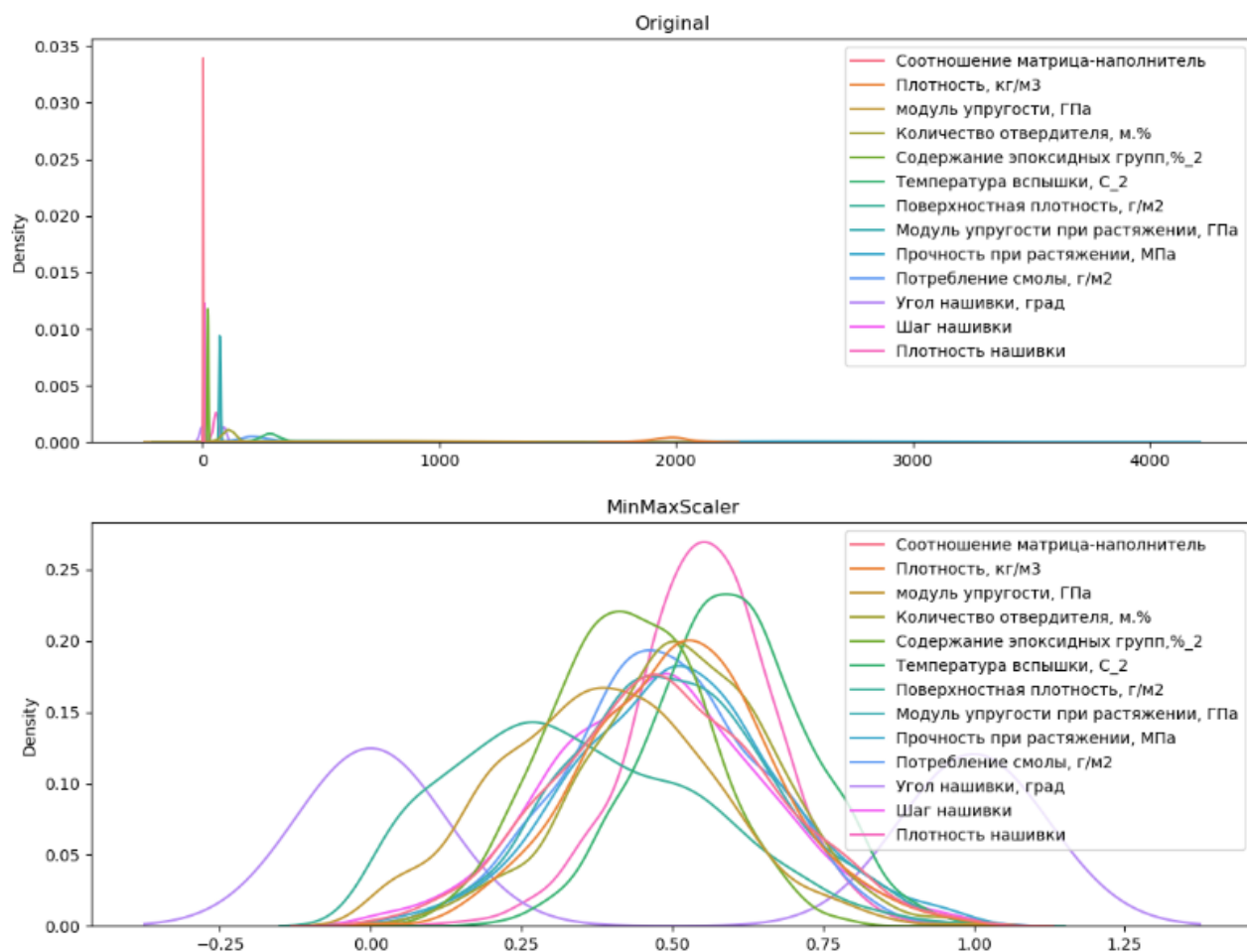


Рисунок 6 - данные до и после нормализации

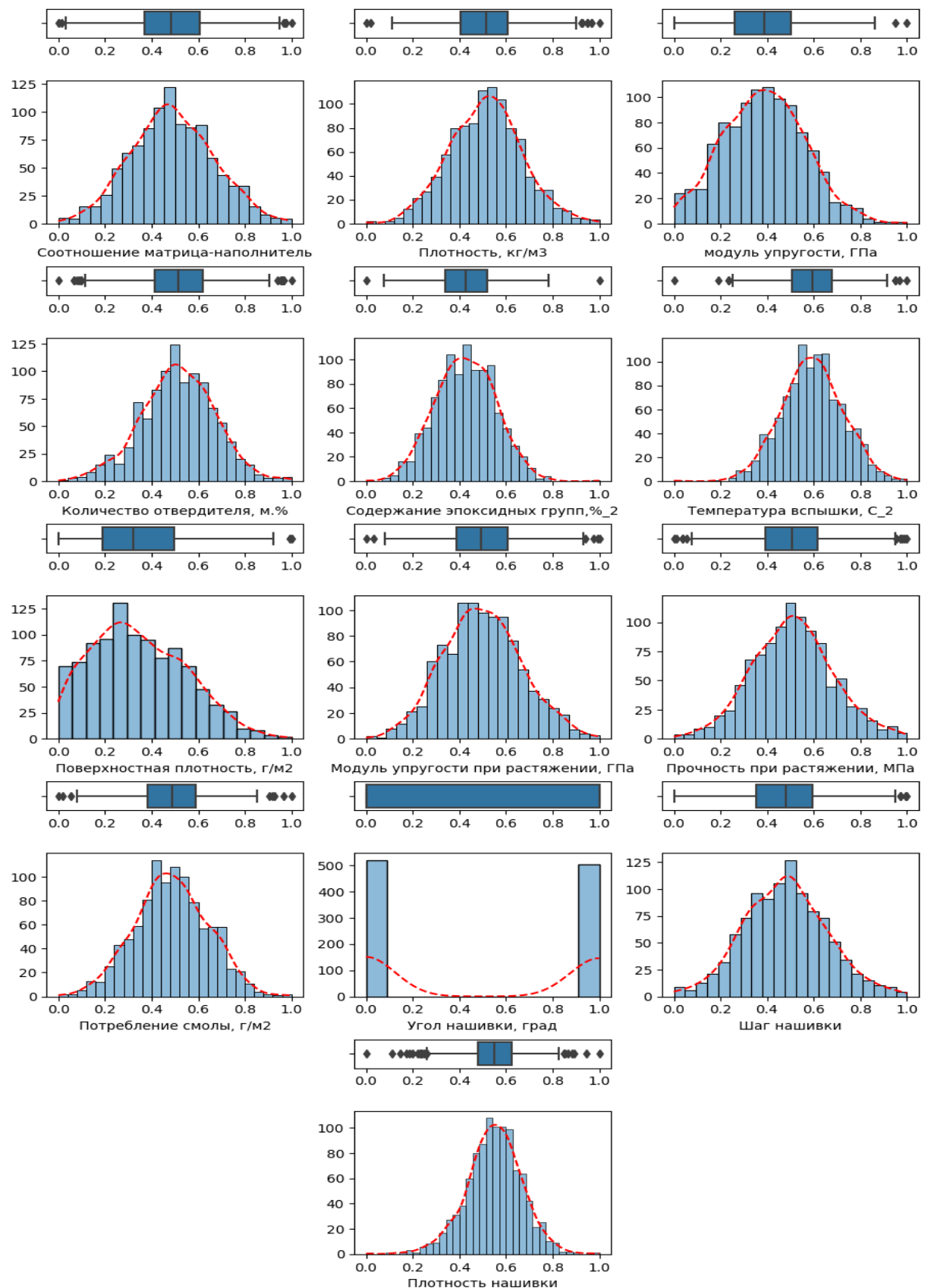


Рисунок 74 – гистограммы распределения каждой из переменной, диаграммы ящика с усами (после нормализации)

– корреляционный анализ: помогает понять, какие переменные взаимосвязаны между собой и насколько сильна эта связь (коэффициент корреляции измеряет степень линейной зависимости между двумя переменными), применялась тепловая карта (графическое представление матрицы данных, где цветовая шкала показывает степень взаимосвязи между переменными).

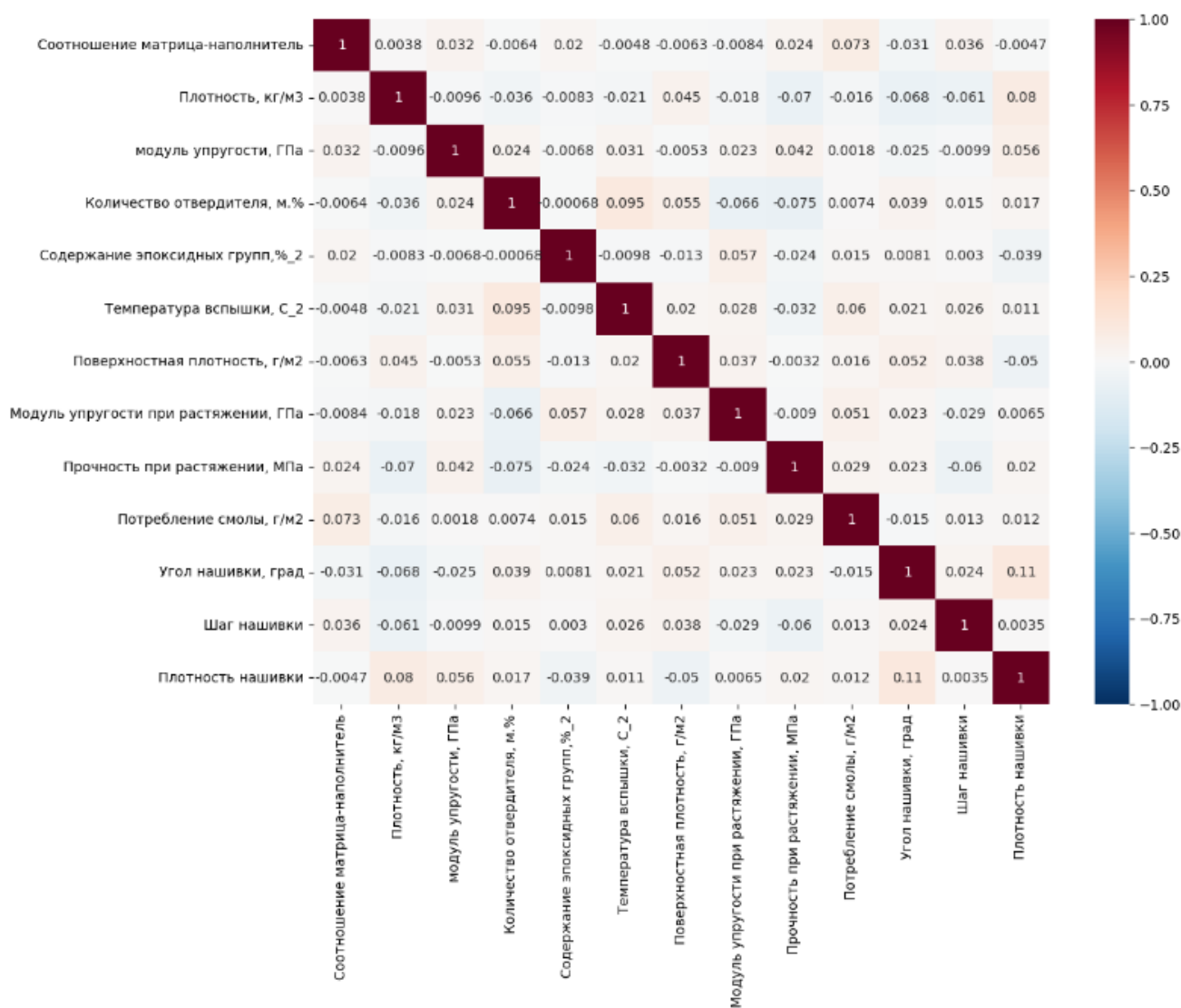


Рисунок 85 – тепловая карта коэффициентов корреляции

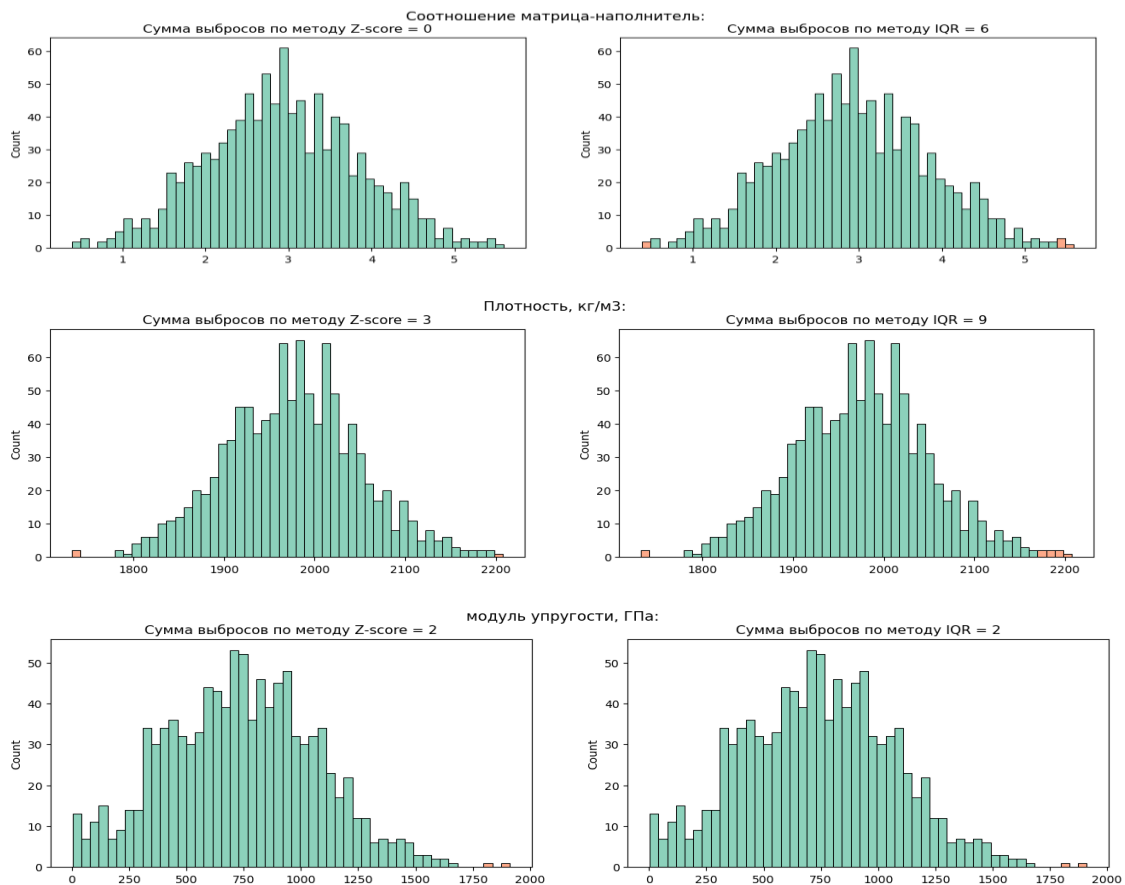
Параметры датасета не имеют чётко выраженной зависимости, корреляция между всеми параметрами слабая.

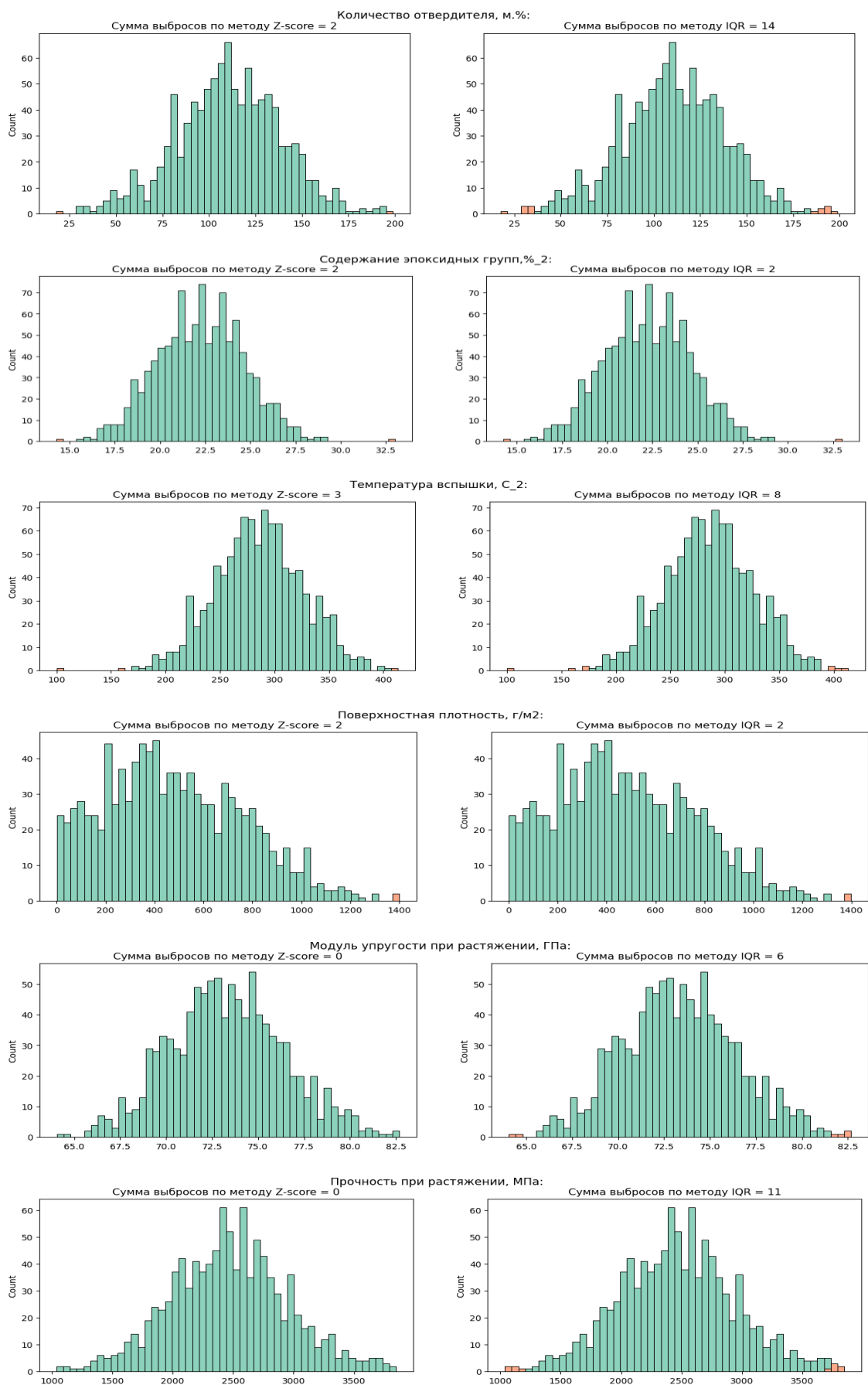
3 Практическая часть

3.1 Предобработка данных

Число обнаруженных выбросов:

	Метод стандартных отклонений(Z-score)	Метод межквартильных расстояний (IQR)
Соотношение матрица-наполнитель	0	6
Плотность, кг/м3	3	9
модуль упругости, ГПа	2	2
Количество отвердителя, м.%	2	14
Содержание эпоксидных групп,%_2	2	2
Температура вспышки, С_2	3	8
Поверхностная плотность, г/м2	2	2
Модуль упругости при растяжении, ГПа	0	6
Прочность при растяжении, МПа	0	11
Потребление смолы, г/м2	3	8
Угол нашивки, град	0	0
Шаг нашивки	0	4
Плотность нашивки	7	21
ИТОГО	24	93





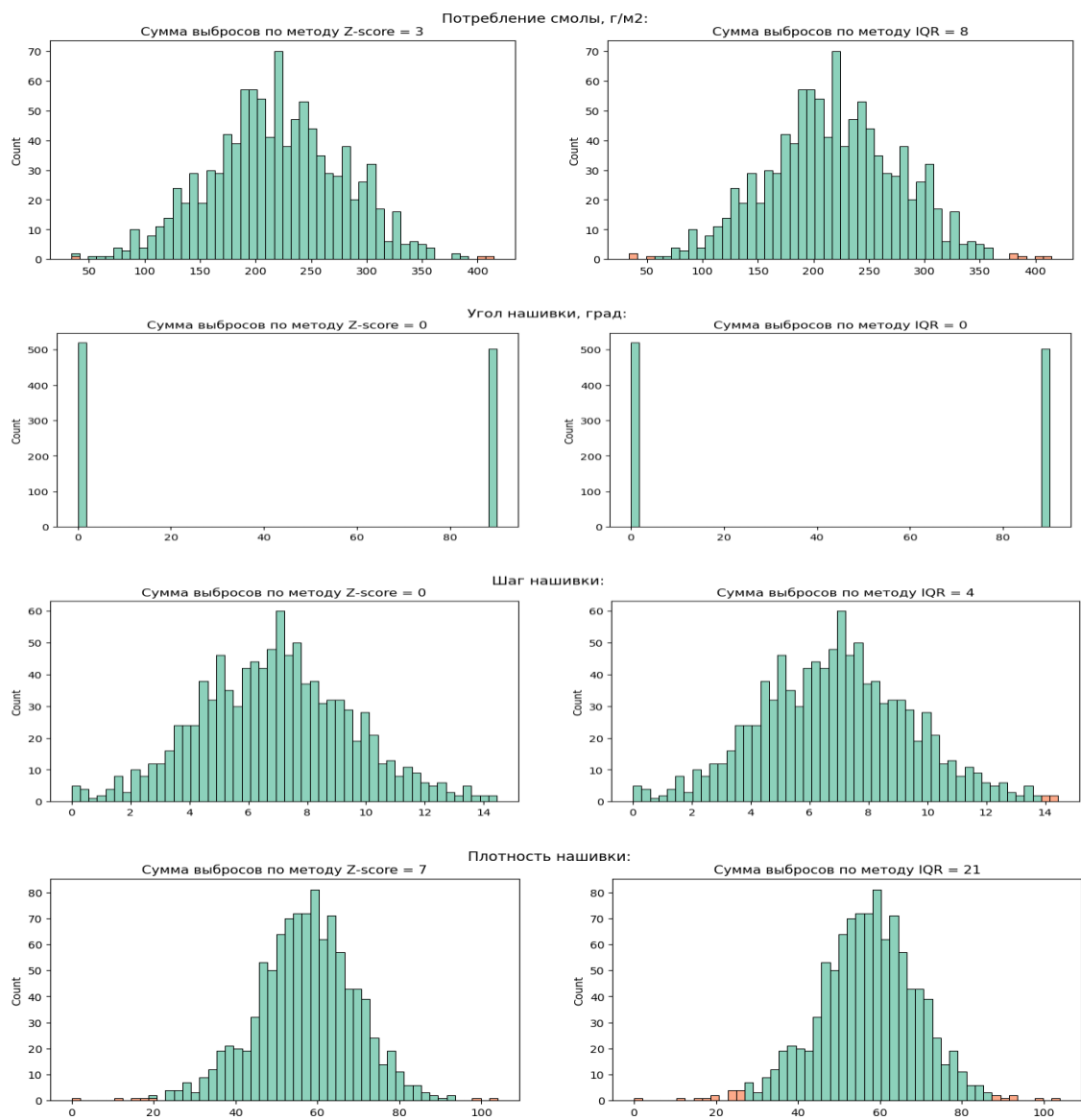


Рисунок 9 – поиск выбросов в датасете методами стандартных отклонений (Z-score) и межквартильных расстояний (IQR)

Число обнаруженных выбросов:
Метод стандартных отклонений (Z-score) - 24
Выбросы после очистки - 3
Размер очищенного датасета: (999, 13)

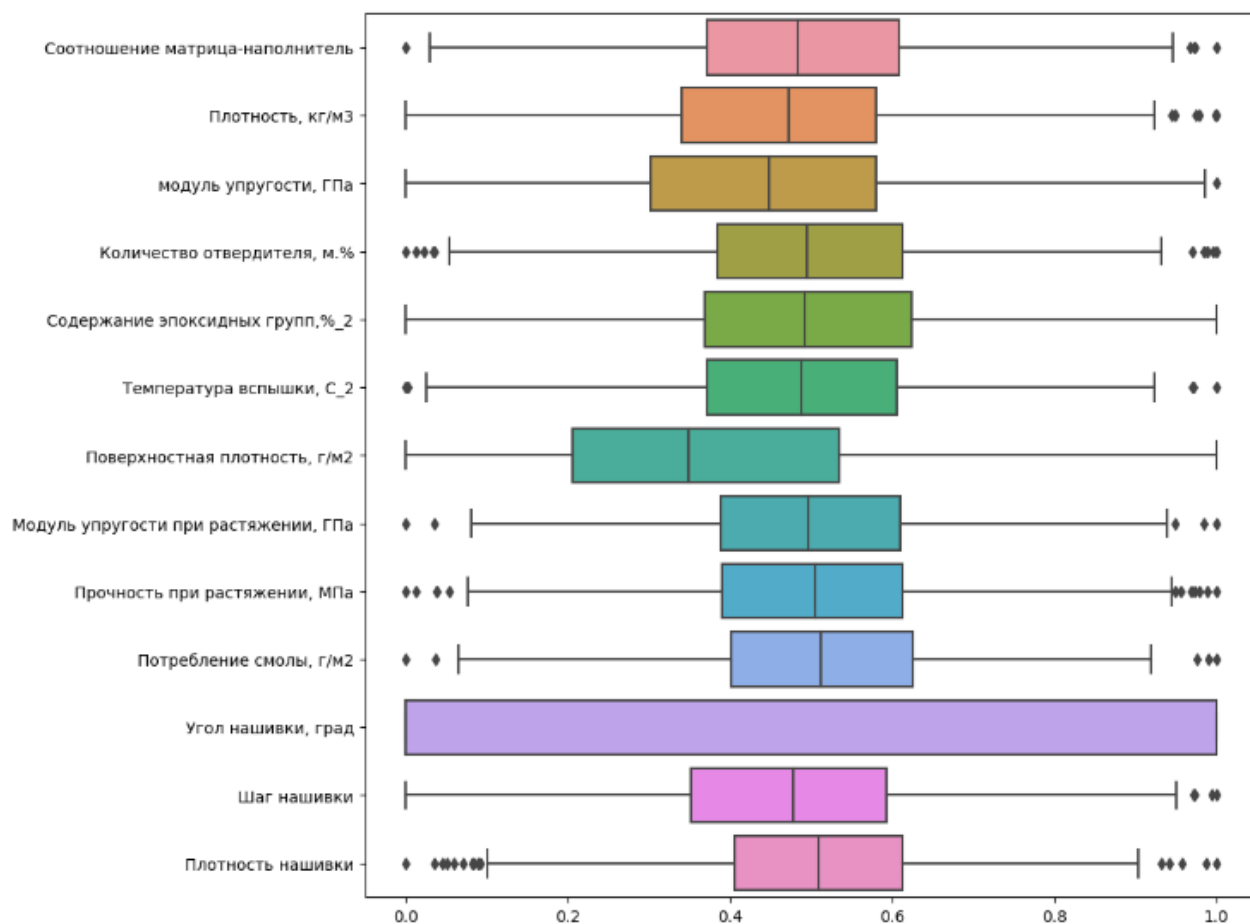


Рисунок 10 – ящик с усами после выявления и удаления выбросов
методом стандартных отклонений (Z-score)

Число обнаруженных выбросов:
Метод межквартильных расстояний (IQR) - 93
Выбросы после очистки - 10
Размер очищенного датасета: (936, 13)

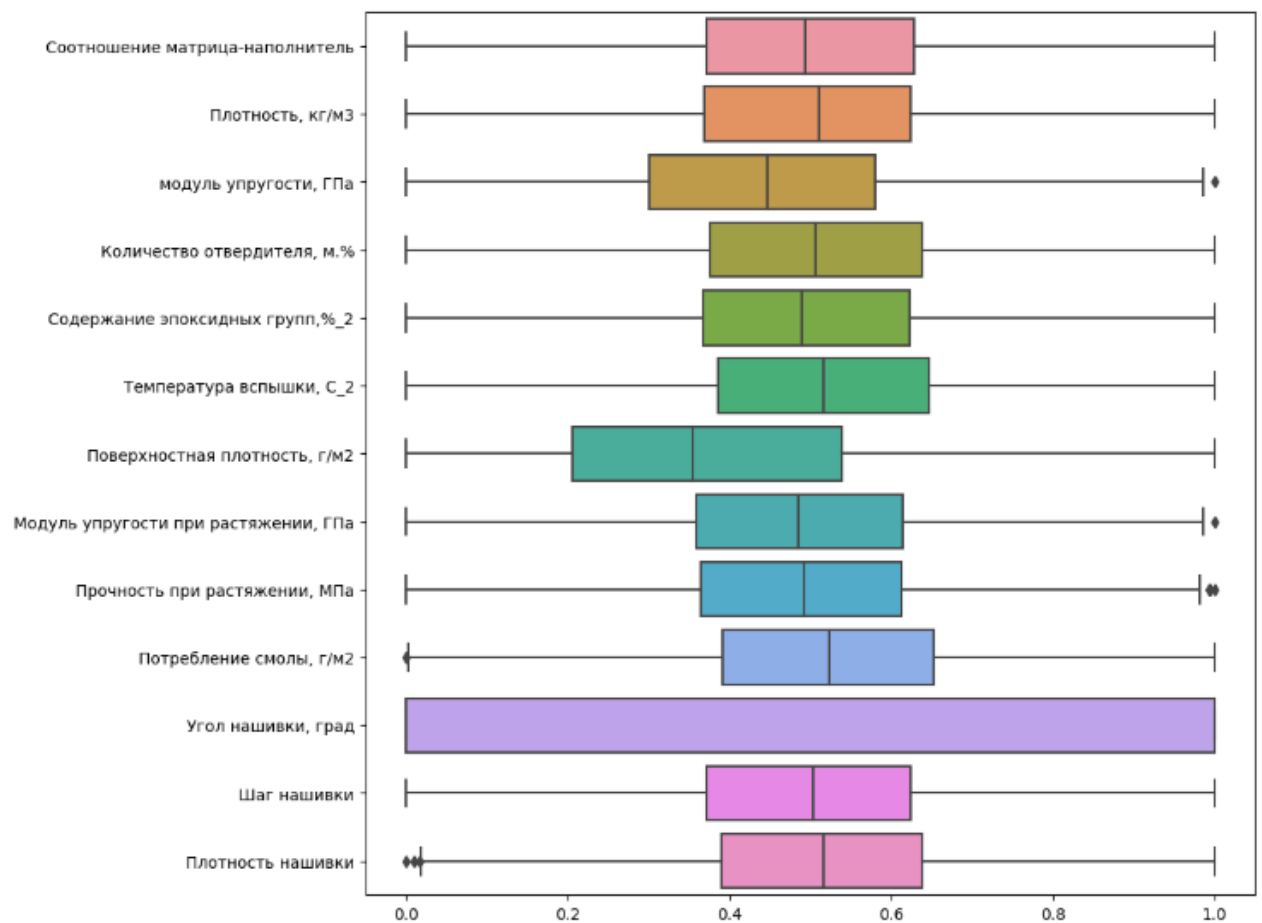


Рисунок 11 – ящик с усами после выявления и удаления выбросов методом межквартильных расстояний (IQR)

3.2 Разработка и обучение модели

Разработка и обучение моделей машинного обучения производились для двух параметров: «Модуль упругости при растяжении» и «Прочность при растяжении».

Для прогноза каждого из параметров использованы следующие модели:

- Линейная регрессия (LinearRegression);
- К-ближайших соседей (KNeighborsRegressor);
- Метод опорных векторов (SVR);
- Регрессия дерева решений (DecisionTreeRegressor);

- Случайный лес (RandomForestRegressor),
- Градиентный бустинг (GradientBoostingRegressor),
- Стохастический градиентный спуск (SGDRegressor),
- Нейронная сеть.

Выборка по каждому из параметров была разделена на обучающую (70%) и тестовую (30%).

Для оценки качества модели использовались следующие метрики:

- коэффициент детерминации (R^2);
- средняя абсолютная ошибка (MAE);
- среднеквадратичная ошибка (MSE);
- корень из среднеквадратичной ошибки (RMSE);
- средняя абсолютная ошибка в процентах (MAPE).

```

# Обучение моделей
def model(x, y, models):
    name = y.name
    print(f'\033[1mМодели для прогноза показателя "{name}" и результаты обучения\n')

    x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                         test_size=0.3,
                                                         random_state=42)

    model_learning = pd.DataFrame()
    cross = pd.DataFrame()
    print(f'\033[0mРазмер обучающей выборки: {X_train.shape[0]}')
    print(f'\033[0mРазмер тестовой выборки: {X_test.shape[0]}')
    for model_name, model in models.items():
        pipe = make_pipeline(MinMaxScaler(), model)
        start = time()
        pipe.fit(X_train, y_train)
        train_time = time() - start
        start = time()
        y_pred = pipe.predict(X_test)
        predict_time = time() - start

        plt.figure(figsize = (10, 2))
        plt.title(model_name)
        plt.plot(y_pred, label = "прогноз", color = "orange")
        plt.plot(y_test.values, label = "тест", color = "green")
        plt.ylabel('', fontsize=7)
        plt.xlabel('Количество наблюдений', fontsize=7)
        plt.legend(loc='upper right')
        plt.grid(True)

        scoring_list = {
            'R2': ['{:.2f}'.format(r2_score(y_true=y_test, y_pred=y_pred))],
            'MAE': ['{:.2f}'.format(mean_absolute_error(y_test, y_pred))],
            'MSE': ['{:.2f}'.format(mean_squared_error(y_test, y_pred))],
            'RMSE': ['{:.2f}'.format(mean_squared_error(y_test, y_pred, squared=False))],
            'MAPE': ['{:.2f}'.format(mean_absolute_percentage_error(y_true=y_test, y_pred=y_pred))],
        }
        model_learning.loc[model_name, ['Training time']] = "%0.3fs" % train_time
        model_learning.loc[model_name, ['Predict time']] = "%0.3fs" % predict_time

        for scoring_name, scoring in scoring_list.items():
            model_learning.loc[model_name, [scoring_name]] = scoring
        for score_name, score in score_list.items():
            kfold = KFold(n_splits=10, random_state=42, shuffle=True)
            cross_val = cross_val_score(estimator=pipe, X=X_train, y=y_train, cv=kfold, scoring=score)
            cross.loc[model_name, score_name] = '{:.2f}'.format(abs(cross_val.mean()) if score_name != "R2" else cross_val.mean())
        display(model_learning.style.highlight_max(color = 'lightgreen').highlight_min(color = 'lightblue'))
    print('\n\033[1mРезультаты кросс-валидации')
    display(cross.style.highlight_max(color = 'lightgreen', axis = 0).highlight_min(color = 'lightblue', axis = 0))

```

Рисунок 12 – обучение модели

3.3 Тестирование модели

В рамках тестирования модели проведено сравнение расчетов показателей на датасете:

- без удаления выбросов;
- с удалением выбросов по методу стандартных отклонений (Z-score);
- с удалением выбросов по методу межквартильных расстояний (IQR).

Разработка и обучение модели для показателя «Модуль упругости при растяжении».

Модели для прогноза показателя "Модуль упругости при растяжении, ГПа" и результаты обучения

Размер обучающей выборки: 716
Размер тестовой выборки: 307

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.000s	-0.02	2.56	10.12	3.18	0.03
KNeighborsRegressor	0.000s	0.016s	-0.27	2.83	12.66	3.56	0.04
SVR	0.024s	0.020s	-0.05	2.59	10.44	3.23	0.04
DecisionTreeRegressor	0.031s	0.000s	-1.14	3.67	21.25	4.61	0.05
RandomForestRegressor	2.067s	0.000s	-0.09	2.62	10.83	3.29	0.04
GradientBoostingRegressor	0.729s	0.000s	-0.11	2.65	10.98	3.31	0.04
SGDRegressor	0.069s	0.000s	-0.05	2.61	10.45	3.23	0.04

Результаты кросс-валидации

	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.04	2.51	9.76	3.12	0.03
KNeighborsRegressor	-0.24	2.72	11.67	3.41	0.04
SVR	-0.05	2.51	9.86	3.13	0.03
DecisionTreeRegressor	-1.26	3.67	20.79	4.56	0.05
RandomForestRegressor	-0.09	2.53	10.11	3.17	0.03
GradientBoostingRegressor	-0.13	2.61	10.71	3.27	0.04
SGDRegressor	-0.11	2.60	10.39	3.22	0.04

CPU times: total: 2min 12s
Wall time: 2min 14s

Рисунок 13 – обучение модели без удаления выбросов

Модели для прогноза показателя "Модуль упругости при растяжении, ГПа" и результаты обучения

Размер обучающей выборки: 699
Размер тестовой выборки: 300

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.000s	-0.03	2.52	9.55	3.09	0.03
KNeighborsRegressor	0.000s	0.016s	-0.14	2.65	10.83	3.26	0.04
SVR	0.027s	0.020s	-0.05	2.53	9.73	3.12	0.03
DecisionTreeRegressor	0.038s	0.000s	-1.29	3.73	21.28	4.61	0.05
RandomForestRegressor	2.128s	0.016s	-0.08	2.56	10.03	3.17	0.04
GradientBoostingRegressor	0.722s	0.000s	-0.12	2.60	10.38	3.22	0.04
SGDRegressor	0.050s	0.000s	-0.10	2.62	10.20	3.19	0.04

Результаты кросс-валидации

	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.05	2.52	9.96	3.14	0.03
KNeighborsRegressor	-0.28	2.78	11.95	3.46	0.04
SVR	-0.07	2.53	10.06	3.16	0.03
DecisionTreeRegressor	-1.41	3.83	23.11	4.70	0.05
RandomForestRegressor	-0.11	2.59	10.57	3.22	0.03
GradientBoostingRegressor	-0.19	2.64	11.19	3.33	0.04
SGDRegressor	-0.10	2.56	10.42	3.22	0.04

CPU times: total: 2min 12s
Wall time: 2min 17s

Рисунок 14 – обучение модели с удалением выбросов по методу стандартных отклонений (Z-score)

Модели для прогноза показателя "Модуль упругости при растяжении, ГПа" и результаты обучения

Размер обучающей выборки: 655

Размер тестовой выборки: 281

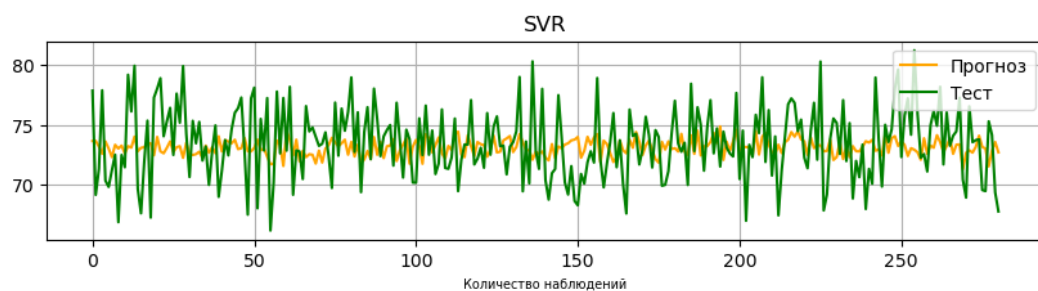
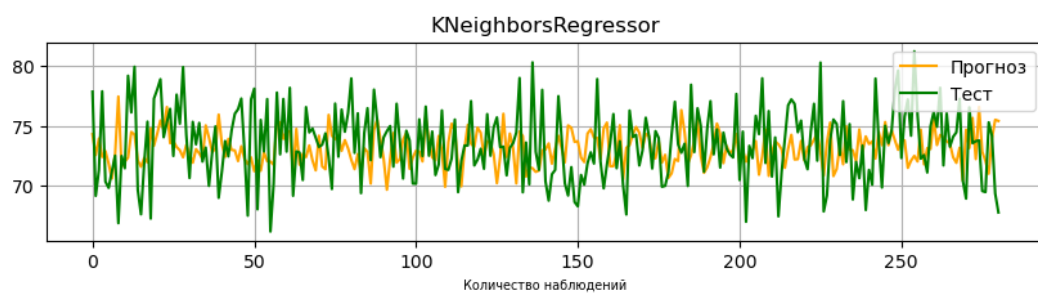
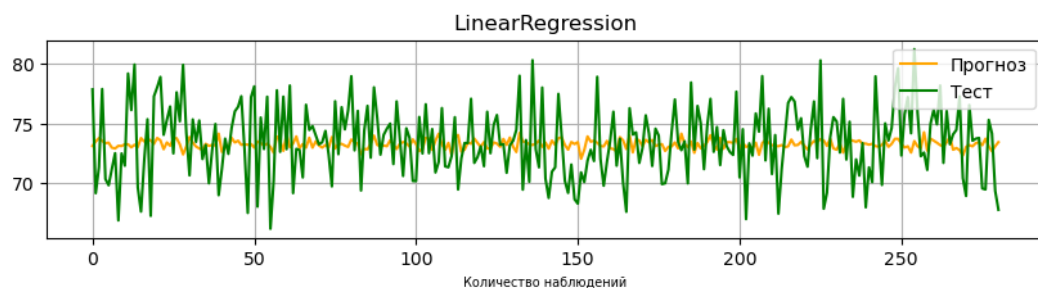
	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.003s	0.002s	-0.01	2.41	8.78	2.98	0.03
KNeighborsRegressor	0.000s	0.000s	-0.21	2.84	10.53	3.24	0.04
SVR	0.018s	0.031s	-0.03	2.44	8.95	2.99	0.03
DecisionTreeRegressor	0.011s	0.018s	-1.28	3.81	19.74	4.44	0.05
RandomForestRegressor	1.982s	0.018s	0.00	2.40	8.71	2.95	0.03
GradientBoostingRegressor	0.881s	0.000s	-0.11	2.49	9.84	3.10	0.03
SGDRegressor	0.050s	0.000s	-0.05	2.47	9.13	3.02	0.03

Результаты кросс-валидации

	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.04	2.60	9.69	3.10	0.03
KNeighborsRegressor	-0.27	2.75	11.75	3.42	0.04
SVR	-0.05	2.61	9.74	3.11	0.03
DecisionTreeRegressor	-1.21	3.68	20.51	4.51	0.05
RandomForestRegressor	-0.08	2.55	10.09	3.20	0.03
GradientBoostingRegressor	-0.14	2.59	10.58	3.24	0.04
SGDRegressor	-0.09	2.53	10.14	3.17	0.03

CPU times: total: 2min

Wall time: 2min 7s



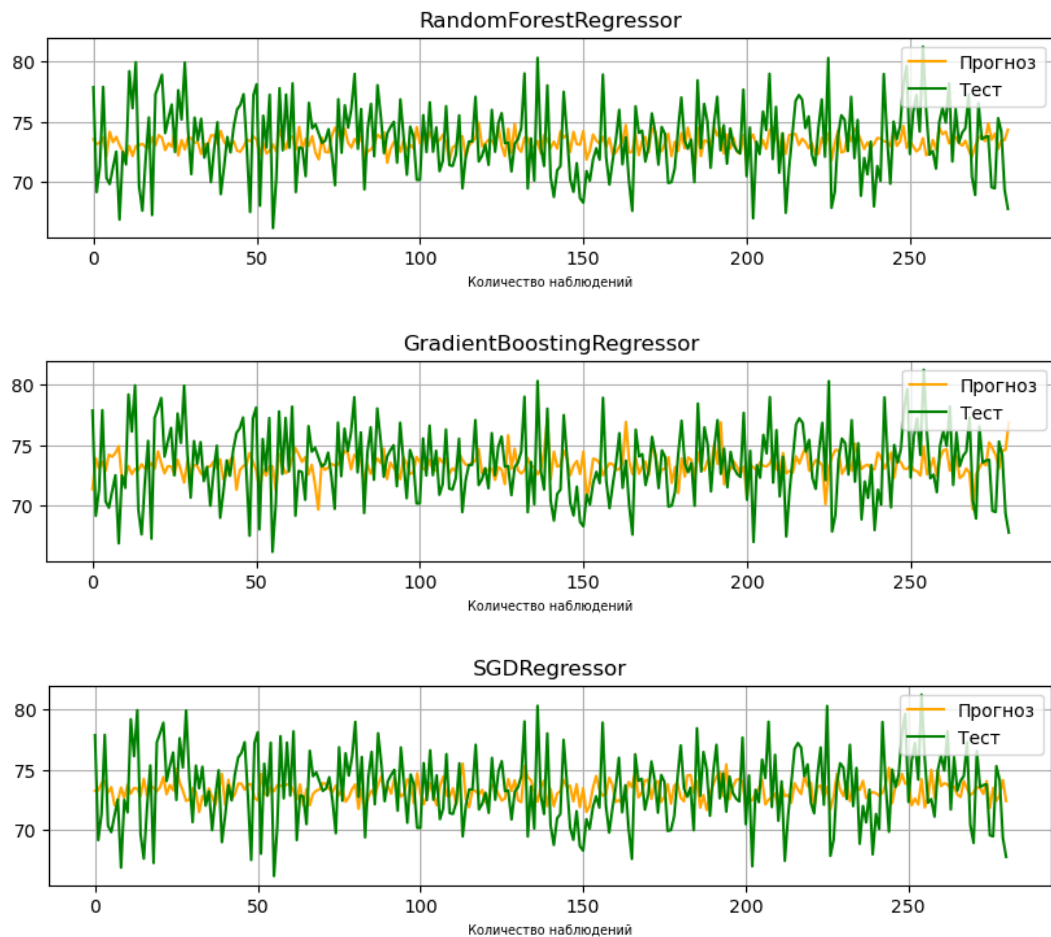


Рисунок 15 – обучение модели с удалением выбросов по методу межквартильных расстояний (IQR)

```
# Применение Нейронной сети с выбросами iqr
y = dataset_clean_iqr['Модуль упругости при растяжении, ГПа']
x = dataset_clean_iqr.drop(columns=['Модуль упругости при растяжении, ГПа'])

X_train, X_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.3,
                                                    random_state=42)

normalizer = Normalization(input_shape=[12,], axis=None)
normalizer.adapt(np.array(X_train))
```

Рисунок 16 – применение нейронной сети с удалением выбросов по методу межквартильных расстояний (IQR)

```

model_1 = Sequential([
    normalizer,
    Dense(8, activation='relu'),
    Dense(6, activation='relu'),
    Dense(3, activation='relu'),
    Dense(1)
])
model_1.summary()

```

Model: "sequential"

Layer (type)	Output Shape	Param #
normalization (Normalizati on)	(None, 12)	3
dense (Dense)	(None, 8)	104
dense_1 (Dense)	(None, 6)	54
dense_2 (Dense)	(None, 3)	21
dense_3 (Dense)	(None, 1)	4

=====
 Total params: 186 (748.00 Byte)
 Trainable params: 183 (732.00 Byte)
 Non-trainable params: 3 (16.00 Byte)

Рисунок 17 – параметры model_1

y_test значение 359 77.825677
 Name: Модуль упругости при растяжении, ГПа, dtype: float64
 Предсказательное значение: [[66.05327]]

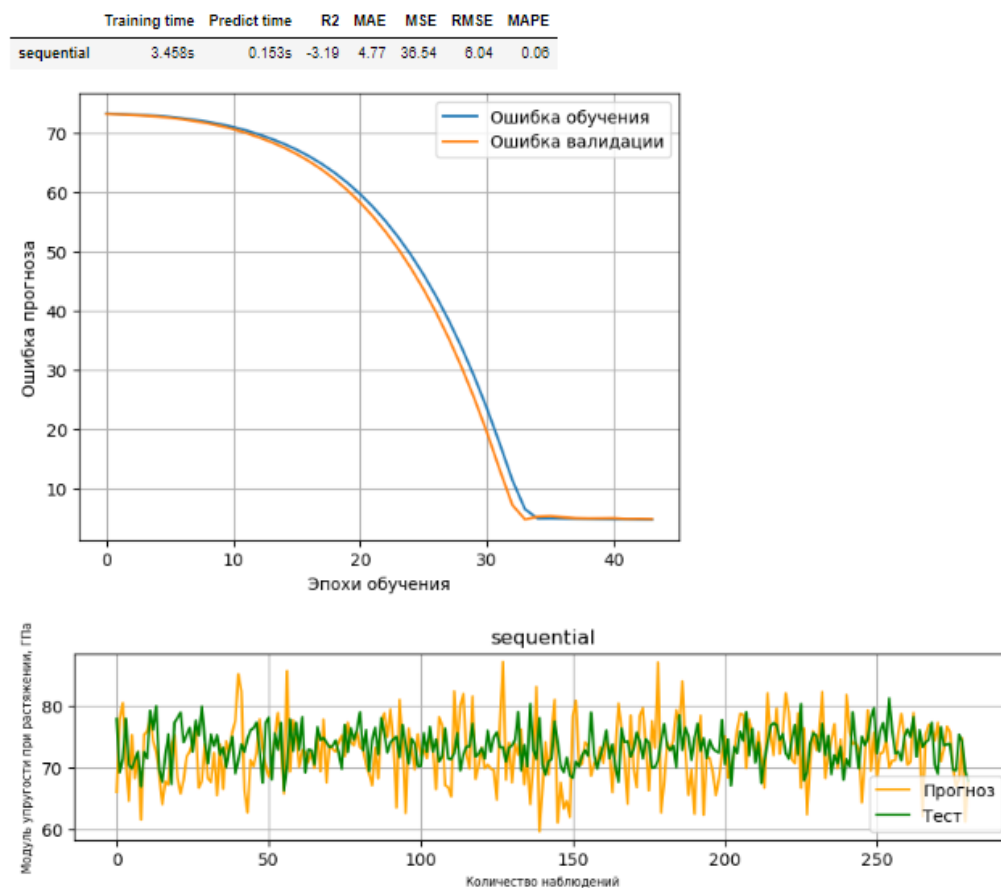


Рисунок 18 – результаты применения нейронной сети (model_1)

```

model_2 = Sequential([
    normalizer,
    Dense(3, activation='relu'),
    Dense(3, activation='linear'),
    Dense(3, activation='linear'),
    Dense(3, activation='linear'),
    Dense(3, activation='linear'),
    Dense(3, activation='linear'),
    Dense(1)
])
model_2.summary()

```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
normalization (Normalization)	(None, 12)	3
dense_4 (Dense)	(None, 3)	39
dense_5 (Dense)	(None, 3)	12
dense_6 (Dense)	(None, 3)	12
dense_7 (Dense)	(None, 3)	12
dense_8 (Dense)	(None, 3)	12
dense_9 (Dense)	(None, 3)	12
dense_10 (Dense)	(None, 1)	4

Total params: 106 (428.00 Byte)
 Trainable params: 103 (412.00 Byte)
 Non-trainable params: 3 (16.00 Byte)

Рисунок 19 – параметры model_2

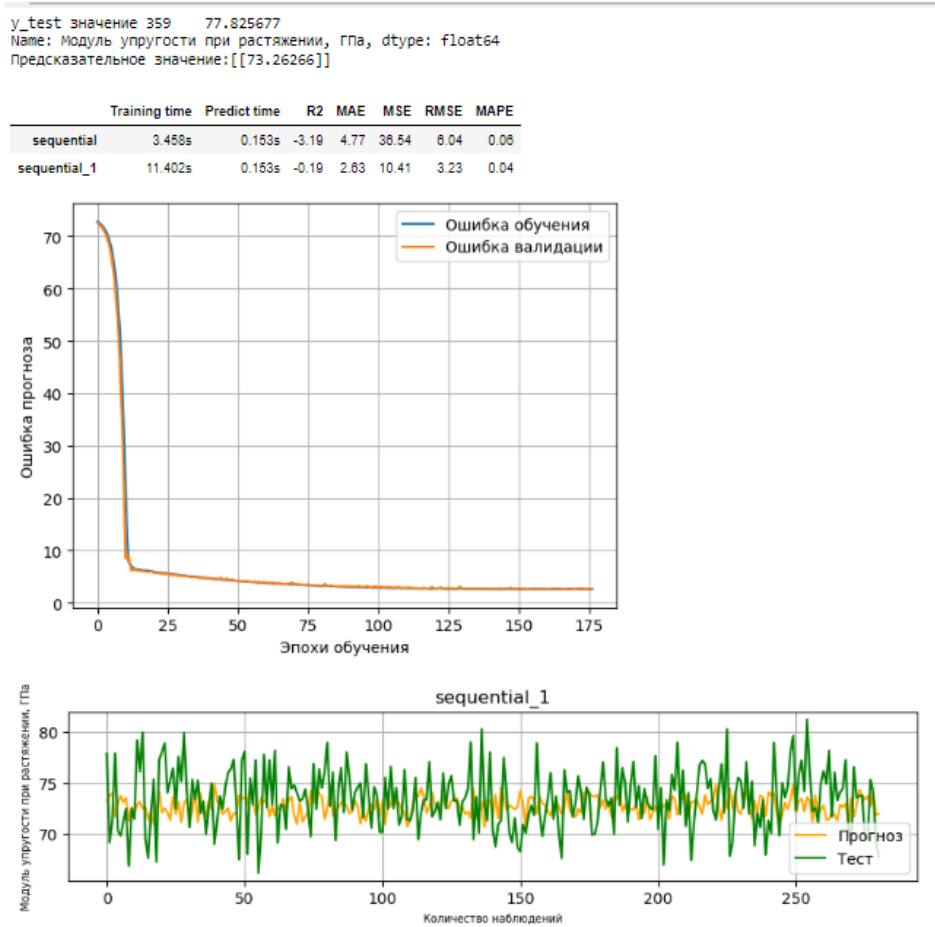


Рисунок 20 – результаты применения нейронной сети (model_1)

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.002s	0.000s	-0.01	2.41	8.78	2.96	0.03
KNeighborsRegressor	0.009s	0.006s	-0.21	2.64	10.53	3.24	0.04
SVR	0.023s	0.017s	-0.03	2.44	8.95	2.99	0.03
DecisionTreeRegressor	0.035s	0.002s	-1.10	3.51	18.28	4.28	0.05
RandomForestRegressor	1.934s	0.012s	-0.02	2.41	8.91	2.99	0.03
GradientBoostingRegressor	0.698s	0.002s	-0.11	2.49	9.63	3.10	0.03
SGDRegressor	0.036s	0.000s	-0.04	2.46	9.06	3.01	0.03
sequential	3.458s	0.153s	-3.19	4.77	36.54	6.04	0.06
sequential_1	11.402s	0.153s	-0.19	2.63	10.41	3.23	0.04

Рисунок 21 – общие результаты обучения моделей для показателя «Модуль упругости при растяжении»

Разработка и обучение модели для показателя «Прочность при растяжении, МПа».

Модели для прогноза показателя "Прочность при растяжении, МПа" и результаты обучения							
Размер обучающей выборки: 716							
Размер тестовой выборки: 397							
	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.003s	0.002s	0.00	379.79	222990.58	472.22	0.17
KNeighborsRegressor	0.000s	0.000s	-0.19	404.98	265867.50	515.43	0.18
SVR	0.031s	0.007s	-0.00	379.60	224399.41	473.71	0.17
DecisionTreeRegressor	0.031s	0.000s	-0.93	515.15	432718.64	657.81	0.22
RandomForestRegressor	2.241s	0.000s	-0.01	378.11	225407.65	474.77	0.17
GradientBoostingRegressor	0.745s	0.002s	-0.07	394.54	239473.91	489.36	0.17
SGDRegressor	0.016s	0.000s	-0.08	392.61	240967.79	490.88	0.17

Результаты кросс-валидации						
	R2	MAE	MSE	RMSE	MAPE	
LinearRegression	-0.03	388.92	245120.25	494.34	0.17	
KNeighborsRegressor	-0.21	426.88	288409.72	535.62	0.19	
SVR	-0.01	384.82	241019.60	490.41	0.17	
DecisionTreeRegressor	-1.14	572.07	520415.05	713.63	0.24	
RandomForestRegressor	-0.03	392.50	252834.30	499.81	0.17	
GradientBoostingRegressor	-0.10	406.80	263665.68	512.17	0.18	
SGDRegressor	-0.07	396.93	254849.57	503.89	0.17	

CPU times: total: 2min 15s						
Wall time: 2min 22s						

Рисунок 22 – обучение модели без удаления выбросов

Модели для прогноза показателя "Прочность при растяжении, МПа" и результаты обучения

Размер обучающей выборки: 699

Размер тестовой выборки: 300

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.000s	-0.01	380.40	227787.17	477.27	0.17
KNeighborsRegressor	0.000s	0.000s	-0.27	423.31	286330.43	535.10	0.19
SVR	0.031s	0.016s	-0.00	375.44	225299.04	474.66	0.17
DecisionTreeRegressor	0.038s	0.000s	-0.92	523.08	433110.65	658.11	0.23
RandomForestRegressor	2.162s	0.000s	-0.04	390.18	234911.56	484.68	0.17
GradientBoostingRegressor	0.712s	0.000s	-0.09	392.72	245790.53	495.77	0.17
SGDRegressor	0.031s	0.000s	-0.01	379.60	228514.44	478.03	0.17

Результаты кросс-валидации

	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.03	389.74	242072.85	490.01	0.17
KNeighborsRegressor	-0.20	417.82	283102.33	529.71	0.18
SVR	-0.01	384.50	238186.74	485.75	0.17
DecisionTreeRegressor	-1.06	554.30	468836.33	693.17	0.24
RandomForestRegressor	-0.06	399.92	250864.45	497.26	0.17
GradientBoostingRegressor	-0.09	400.33	257358.39	504.02	0.17
SGDRegressor	-0.06	396.24	253310.16	495.94	0.17

CPU times: total: 2min 11s

Wall time: 2min 19s

Рисунок 23 – обучение модели с удалением выбросов по методу стандартных отклонений (Z-score)

Модели для прогноза показателя "Прочность при растяжении, МПа" и результаты обучения

Размер обучающей выборки: 655

Размер тестовой выборки: 281

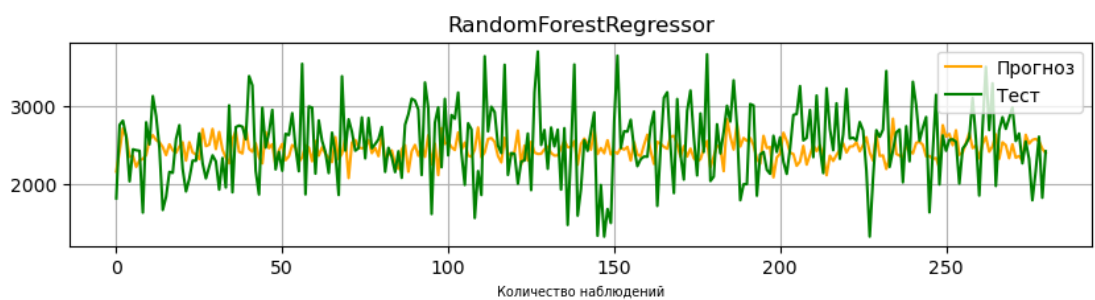
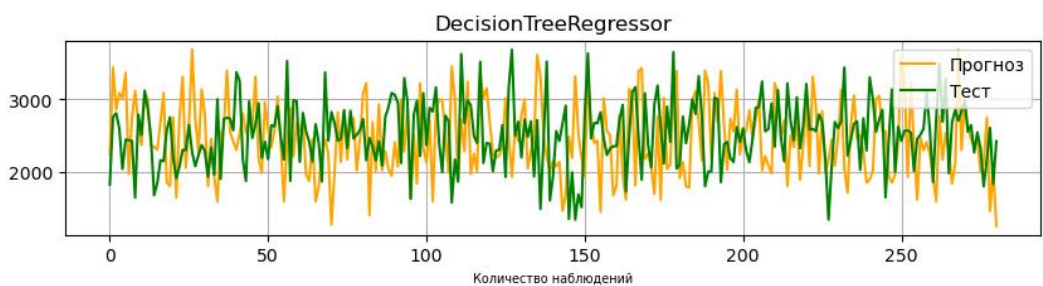
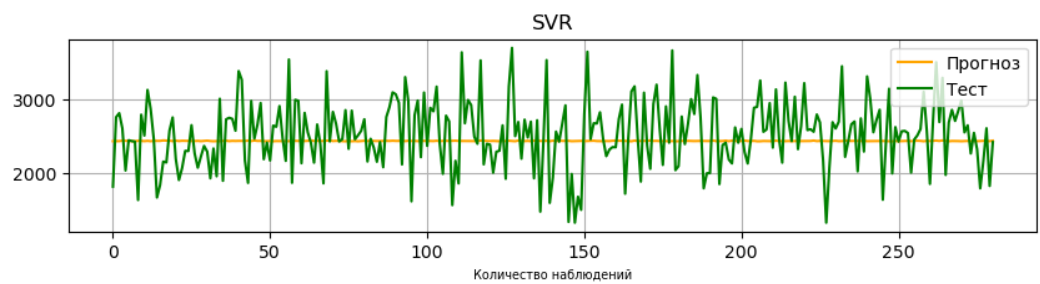
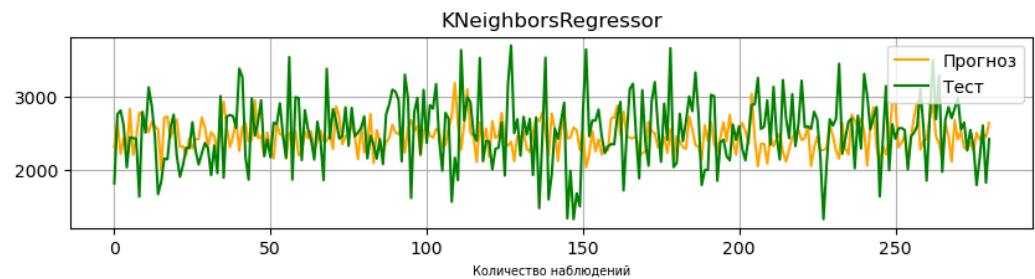
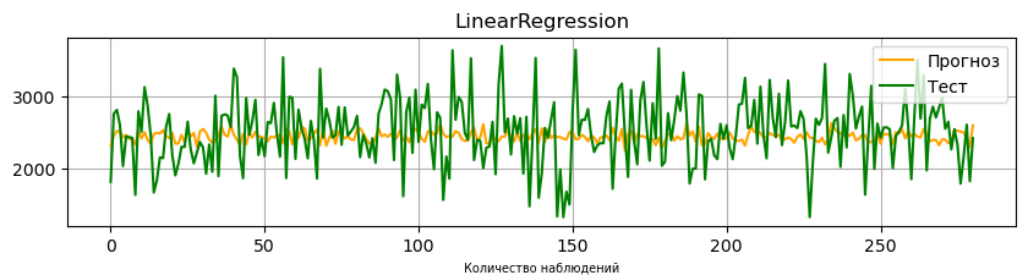
	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.008s	-0.05	363.14	213699.15	462.28	0.15
KNeighborsRegressor	0.016s	0.000s	-0.21	387.41	247988.21	497.98	0.16
SVR	0.016s	0.016s	-0.04	361.46	211321.11	459.70	0.15
DecisionTreeRegressor	0.038s	0.000s	-1.37	554.87	483205.11	695.13	0.23
RandomForestRegressor	1.902s	0.018s	-0.10	374.40	224713.09	474.04	0.15
GradientBoostingRegressor	0.696s	0.000s	-0.13	381.93	229672.12	479.24	0.16
SGDRegressor	0.007s	0.000s	-0.09	369.19	221973.99	471.14	0.15

Результаты кросс-валидации

	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.02	374.93	219981.89	467.74	0.16
KNeighborsRegressor	-0.22	410.77	261977.56	510.81	0.18
SVR	-0.01	372.65	217943.05	465.94	0.16
DecisionTreeRegressor	-1.08	523.09	422603.88	658.96	0.23
RandomForestRegressor	-0.05	383.07	229260.14	477.17	0.17
GradientBoostingRegressor	-0.11	391.14	240134.74	488.43	0.17
SGDRegressor	-0.07	384.83	226419.70	474.80	0.17

CPU times: total: 1min 58s

Wall time: 2min 2s



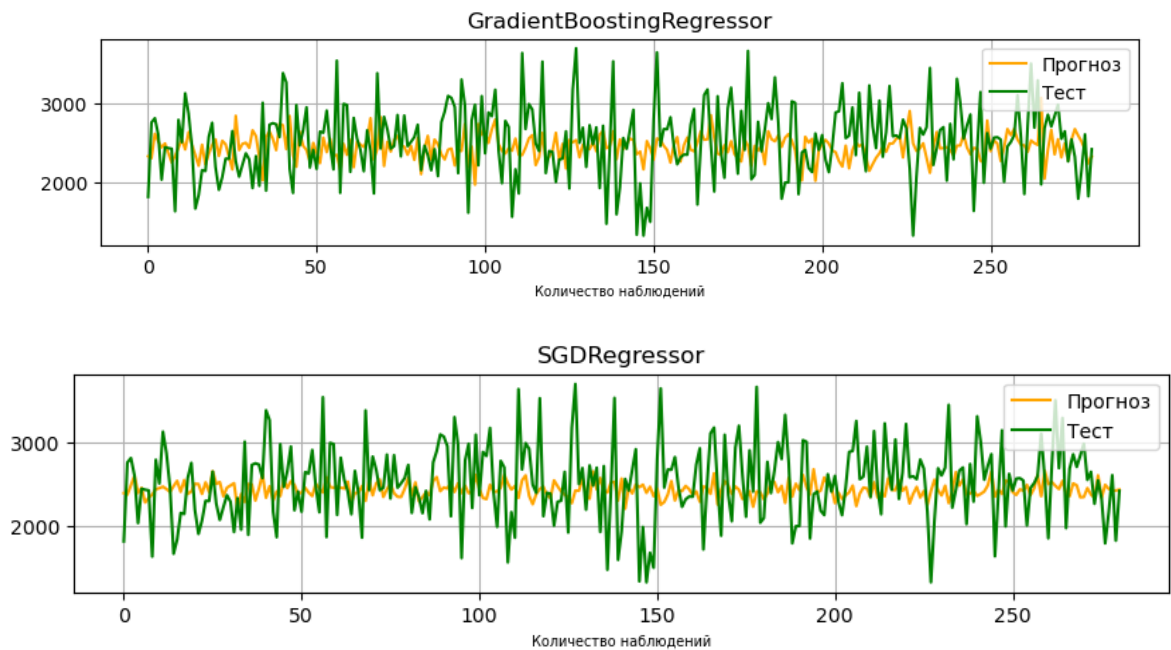


Рисунок 24 – обучение модели с удалением выбросов по методу межквартильных расстояний (IQR)

Поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой.

```
# гиперпараметры
parameters = {
    'LinearRegression()':
        {'fit_intercept': [True, False],
         'n_jobs': [None, 1, 2],
         },
    'KNeighborsRegressor()':
        {'n_neighbors': range(44, 50),
         'weights': ['uniform', 'distance'],
         },
    'SVR()':
        {'kernel': ['linear', 'poly', 'rbf'],
         'C': [0.01, 0.1, 0.9],
         },
    'DecisionTreeRegressor()':
        {'criterion': ['squared_error', 'friedman_mse', 'absolute_error'],
         'splitter': ['best', 'random'],
         'max_depth': [2, 3, None],
         'max_features': [7, 8, 'sqrt', 'log2', None],
         'ccp_alpha': [0.0, 0.001, 0.01, 0.1],
         },
    'RandomForestRegressor()':
        {'n_estimators': [50, 90, 100, 150, 200],
         'max_depth': [None, 2, 5, 10, 12],
         'max_features': [3, 7, 9, 12, 21, 'sqrt', 'log2', None],
         },
    'GradientBoostingRegressor()':
        {'n_estimators': [30, 35, 40],
         'max_depth': [3, 7, 9],
         },
    'SGDRegressor()':
        {'penalty': ['l2', 'l1'],
         'learning_rate': ['constant', 'optimal'],
         },
}
```

Рисунок 25 – гиперпараметры

Гиперпараметрическая оптимизация модели методом поиска по сетке показателя "Прочность при растяжении, МПа"

Размер обучающей выборки: 655

Размер тестовой выборки: 281

```
{'LinearRegression()': [LinearRegression()]}
```

	Best parameters	R2	MAE	MSE	RMSE	MAPE
LinearRegression()	R2:LinearRegression(); MAE:LinearRegression(); MSE:LinearRegression(); RMSE:LinearRegression(); MAPE:LinearRegression()	-0.02	374.93	219981.89	467.74	0.16

CPU times: total: 344 ms

Wall time: 2.48 s

	Best parameters	R2	MAE	MSE	RMSE	MAPE
KNeighborsRegressor()	R2:KNeighborsRegressor(n_neighbors=49); MAE:KNeighborsRegressor(n_neighbors=49); MSE:KNeighborsRegressor(n_neighbors=49); RMSE:KNeighborsRegressor(n_neighbors=49); MAPE:KNeighborsRegressor(n_neighbors=49)	-0.03	375.73	221140.35	469.47	0.16

CPU times: total: 422 ms

Wall time: 933 ms

	Best parameters	R2	MAE	MSE	RMSE	MAPE
SVR()	R2:SVR(C=0.9, kernel='poly'); MAE:SVR(C=0.9, kernel='poly'); MSE:SVR(C=0.9); RMSE:SVR(C=0.9, kernel='poly'); MAPE:SVR(C=0.9, kernel='poly')	-0.01	372.51	217949.40	465.88	0.16

CPU times: total: 531 ms

Wall time: 2.44 s

	Best parameters	R2	MAE	MSE	RMSE	MAPE
DecisionTreeRegressor()	R2:DecisionTreeRegressor(ccp_alpha=0.1, max_depth=2, max_features='sqrt'); MAE:DecisionTreeRegressor(criterion='friedman_mse', max_depth=2, max_features='sqrt'); MSE:DecisionTreeRegressor(criterion='absolute_error', max_depth=2, max_features=7, splitter='random'); RMSE:DecisionTreeRegressor(max_depth=2, splitter='random'); MAPE:DecisionTreeRegressor(ccp_alpha=0.01, criterion='absolute_error', max_depth=3, max_features=7, splitter='random')	-0.00	371.07	214643.98	461.39	0.16

CPU times: total: 3.78 s

Wall time: 45.7 s

	Best parameters	R2	MAE	MSE	RMSE	MAPE
RandomForestRegressor()	R2:RandomForestRegressor(max_depth=2, max_features=3); MAE:RandomForestRegressor(max_depth=2, max_features=3, n_estimators=50); MSE:RandomForestRegressor(max_depth=2, max_features=3); RMSE:RandomForestRegressor(max_depth=2, max_features='sqrt', n_estimators=150); MAPE:RandomForestRegressor(max_depth=5, max_features=7, n_estimators=50)	-0.00	371.66	215797.58	464.00	0.16

CPU times: total: 20.9 s

Wall time: 41min 31s

	Best parameters	R2	MAE	MSE	RMSE	MAPE
GradientBoostingRegressor()	R2:GradientBoostingRegressor(n_estimators=30); MAE:GradientBoostingRegressor(n_estimators=30); MSE:GradientBoostingRegressor(n_estimators=30); RMSE:GradientBoostingRegressor(n_estimators=30); MAPE:GradientBoostingRegressor(n_estimators=30)	-0.04	377.41	224967.40	473.12	0.16

CPU times: total: 1.8 s

Wall time: 56.1 s

	Best parameters	R2	MAE	MSE	RMSE	MAPE
SGDRegressor()	R2:SGDRegressor(learning_rate='constant'); MAE:SGDRegressor(learning_rate='constant', penalty='l1'); MSE:SGDRegressor(learning_rate='constant'); RMSE:SGDRegressor(learning_rate='constant'); MAPE:SGDRegressor(learning_rate='constant', penalty='l1')	-0.04	375.96	225967.73	472.72	0.16

CPU times: total: 266 ms

Wall time: 432 ms

Рисунок 26 – гиперпараметрическая оптимизация модели

```
# Применение Нейронной сети с выбросами iqr
y = dataset_clean_iqr['Прочность при растяжении, МПа']
x = dataset_clean_iqr.drop(columns=['Прочность при растяжении, МПа'])

x_train, x_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.3,
                                                    random_state=42)

normalizer = Normalization(input_shape=[12,], axis=None)
normalizer.adapt(np.array(x_train))
```

Рисунок 27 – применение нейронной сети с удалением выбросов по методу межквартильных расстояний (IQR)

y_test значение 359 1823.256512
 Name: Прочность при растяжении, МПа, dtype: float64
 Предсказательное значение: [[2409.5596]]

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_9	6.185s	0.131s	-0.14	376.02	231706.12	481.36	0.15

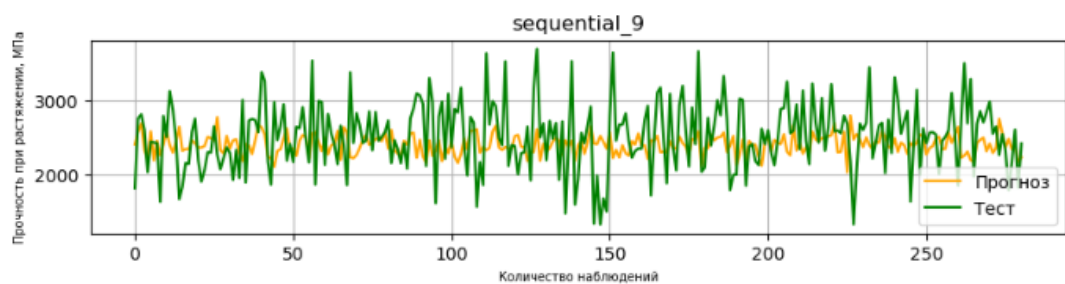
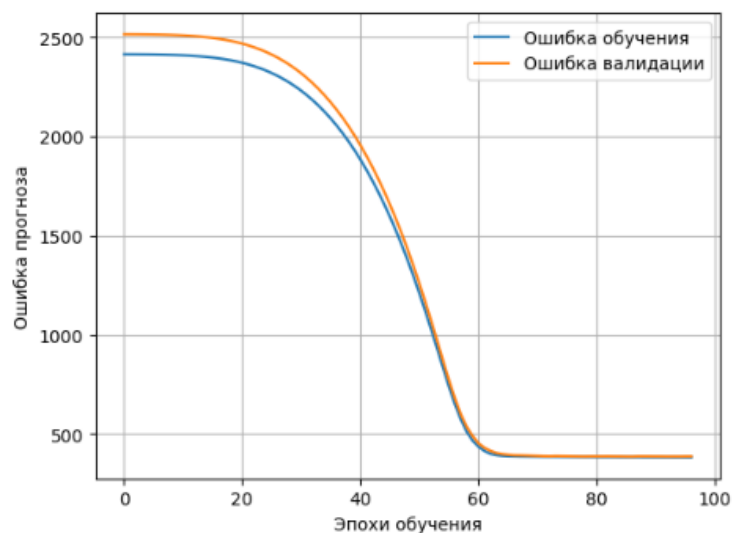


Рисунок 28 – результаты применения нейронной сети (model_3)

y_test значение 359 1823.256512
Name: Прочность при растяжении, МПа, dtype: float64
Предсказательное значение: [[2415.3328]]

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_9	6.185s	0.131s	-0.14	376.02	231706.12	481.36	0.15
sequential_10	6.263s	0.148s	-0.09	370.05	221833.85	470.99	0.15

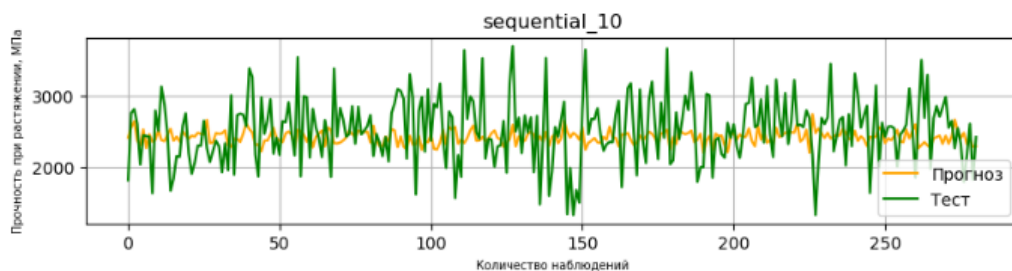
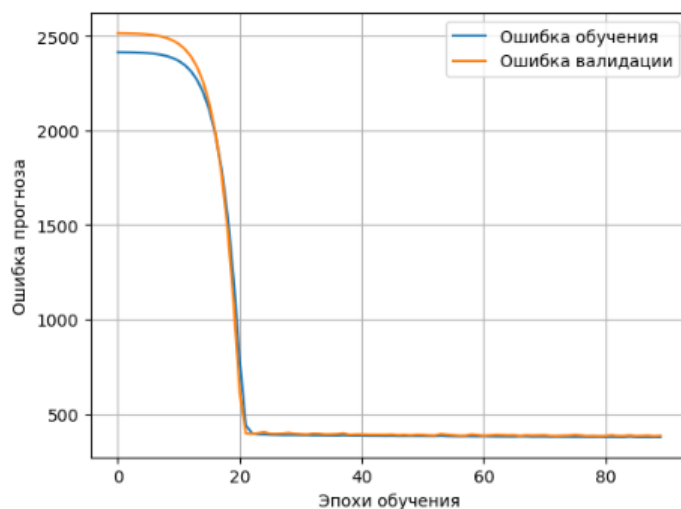


Рисунок 29 – результаты применения нейронной сети (model_4)

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.008s	-0.05	363.14	213699.15	462.28	0.15
KNeighborsRegressor	0.016s	0.000s	-0.21	387.41	247988.21	497.98	0.16
SVR	0.016s	0.016s	-0.04	361.46	211321.11	459.70	0.15
DecisionTreeRegressor	0.038s	0.000s	-1.37	554.87	483205.11	695.13	0.23
RandomForestRegressor	1.902s	0.018s	-0.10	374.40	224713.09	474.04	0.15
GradientBoostingRegressor	0.696s	0.000s	-0.13	381.93	229672.12	479.24	0.16
SGDRegressor	0.007s	0.000s	-0.09	369.19	221973.99	471.14	0.15
sequential_9	6.185s	0.131s	-0.14	376.02	231706.12	481.36	0.15
sequential_10	6.263s	0.148s	-0.09	370.05	221833.85	470.99	0.15

Рисунок 30 – общие результаты обучения моделей для показателя
«Прочность при растяжении»

3.4 Нейронная сеть

Данные для построения нейронной сети были очищены от выбросов по методу межквартильных расстояний (IQR) и разделены на обучающую и тестовую выборки.

```
# удаление выбросов по методу IQR
data_iqr = dataset.copy()
dataset_clean_iqr = outliers_IQR(data_iqr)
dataset_clean_iqr.shape
```

Число обнаруженных выбросов:
Метод межквартильных расстояний (IQR) - 93
Выбросы после очистки - 10
Размер очищенного датасета: (936, 13)

Рисунок 31 – результаты очистки датасета от выбросов

```
#Разделение данных на обучающие и тестовые наборы
y = dataset_clean_iqr['Соотношение матрица-наполнитель']
x = dataset_clean_iqr.drop(columns=['Соотношение матрица-наполнитель'])

X_train, X_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.3,
                                                    random_state=42)

normalizer = Normalization(input_shape=[12,], axis=None)
normalizer.adapt(np.array(X_train))
```

Рисунок 32 – разбивка и нормализация датасета

Пять вариантов архитектуры:

```
model_5 = Sequential([
    normalizer,
    Dense(units=1)
])
model_5.summary()
```

Model: "sequential_4"

Layer (type)	Output Shape	Param #
normalization_6 (Normalization)	(None, 12)	3
dense_22 (Dense)	(None, 1)	13

=====
Total params: 16 (68.00 Byte)
Trainable params: 13 (52.00 Byte)
Non-trainable params: 3 (16.00 Byte)

Рисунок 33– архитектура модели (model_5)

y_test значение 359 2.871562
 Name: Соотношение матрица-наполнитель, dtype: float64
 Предсказательное значение: [[2.8620236]]

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.220s	0.100s	-0.03	0.71	0.78	0.89	0.30

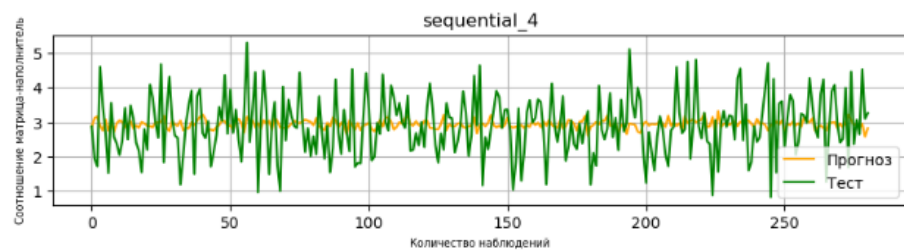
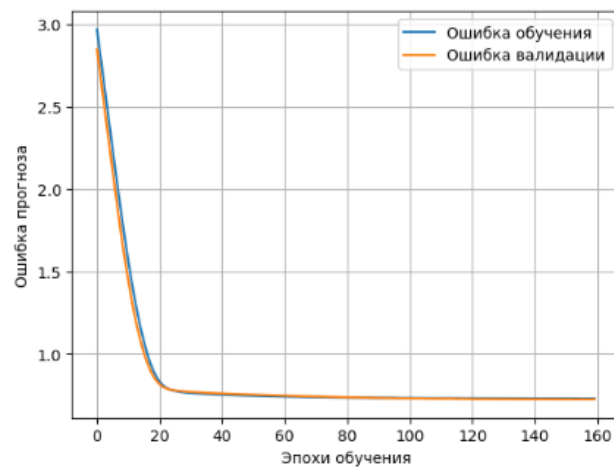


Рисунок 34 – результаты применения нейронной сети (model_5)

```
model_6 = Sequential([
    normalizer,
    Dense(64, activation='relu'),
    Dense(64, activation='relu'),
    Dense(1)
])
model_6.summary()
```

Model: "sequential_5"

Layer (type)	Output Shape	Param #
normalization_6 (Normalization)	(None, 12)	3
dense_23 (Dense)	(None, 64)	832
dense_24 (Dense)	(None, 64)	4160
dense_25 (Dense)	(None, 1)	65

=====
 Total params: 5060 (19.77 KB)
 Trainable params: 5057 (19.75 KB)
 Non-trainable params: 3 (16.00 Byte)

Рисунок 35– архитектура модели (model_6)

y_test значение 359 2.871562
 Name: Соотношение матрица-наполнитель, dtype: float64
 Предсказательное значение:[[2.8480115]]

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.226s	0.100s	-0.03	0.71	0.78	0.89	0.30
sequential_5	1.988s	0.132s	-0.02	0.70	0.78	0.88	0.30

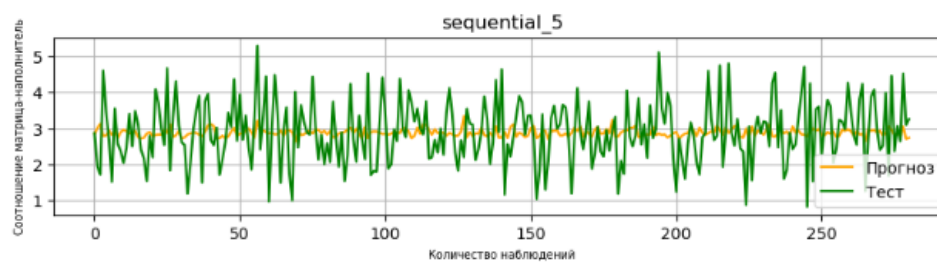
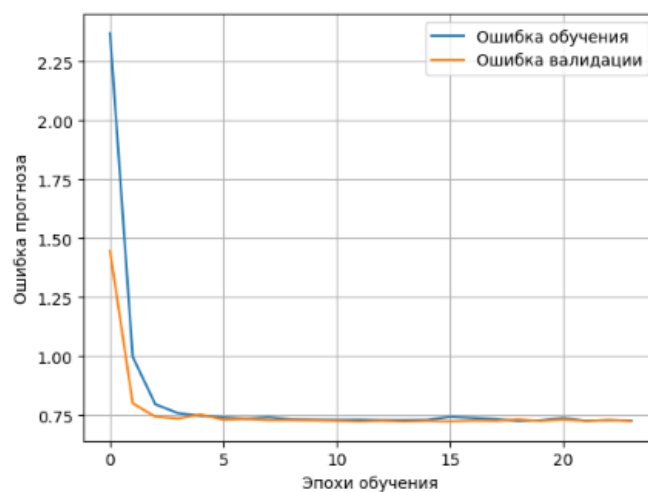


Рисунок 36 – результаты применения нейронной сети (model_6)

```
model_7 = Sequential([
    normalizer,
    Dense(8, activation='relu'),
    Dense(8, activation='relu'),
    Dense(1)
])
model_7.summary()
```

Model: "sequential_6"

Layer (type)	Output Shape	Param #
normalization_6 (Normaliza tion)	(None, 12)	3
dense_26 (Dense)	(None, 8)	104
dense_27 (Dense)	(None, 8)	72
dense_28 (Dense)	(None, 1)	9
Total params: 188 (756.00 Byte)		
Trainable params: 185 (740.00 Byte)		
Non-trainable params: 3 (16.00 Byte)		

Рисунок 37– архитектура модели (model_7)

y_test значение 359 2.871562
 Name: Соотношение матрица-наполнитель, dtype: float64
 Предсказательное значение:[[2.7993858]]

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.228s	0.100s	-0.03	0.71	0.78	0.89	0.30
sequential_5	1.988s	0.132s	-0.02	0.70	0.78	0.88	0.30
sequential_6	1.792s	0.216s	-0.01	0.71	0.77	0.88	0.30

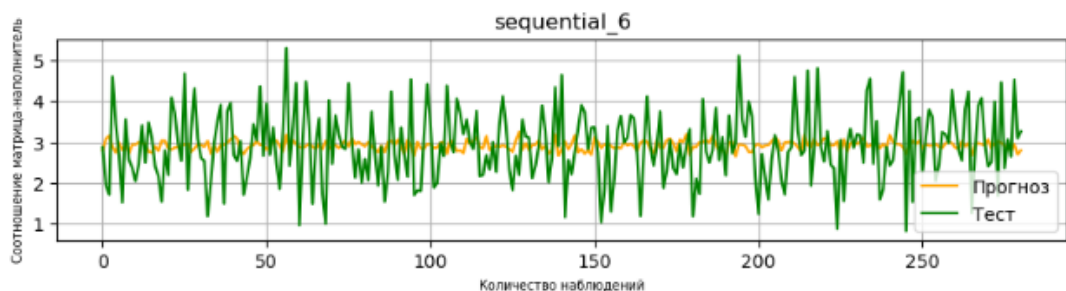
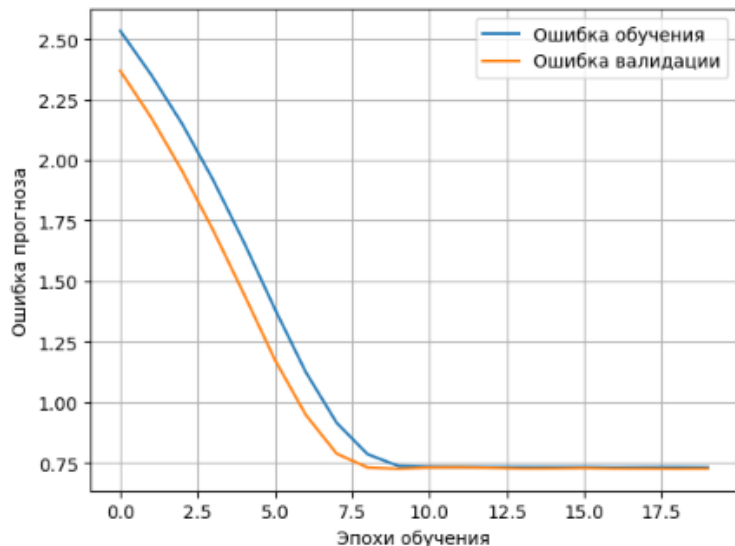


Рисунок 38 – результаты применения нейронной сети (model_7)

```
model_8 = Sequential([
    normalizer,
    Dense(64, activation='relu'),
    Dense(8, activation='relu'),
    Dense(8, activation='relu'),
    Dense(1)
])
model_8.summary()
```

Model: "sequential_7"

Layer (type)	Output Shape	Param #
normalization_6 (Normaliza tion)	(None, 12)	3
dense_29 (Dense)	(None, 64)	832
dense_30 (Dense)	(None, 8)	520
dense_31 (Dense)	(None, 8)	72
dense_32 (Dense)	(None, 1)	9

Total params: 1436 (5.61 KB)
Trainable params: 1433 (5.60 KB)
Non-trainable params: 3 (16.00 Byte)

Рисунок 39– архитектура модели (model_8)

y_test значение 359 2.871562
Name: Соотношение матрица-наполнитель, dtype: float64
Предсказательное значение:[[2.7823238]]

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.226s	0.100s	-0.03	0.71	0.78	0.89	0.30
sequential_5	1.988s	0.132s	-0.02	0.70	0.78	0.88	0.30
sequential_6	1.792s	0.216s	-0.01	0.71	0.77	0.88	0.30
sequential_7	6.202s	0.122s	-0.02	0.71	0.77	0.88	0.30

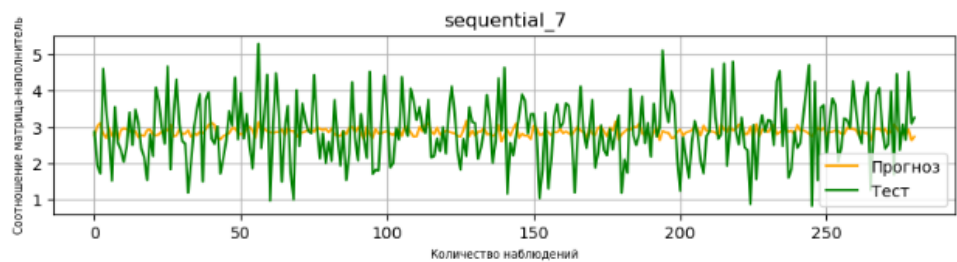
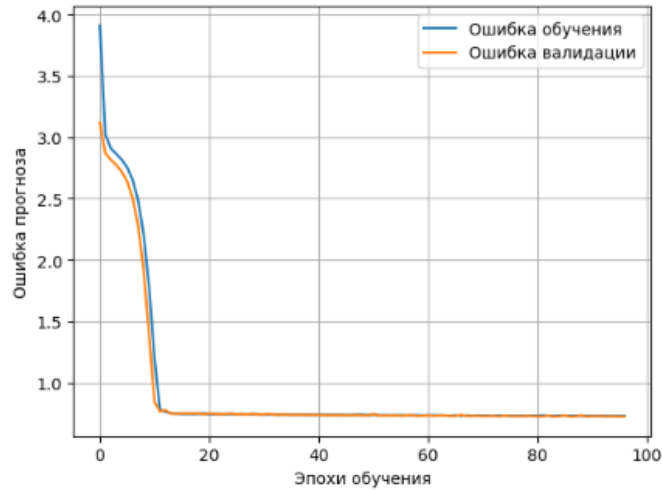


Рисунок 40 – результаты применения нейронной сети (model_8)

```
model_9 = Sequential([
    normalizer,
    Dense(8, activation='linear'),
    Dense(8, activation='linear'),
    Dense(1)
])
model_9.summary()
```

```
Model: "sequential_8"
Layer (type)                Output Shape         Param #
=====
normalization_6 (Normaliza  (None, 12)           3
tion)

dense_33 (Dense)             (None, 8)            104
dense_34 (Dense)             (None, 8)             72
dense_35 (Dense)             (None, 1)             9
=====
Total params: 188 (756.00 Byte)
Trainable params: 185 (740.00 Byte)
Non-trainable params: 3 (16.00 Byte)
```

Рисунок 41 – архитектура модели (model_9)


```

y_test значение 359    2.871562
Name: Соотношение матрица-наполнитель, dtype: float64
Предсказательное значение:[[2.8346715]]

```

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.228s	0.100s	-0.03	0.71	0.78	0.89	0.30
sequential_5	1.988s	0.132s	-0.02	0.70	0.78	0.88	0.30
sequential_6	1.792s	0.218s	-0.01	0.71	0.77	0.88	0.30
sequential_7	6.202s	0.122s	-0.02	0.71	0.77	0.88	0.30
sequential_8	3.294s	0.118s	-0.02	0.71	0.78	0.88	0.30

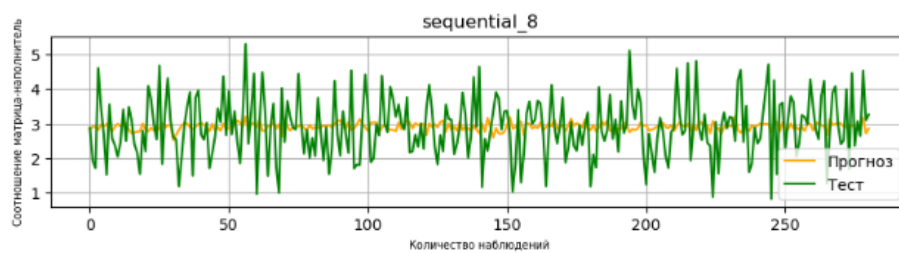
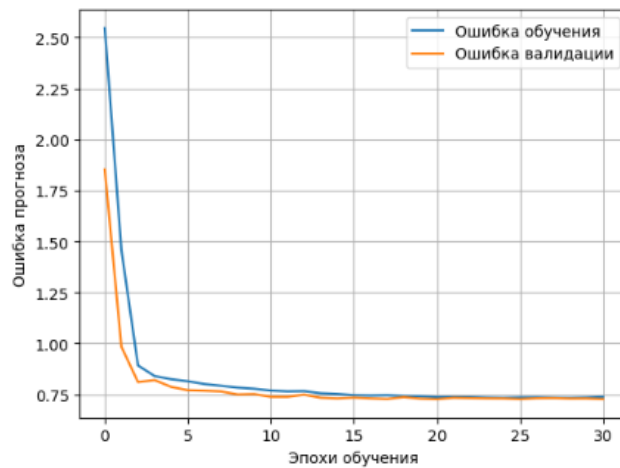


Рисунок 42 – результаты применения нейронной сети (model_9)

```

import pickle
pickle.dump(model_7, open(r'D:\PROJECTS\VKR\model\model.pkl', 'wb'))

```

Рисунок 43 – сохранение модели для дальнейшего использования в приложении

3.5 Разработка приложения

Приложение было разработано с помощью Flask.

← → ↻ ⓘ 127.0.0.1:5000

**Рекомендация параметра
«Соотношение матрица-наполнитель»**

Введите значения свойств композитов

Плотность, кг/м3:

модуль упругости, ГПа:

Количество отвердителя, м.-%:

Содержание эпоксидных групп, %_2:

Температура вспышки, С_2:

Поверхностная плотность, г/м2:

Модуль упругости при растяжении, ГПа:

Прочность при растяжении, МПа:

Потребление смолы, г/м2:

Угол нашивки, град:

Шаг нашивки:

Плотность нашивки:

Рисунок 44 – интерфейс приложения

В приложении необходимы следующие шаги:

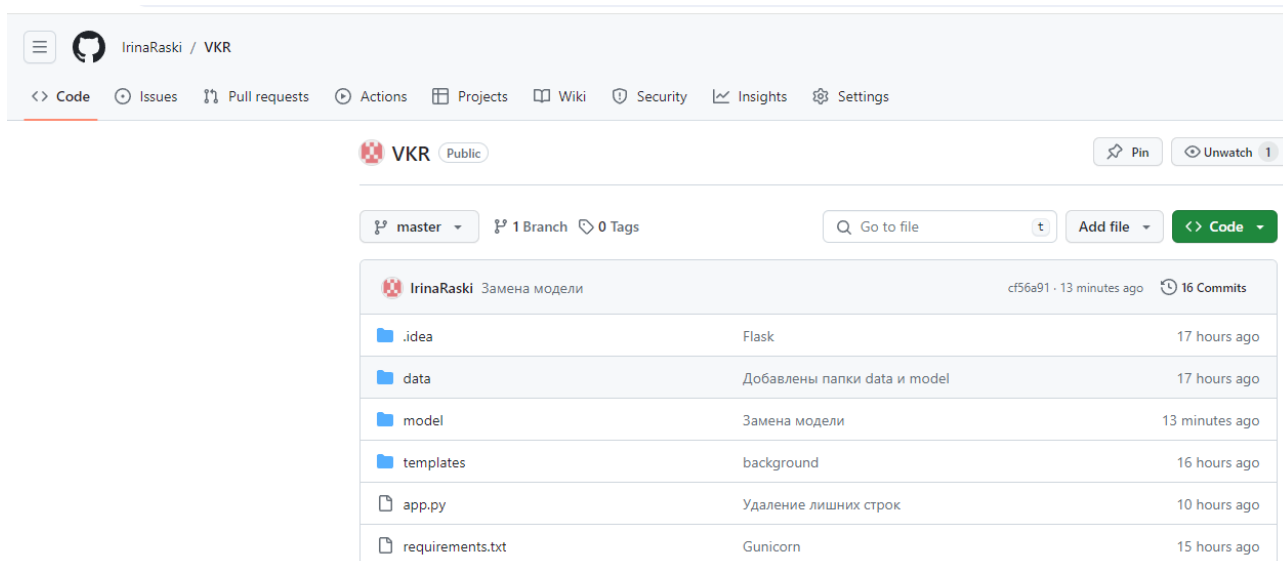
- ✓ ввести значения следующих показателей:
 - Плотность, кг/м3,
 - модуль упругости, ГПа,
 - Количество отвердителя, м.-%,
 - Содержание эпоксидных групп,%_2,
 - Температура вспышки, С_2,
 - Модуль упругости при растяжении, ГПа,
 - Прочность при растяжении, МПа,
 - Угол нашивки, град,
 - Шаг нашивки,
 - Плотность нашивки,
- ✓ нажать кнопку «Отправить».

Далее на основании сохраненной модели, будет выводиться рекомендуемое значение соотношение матрица – наполнитель.

3.6 Создание удаленного репозитория

Репозиторий создан на github.com по адресу:

<https://github.com/IrinaRaski/VKR>



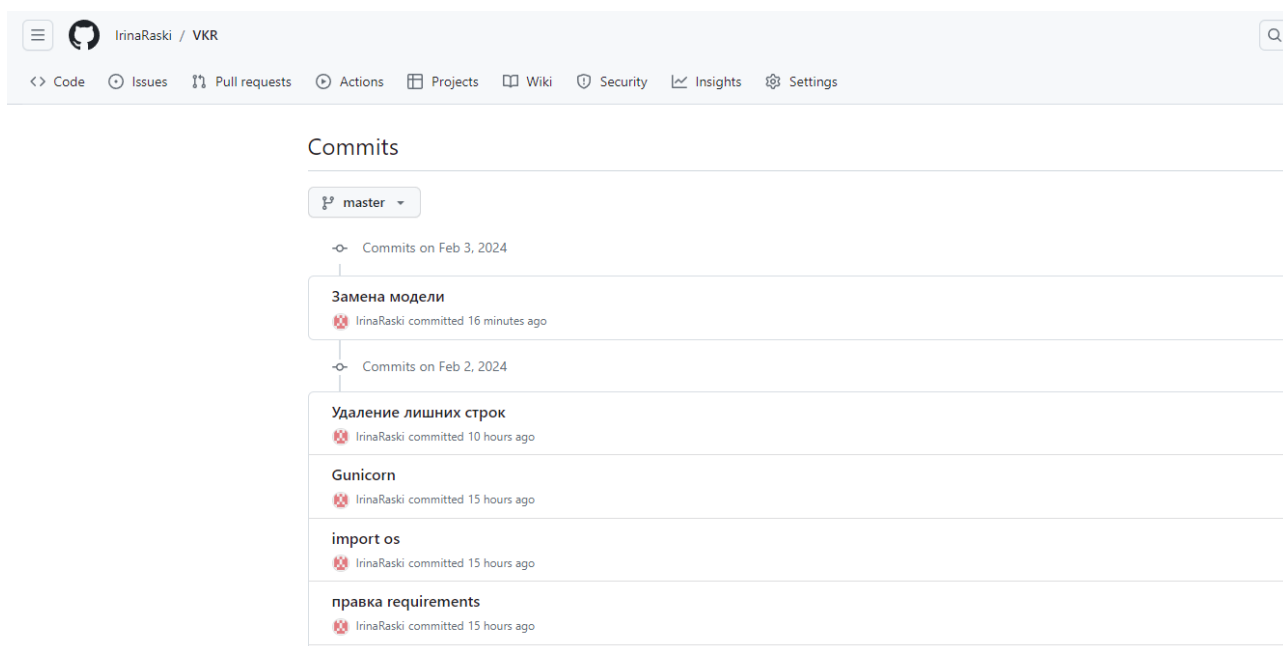
Repository: IrinaRaski / VKR (Public)

Branch: master | 1 Branch | 0 Tags

Search: Go to file | Add file | Code

Commit Hash	Commit Message	Time Ago
cf56a91	Замена модели	13 minutes ago
	Flask	17 hours ago
	Добавлены папки data и model	17 hours ago
	Замена модели	13 minutes ago
	background	16 hours ago
	Удаление лишних строк	10 hours ago
	Gunicorn	15 hours ago

Рисунок 45 – профиль на github.com



Commits

Branch: master

Commits on Feb 3, 2024

- Замена модели**
IrinaRaski committed 16 minutes ago

Commits on Feb 2, 2024

- Удаление лишних строк**
IrinaRaski committed 10 hours ago
- Gunicorn**
IrinaRaski committed 15 hours ago
- import os**
IrinaRaski committed 15 hours ago
- правка requirements**
IrinaRaski committed 15 hours ago

Рисунок 46 – commit приложения

4 Заключение

В ходе работы были разработаны модели машинного обучения и нейронные сети для прогноза конечных свойств получаемых композиционных материалов.

Использованные при разработке моделей подходы не позволили получить достоверные прогнозы. Примененные модели регрессии не показали высокой эффективности в прогнозировании свойств композитов. Лучшие метрики для модуля упругости при растяжении, ГПа – линейная регрессия, для прочности при растяжении, МПа – метод опорных векторов.

5 Библиографический список

1. Современные композиционные строительные материалы: учеб. пособие / И.Ю. Шитова, Е.Н. Самошина, С.Н. Кислицына, С.А. Болтышев. – Пенза: ПГУАС, 2015. – 136 с.
2. Документация по библиотеке matplotlib: – Режим доступа: <https://matplotlib.org/stable/users/index.html>.
3. Документация по библиотеке seaborn: – Режим доступа: <https://seaborn.pydata.org/tutorial.html>.
4. Документация по библиотеке sklearn: – Режим доступа: https://scikit-learn.org/stable/user_guide.html.
5. Документация по библиотеке keras: – Режим доступа: <https://keras.io/api/>.
6. Документация по библиотеке numpy: – Режим доступа: <https://numpy.org/doc/1.22/user/index.html>.
7. Документация по библиотеке pandas: – Режим доступа: https://pandas.pydata.org/docs/user_guide/index.html.