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 2022 4.0»

на тему

«Способы прогнозирования конечных свойств новых композиционных материалов
и разработка моделей для выполнения прогнозов»

Слушатель: Куринной Александр Васильевич

Симферополь, 2023

Содержание

Введение. 3

1. Аналитическая часть 5

1.1 Постановка задачи. 5

1.2. Описание используемых методов 9

1.3 Разведочный анализ данных. 21

2. Практическая часть. 31

2.1 Предобработка данных. 31

2.2 Разработка и обучение модели. 42

2.3 Тестирование модели. 50

2.4 Нейронная сеть для рекомендации соотношения «матрица – наполнитель». 52

2.5 Разработка приложения для прогнозирования соотношения «матрица–наполнитель». 56

2.6 Создание удаленного репозитория и загрузка результатов работы. 60

Заключение. 61

Список литературы. 63

Введение.

Композиционный материал (композит) представляет собой неоднородный сплошной материал, состоящий из двух или более компонентов, среди которых можно выделить армирующие элементы (наполнители), обеспечивающие необходимые механические характеристики материала, и матрицу (или связующее), обеспечивающую совместную работу армирующих элементов. Матрица может быть металлическая, керамическая, углеродная, полимерная и т.д. Наполнитель может состоять из частиц, волокон, тканей или листов.

По сравнению с традиционными материалами, композиты обладают рядом преимуществ, в числе которых малый вес, высокая жёсткость, незначительное расширение при нагреве, коррозионная стойкость. Важнейшая особенность композитов в том, что с помощью различных комбинаций материалов и добавок можно улучшить такие критические свойства как жёсткость, эластичность и прочность.

На сегодняшний день основными тенденциями, формирующими ёмкость и структуру рынка композитных материалов, являются:

- рост спроса со стороны инновационных сфер экономики – строительного, авиа-, автомобилестроения, нефтегазовой сферы, ветроэнергетики и др.;
- улучшение технологических и экологических характеристик известных видов связующих и волокнистых наполнителей;
- расширение использования компьютерного моделирования и проектирования деталей из композиционных материалов.

Целью выпускной квалификационной работы является изучение способов прогнозирования конечных свойств новых композиционных материалов и разработка моделей для выполнения прогнозов.

Для достижения данной цели необходимо решение ряда задач:

- разработка алгоритма машинного обучения для прогноза значений модуля упругости при растяжении и прочности при растяжении;
- создание нейронной сеть для рекомендации соотношения «матрица-наполнитель».

Объект выпускной квалификационной работы – производственные данные Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

Предмет выпускной квалификационной работы – прогнозирование конечных свойств новых материалов (композиционных материалов).

1. Аналитическая часть.

1.1. Постановка задачи.

В определениях композитов особо подчеркивают, что данный материал обладает рядом свойств, которых не имеют его компоненты, взятые по отдельности. Таким образом, путем подбора состава и свойств наполнителя и матрицы (связующего), их соотношения, ориентации наполнителя можно получить материалы с требуемым сочетанием эксплуатационных и технологических свойств.

В связи с широким применением композиционных материалов во многих сферах экономической деятельности появилась необходимость разработки автоматизированной информационной системы прогнозирования свойств исследуемых материалов на основе регрессионного анализа, которая позволит снизить трудоемкость экспериментальных исследований, повысить качество прогнозирования физико-механических и технологических свойств материалов, снизить себестоимость изготавливаемых изделий.

В настоящее время при производстве композитов свойства конечных изделий определяются по контрольным образцам, полученным из соответствующей серии партии деталей. Практически не существует эффективных методик, позволяющих прогнозировать свойства конечных изделий на основе информации о компонентах. Конечная концентрация компонентов в композитах зависит от исходных компонентов и параметров технологического процесса их изготовления.

Анализ композиционных материалов, методов обработки экспериментальных данных и программных средств, применяемых для исследования свойств материалов, показал возможность использования корреляционно-регрессионного анализа для формирования прогнозных моделей с помощью прикладных программ.

Для получения прогнозных моделей свойств композитных материалов, а именно модуля упругости при растяжении, прочности при растяжении и соотношения «матрица-наполнитель», необходимо выполнить следующие шаги:

- провести разведочный анализ предложенных данных, построить гистограммы распределения каждой из переменной, диаграммы ящика с усами, попарные графики рассеяния точек;
- провести предобработку данных (удаление шумов, нормализация и стандартизация данных и т.д.);
- обучить несколько моделей для прогноза модуля упругости при растяжении и прочности при растяжении;
- написать нейронную сеть, которая будет рекомендовать соотношение «матрица-наполнитель»;
- разработать приложение, которое будет выдавать прогноз соотношения «матрица-наполнитель»;
- провести оценку точности модели на тренировочном и тестовом датасете.

Для выполнения исследования использованы производственные данные Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

Информация представлена в виде двух файлов формата excel: X_bp.xlsx (характеристики базальтопластика) и X_nup.xlsx (характеристики нашивки из углепластика). Датасет X_bp. Датасет из файла X_bp.xlsx содержит 1023 строки и 11 столбцов, из файла X_nup.xlsx – 1040 строк и 4 столбца (рисунки 1-2).

```
[ ] # Загрузим датасет из файла "X_bp.xlsx" (характеристики базальтопластика) и выведем 5 первых позиций датасета
    dataset_bp = pd.read_excel('/content/X_bp.xlsx')
    dataset_bp.enable_dataframe_formatter()
    # dataset_bp.head()
```

	Unnamed: 0	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
0	0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0
1	1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0
2	2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0
3	3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0
4	4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0

```
[ ] # Посмотрим размерность первого датасета dataset_bp
    dataset_bp.shape
```

```
(1023, 11)
```

Рисунок 1. Датасет из файла X_bp.xlsx.

```
[ ] # Загрузим датасет из файла "X_nup.xlsx" (характеристики нашивки из углепластика) и выведем 5 первых позиций датасета
    dataset_nup = pd.read_excel('/content/X_nup.xlsx')
    dataset_nup.head()
```

	Unnamed: 0	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	0	0	4.0	57.0
1	1	0	4.0	60.0
2	2	0	4.0	70.0
3	3	0	5.0	47.0
4	4	0	5.0	57.0

```
[ ] # Посмотрим размерность второго датасета dataset_nup
    dataset_nup.shape
```

```
(1040, 4)
```

Рисунок 2. Датасет из файла X_nup.xlsx.

Для дальнейшей работы указанные выше файлы были объединены в один. Полученный датасет содержит 1023 строки и 13 столбцов (рисунок 3).

Это означает, что часть данных (а именно 17 строк из датасета dataset_nup) была удалена из таблицы и исключена из дальнейшего исследования.

Датасеты имеют разное количество строк: 1023 строки у dataset_bp, 1040 строк у dataset_nup

```
[ ]
# Объединеним датесеты в один
# Объединение выполним с помощью функции merge() по индексу с типом объединения INNER
dataset = pd.merge(dataset_bp, dataset_nup,
                   left_index=True,
                   right_index=True,
                   how = "inner")

# Выведем 5 первых позиций нового датасета
dataset.head()
```

	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	Шаг нашивки	Плотность нашивки
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0	0	4.0	57.0
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0	0	4.0	60.0
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0	0	4.0	70.0
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0	0	5.0	47.0
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0	0	5.0	57.0

```
[ ]
# Посмотрим размерность объединенного датасета
dataset.shape
```

```
(1023, 13)
```

Объединенный датасет содержит 1023 строки и 13 столбцов. Это означает, что часть данных (а именно 17 строк из датасета dataset_nup) была удалена из таблицы и исключена из дальнейшего исследования.

Рисунок 3. Объединенный датасет для работы.

Детальный анализ датасета будет приведен в следующих разделах.

1.2. Описание используемых методов.

При решении конкретной задачи машинного обучения нельзя сказать заранее, какой вид модели будет наиболее эффективен. Проблему выбора модели чаще всего решают перебором – нужно попробовать разные типы моделей и выбрать те, которые показывают наибольшую эффективность. Для задачи прогнозирования модуля упругости при растяжении и прочности при растяжении рассмотрим следующие модели:

1. Метод К-ближайших соседей;
2. Метод опорных векторов;
3. Линейная регрессия;
4. Дерево решений;
5. AdaBoost;
6. Градиентный бустинг;
7. XGBoost;
8. Случайный лес;
9. Стохастический градиентный спуск;
10. Метод регрессии «Lasso».

1. Метод К-ближайших соседей (k-Nearest Neighbors, или kNN). Суть метода достаточно проста: посмотри на соседей вокруг, какие из них преобладают, таковым ты и являешься. В случае использования метода для классификации объект присваивается тому классу, который является наиболее распространённым среди k соседей данного элемента, классы которых уже известны. В случае использования метода для регрессии, объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны. Регрессия на основе соседей может использоваться в случаях, когда метки данных являются непрерывными, а не дискретными переменными.

Преимущества:

- Легкая и простая модель машинного обучения.

- Легко добавить больше данных во множество данных.
- Модель принимает только 2 параметра: К и метрика расстояния (обычно это Евклидово расстояние).

Недостатки:

- К следует выбирать мудро.
- Большие вычислительные затраты во время выполнения, если размер выборки велик.
- Работает не так хорошо с категорическими параметрами.

2. Метод опорных векторов (Support Vector Machines, или SVM) – это один из наиболее широко используемых алгоритмов машинного обучения, применяемый для решения задач классификации, регрессии и обнаружения выбросов. Алгоритм для решения задач классификации строит гиперплоскость в n-мерном пространстве для разделения объектов двух или более классов. Гиперплоскость выбирается таким образом, чтобы максимизировать расстояние между гиперплоскостью и ближайшими объектами разных классов (зазор). Объекты, которые расположены ближе всего к гиперплоскости, называются опорными векторами.

Метод классификации опорных векторов может быть расширен для решения задач регрессии. Этот метод называется регрессией опорных векторов (Support Vector Regression, или SVR). В SVR идентифицируется гиперплоскость с максимальным запасом, так что максимальное количество точек данных находится в пределах этого поля.

Преимущества:

- Является одним из наиболее точных алгоритмов машинного обучения, которые могут обучаться на больших наборах данных.
- Хорошо работает с данными, которые имеют большое количество признаков.
- Работает с небольшими выборками данных.

- Малое количество гиперпараметров: имеет только несколько гиперпараметров, что делает его относительно простым для настройки.

Недостатки:

- Чувствительность к шуму. Шум в данных может привести к тому, что SVM строит границу принятия решений, которая не обобщается хорошо на новые данные.
- Вычислительная сложность для обучения на больших наборах данных.
- Выбор правильной функции ядра может быть сложной задачей. Некоторые ядра работают лучше на определенных типах данных, и выбор неправильного ядра может привести к плохим результатам.

3. Линейная регрессия (Linear regression) – это метод машинного обучения с учителем, который используется для предсказания непрерывной целевой переменной от одного или нескольких независимых признаков. В основе метода лежит предположение предполагает о том, что существует линейная связь между признаками и целевой переменной. Эта связь моделируется с помощью линейной функции. Модель линейной регрессии пытается найти лучшую прямую, которая может описывать зависимость между независимыми признаками и зависимой переменной. Это делается с помощью поиска оптимальных коэффициентов, которые могут быть использованы для описания линейной функции. Эта модель может быть использована как для предсказания, так и для анализа влияния признаков на целевую переменную.

Преимущества линейной регрессии:

- Простота и удобство в использовании.
- Эффективность при линейных зависимостях.
- Интерпретируемость: в линейной регрессии каждый коэффициент регрессии может быть использован для определения влияния каждой независимой переменной на зависимую переменную.

Недостатки линейной регрессии:

- Ограниченная эффективность при нелинейных зависимостях: если между независимыми и зависимыми переменными существует нелинейная зависимость, линейная регрессия может давать неточные предсказания.
- Необходимость проведения предварительной подготовки данных: линейная регрессия чувствительна к выбросам и мультиколлинеарности, так что необходимо выполнить предварительную подготовку данных.

4. Дерево принятия решений (Decision Tree) – это непараметрический контролируемый метод обучения, используемый для классификации и регрессии. Цель состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной, изучая простые правила принятия решений, выведенные из характеристик данных. Дерево можно рассматривать как кусочно-постоянное приближение.

Некоторые преимущества деревьев решений:

- Просто понять и интерпретировать. Деревья можно визуализировать.
- Требуется небольшая подготовка данных.
- Стоимость использования дерева является логарифмической по количеству точек данных, используемых для обучения дерева.
- Может обрабатывать как числовые, так и категориальные данные.
- Способен обрабатывать проблемы с несколькими выходами.
- Возможна проверка модели с помощью статистических тестов. Это позволяет учитывать надежность модели.

К недостаткам деревьев решений можно отнести:

- Обучающиеся дереву решений могут создавать слишком сложные деревья, которые плохо обобщают данные. Это называется переобучением.
- Деревья решений могут быть нестабильными, поскольку небольшие изменения в данных могут привести к созданию совершенно другого дерева.

- Предсказания деревьев решений не являются ни гладкими, ни непрерывными, а являются кусочно-постоянными приближениями. Следовательно, они не годятся для экстраполяции.
- Ученики дерева решений создают предвзятые деревья, если некоторые классы доминируют. Поэтому рекомендуется сбалансировать набор данных перед подгонкой к дереву решений.

Одно решающее дерево имеет достаточно грубые границы между листьями, и при этом либо существенно недообучается (при малом количестве листьев), либо сильно переобучается (при большом количестве листьев). Эту проблему можно исправить, обучая сразу много разных решающих деревьев.

В целом ансамблированием называется комбинация нескольких моделей машинного обучения в одну модель. Ансамблирование решающих деревьев как правило осуществляется «одноуровнево», то есть все деревья работают параллельно и независимо выдают ответ, а затем их предсказания складываются или усредняются. Процесс обучения при этом может выполняться параллельно (бэггинг) или последовательно (бустинг).

Наиболее распространенные алгоритмы ансамблирования решающих деревьев: Random forest, Adaboost, Gradient Boosting, XGBoost, LightGBM, CatBoost.

Бустинг (Boosting) – это способ построения ансамбля, в котором обучается много копий более слабой модели («weak learner»), то есть такой модели, которая не может достичь высокой точности на обучающем датасете, переобучившись на нем. Как правило такой моделью является решающее дерево небольшой глубины. На каждом шаге новый weak learner концентрируется на исправлении ошибок, допущенных предыдущими weak learner'ами. В итоге предсказания всех weak learner'ов суммируются с определенными весами. Бустинг чем-то похож на бэггинг, но в бэггинге модели обучаются совершенно независимо и параллельно, а в бустинге последовательно, с оглядкой на предыдущие.

5. AdaBoost (AdaBoostRegressor) – алгоритм машинного обучения, в котором каждый следующий weak learner фокусировал внимание на тех примерах, на которых предыдущие weak learner'ы дали неверные ответы. При этом он не знал, какие именно ответы даны предыдущими weak learner'ами - было лишь известно, что ответы неверны или неточны. Задачей нового weak learner'a было дать верные ответы преимущественно на этих примерах. Заметим, что при этом не используется никакого валидационного датасета. Используется только обучающий датасет, на нем же оценивается точность предыдущих weak learner'ов. Это означает, что если очередной weak learner после обучения дал верные ответы на все примеры, то бустинг продолжить будет невозможно. Например, если в качестве weak learner'a мы используем решающее дерево неограниченной глубины, то так и произойдет. Нужно использовать решающие деревья небольшой глубины: weak learner должен быть действительно "слабым", не переобучаясь слишком сильно.

Преимущества алгоритма AdaBoost:

- Можно легко, быстро и просто запрограммировать.
- Достаточно гибкий, чтобы комбинировать его с любым алгоритмом машинного обучения без настройки параметров.
- Расширяем до задач обучения сложнее, чем двоичная классификация, и достаточно универсален, поскольку его можно использовать с числовыми или текстовыми данными.

Недостатки:

- Этот алгоритм доказывается эмпирически и очень уязвим к равномерно распределенному шуму.
- Слабые классификаторы в случае, если они слишком слабые, могут привести к плохим результатам и переобучению.

6. Градиентный бустинг (GradientBoostingRegressor) – это продвинутый алгоритм машинного обучения для решения задач классификации и регрессии. Он строит предсказание в виде ансамбля слабых предсказывающих моделей,

которыми в основном являются деревья решений. Из нескольких слабых моделей в итоге собирается одна, но уже эффективная. В градиентном бустинге целевыми данными для следующего weak learner'a является градиент (со знаком минус) функции потерь по предсказаниям предыдущих алгоритмов. Таким образом следующий weak learner корректирует предсказания предыдущих. Общая идея алгоритма – последовательное применение предиктора (предсказателя) таким образом, что каждая последующая модель сводит ошибку предыдущей к минимуму.

Преимущества:

- Алгоритм работает с любыми функциями потерь.
- Предсказания в среднем лучше, чем у других алгоритмов.
- Самостоятельно справляется с пропущенными данными.

Недостатки:

- Алгоритм крайне чувствителен к выбросам и при их наличии будет тратить огромное количество ресурсов на эти моменты.
- Модель будет склонна к переобучению при слишком большом количестве деревьев. Данная проблема присутствует в любом алгоритме, связанном с деревьями.
- Вычисления могут занять много времени.

7. XGBoost (XGBRegressor) – является вычислительно эффективной реализацией градиентного бустинга над решающими деревьями. Помимо оптимизированного программного кода, авторы предлагают различные улучшения алгоритма. Основной ценностью библиотеки XGBoost является эффективная программная реализация. За счет разных оптимизаций, таких как эффективная работа с пропущенными значениями, поиск порога только среди перцентилей, оптимизация работа с кэшем и распределенное обучение, достигается выигрыш в десятки или даже сотни раз по сравнению с наивной реализацией.

8. Случайный лес (Random forest) – это метод использующий ансамбль деревьев решений, созданных на случайно разделенном датасете. Набор таких деревьев-классификаторов образует лес. Каждое отдельное дерево решений генерируется с использованием метрик отбора показателей, таких как критерий прироста информации, отношение прироста и индекс Джини для каждого признака.

Любое такое дерево создается на основе независимой случайной выборки. В задаче классификации каждое дерево голосует, и в качестве окончательного результата выбирается самый популярный класс. В случае регрессии конечным результатом считается среднее значение всех выходных данных ансамбля. Метод случайного леса является более простым и эффективным по сравнению с другими алгоритмами нелинейной классификации.

Преимущества:

- Имеет высокую точность предсказания.
- Не требует тщательной настройки параметров.
- Практически не чувствителен к выбросам в данных из-за случайного семплирования (random sample).
- Не чувствителен к масштабированию и к другим монотонным преобразованиям значений признаков.
- Редко переобучается. На практике добавление деревьев только улучшает композицию.
- В случае наличия проблемы переобучения, она преодолевается путем усреднения или объединения результатов различных деревьев решений.
- Способен эффективно обрабатывать данные с большим числом признаков и классов.
- Хорошо работает с пропущенными данными – сохраняет хорошую точность даже при их наличии.
- Одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки.

- Высокая параллелизуемость и масштабируемость.

Недостатки:

- Для реализации алгоритма случайного дерева требуется значительный объем вычислительных ресурсов.
- Большой размер моделей.
- Построение случайного леса отнимает больше времени, чем деревья решений или линейные алгоритмы.
- Алгоритм склонен к переобучению на зашумленных данных.
- Нет формальных выводов, таких как p-values, которые используются для оценки важности переменных.
- В отличие от более простых алгоритмов, результаты случайного леса сложнее интерпретировать.
- Когда в выборке очень много разреженных признаков, таких как тексты или наборы слов (bag of words), алгоритм работает хуже, чем линейные методы.
- В отличие от линейной регрессии, Random Forest не обладает возможностью экстраполяции. Это можно считать и плюсом, так как в случае выбросов не будет экстремальных значений.
- Если данные содержат группы признаков с корреляцией, которые имеют схожую значимость для меток, то предпочтение отдается небольшим группам перед большими, что ведет к недообучению.
- Процесс прогнозирования с использованием случайных лесов очень трудоемкий по сравнению с другими алгоритмами.

9. Стохастический градиентный спуск (Stochastic Gradient Descent, или SGD) – это простой, но очень эффективный подход к подгонке линейных классификаторов и регрессоров под выпуклые Функции потерь (Loss Function), такие как Метод опорных векторов (SVM) и Логистическая регрессия (Logistic Regression).

Преимущества:

- Эффективность.
- Простота реализации (множество возможностей для настройки кода).

К недостаткам можно отнести:

- SGD требует ряда гиперпараметров, таких как параметр регуляризации и количество итераций.
- SGD чувствителен к масштабированию функций.

10. Регрессия по методу «лассо» (LASSO, Least Absolute Shrinkage and Selection Operator). Регрессия по методу наименьших квадратов часто может стать неустойчивой, то есть сильно зависящей от обучающих данных, что обычно является проявлением тенденции к переобучению. Избежать такого переобучения помогает регуляризация – общий метод, заключающийся в наложении дополнительных ограничений на искомые параметры, которые могут предотвратить излишнюю сложность модели. Смысл процедуры заключается в «стягивании» в ходе настройки вектора коэффициентов таким образом, чтобы они в среднем оказались несколько меньше по абсолютной величине, чем это было бы при оптимизации по методу наименьших квадратов.

Метод регрессии «Лассо» заключается во введении дополнительного слагаемого регуляризации в функционал оптимизации модели, что часто позволяет получать более устойчивое решение.

Преимущества:

- Более точные и стабильные оценки истинных параметров.
- Уменьшение ошибок выборки и отсутствия выборки.

Метрики эффективности для регрессии оценивают отклонение (расстояние) между предсказанными значениями и реальными. Обычно метрики сравнивают данную модель с тривиальной – моделью, которая всегда предсказывает среднее реальное значение целевой переменной. Модели могут быть точны на 100%, но плохи они могут быть без ограничений.

Коэффициент детерминации (R2) показывает силу связи между двумя случайными величинами. Если модель всегда предсказывает точно, метрика равна 1. Для тривиальной модели – 0. Значение метрики может быть отрицательно, если модель предсказывает хуже, чем тривиальная. Это одна из немногих несимметричных метрик эффективности.

Именно коэффициент детерминации чаще всего используется как метрика по умолчанию, которую можно посмотреть при помощи метода score() у модели регрессии. Этот метод принимает на вход саму обучающую выборку. Но более универсально будет использовать эту метрику независимо от модели. Данная метрика называется r2_score. При использовании этой функции ей надо передавать два вектора целевой переменной – сначала эмпирический, а вторым аргументом - теоретический.

Средняя абсолютная ошибка (mean absolute error, MAE) показывает среднее абсолютное отклонение предсказанных значений от реальных. Чем выше значение MAE, тем модель хуже. У идеальной модели MAE = 0. MAE очень легко интерпретировать – на сколько в среднем ошибается модель.

Средний квадрат ошибки (mean squared error, MSE) показывает средний квадрат отклонений предсказанных значений от реальных. Чем выше значение MSE, тем модель хуже. У идеальной модели MSE = 0. MSE больше учитывает сильные отклонения, но хуже интерпретируется, чем MAE.

Среднеквадратичная ошибка (root mean squared error, RMSE) – это, по сути, корень из MSE. Выражается в тех же единицах, что и целевая переменная. Чаще применяется при статистическом анализа данных. Данная метрика очень чувствительна к аномалиям и выбросам.

Средняя абсолютная процентная ошибка (mean absolute percentage error, MAPE). В ней каждое отклонение оценивается в процентах от истинного значения целевой переменной. Идеальная модель имеет MAPE = 0. Верхний предел – не ограничен. Эта метрика имеет одно критическое преимущество над остальными –

с ее помощью можно сравнивать эффективность моделей на разных обучающих выборках. Ведь если мы возьмем классические метрики (например, MAE), то размер отклонений будет очевидно зависеть от самих данных. А в двух разных выборках и средняя величина скорее всего будет разная. Поэтому метрики MAE, MSE, RMSE не сопоставимы при сравнении предсказаний, сделанных на разных выборках.

Анализируя описанные выше метрики, можно сделать следующие выводы. R2 значение очень интуитивно понятно. Но исследования показывают, что R2 действителен только для линейной регрессии. Однако большинство моделей регрессии, такие, например, как дерево решений или KNN, являются нелинейными моделями. Для нелинейных моделей не следует полностью доверять $R2$. Предпочтительно всегда использовать $R2$ вместе с другими показателями, такими как MAE и RMSE. Когда необходимо уменьшить влияние выбросов, лучше использовать MAE, когда выбросы нельзя игнорировать, лучше использовать RMSE.

Для сравнения моделей будем использовать коэффициент детерминации R2 и средняя абсолютная ошибка MAE, которые показывают, насколько модель улавливает изменение объясняемой переменной и среднее абсолютное отклонение предсказанных значений от реальных.

1.3. Разведочный анализ данных.

Разведочный анализ данных (Exploratory Data Analysis) играет важнейшую роль после получения набора данных и ставит своей целью выяснить, как с ним работать и получить требуемый результат.

Разведочный анализ данных – предварительное исследование датасета с целью определения его основных характеристик, взаимосвязей между признаками, а также сужения набора методов, используемых для создания модели машинного обучения.

Вычислительные методы разведочного анализа данных включают основные статистические методы, а также более сложные, специально разработанные методы многомерного анализа, предназначенные для отыскания закономерностей в многомерных данных. К основным методам разведочного статистического анализа относится процедура анализа распределений переменных, просмотр корреляционных матриц с целью поиска коэффициентов, превосходящих по величине определенные пороговые значения, или анализ многовходовых таблиц частот.

Исследование предоставленных данных по композитам целесообразно начать с анализа числовых статистик (рисунок 4): количества элементов, среднего арифметическое, медианы, стандартного отклонения, минимального и максимального значения, перцентилей.

```
[ ]  1  # Посмотрим числовые статистики с помощью метода describe()
     2  df.describe()
```

	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	Шаг нашивки	Плотность нашивки
count	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000
mean	2.930366	1975.734888	739.923233	110.570769	22.244390	285.882151	482.731833	73.328571	2466.922843	218.423144	0.491691	6.899222	57.153929
std	0.913222	73.729231	330.231581	28.295911	2.406301	40.943260	281.314690	3.118983	485.628006	59.735931	0.500175	2.563467	12.350969
min	0.389403	1731.764635	2.436909	17.740275	14.254985	100.000000	0.603740	64.054061	1036.856605	33.803026	0.000000	0.000000	0.000000
25%	2.317887	1924.155467	500.047452	92.443497	20.608034	259.066528	266.816645	71.245018	2135.850448	179.627520	0.000000	5.080033	49.799212
50%	2.906878	1977.621657	739.664328	110.564840	22.230744	285.896812	451.864365	73.268805	2459.524526	219.198882	0.000000	6.916144	57.341920
75%	3.552660	2021.374375	961.812526	129.730366	23.961934	313.002106	693.225017	75.356612	2767.193119	257.481724	1.000000	8.586293	64.944961
max	5.591742	2207.773481	1911.536477	198.953207	33.000000	413.273418	1399.542362	82.682051	3848.436732	414.590628	1.000000	14.440522	103.988901

Здесь:

- count - количество элементов;
- mean - среднее арифметическое;
- std - стандартное отклонение;
- min, max - минимальное и максимальное значения соответсвенно;
- 25%, 50%, 75% - 25, 50 и 75 процентные перцентили.

Рисунок 4. Описательная статистика датасета.

Следующим по важности шагом является обнаружение недостающих значений: пропусков в данных нет (рисунок 5).

Датасеты имеют разное количество строк: 1023 строки у dataset_bp, 1040 строк у dataset_nup

```
[ ]  1  # Объединеним датесеты в один
     2  # Объединение выполним с помощью функции merge() по индексу с типом объединения INNER
     3  dataset = pd.merge(dataset_bp, dataset_nup,
     4                     left_index=True,
     5                     right_index=True,
     6                     how = "inner")
     7
     8  # Выведем 5 первых позиций нового датасета
     9  dataset.head()
```

	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	Шаг нашивки	Плотность нашивки
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0	0	4.0	57.0
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0	0	4.0	60.0
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0	0	4.0	70.0
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0	0	5.0	47.0
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0	0	5.0	57.0

```
[ ]  1  # Посмотрим размерность объединенного датасета
     2  dataset.shape
```

```
(1023, 13)
```

Объединенный датасет содержит 1023 строки и 13 столбцов. Это означает, что часть данных (а именно 17 строк из датасета dataset_nup) была удалена из таблицы и исключена из дальнейшего исследования.

Рисунок 5. Проверка пропусков в датасете.

Важным способом описания переменной является форма ее распределения, которая показывает, с какой частотой значения переменной попадают в определенные интервалы. Обычно исследователя интересует, насколько точно распределение можно аппроксимировать нормальным.

Гистограмма – это классический инструмент визуализации, который представляет собой распределение одной или нескольких переменных путем подсчета количества наблюдений, попадающих в интервалы разбиения. Гистограмма позволяет качественно оценить различные характеристики распределения, в том числе оценить нормальность эмпирического распределения. На гистограмму также накладывается кривая распределения (рисунок 6).

```
In [29]: # 2. Гистограммы и графики плотности
         n_subplots = len(df.columns)                    # считаем количество графиков
         n_cols = 3                                      # задаем количество столбцов
         n_rows = (n_subplots + n_cols - 1) // n_cols    # вычисляем количество строк

         plt.figure(figsize=(15,20))
         plt.suptitle('Гистограммы и графики плотности', fontsize=20,  y=0.92)
         for i, col in enumerate(df.columns, 1):
             plt.subplot(n_rows, n_cols, i)
             sns.histplot(df[col], kde=True, color='darkmagenta')
             plt.ylabel("")
         plt.show()
```

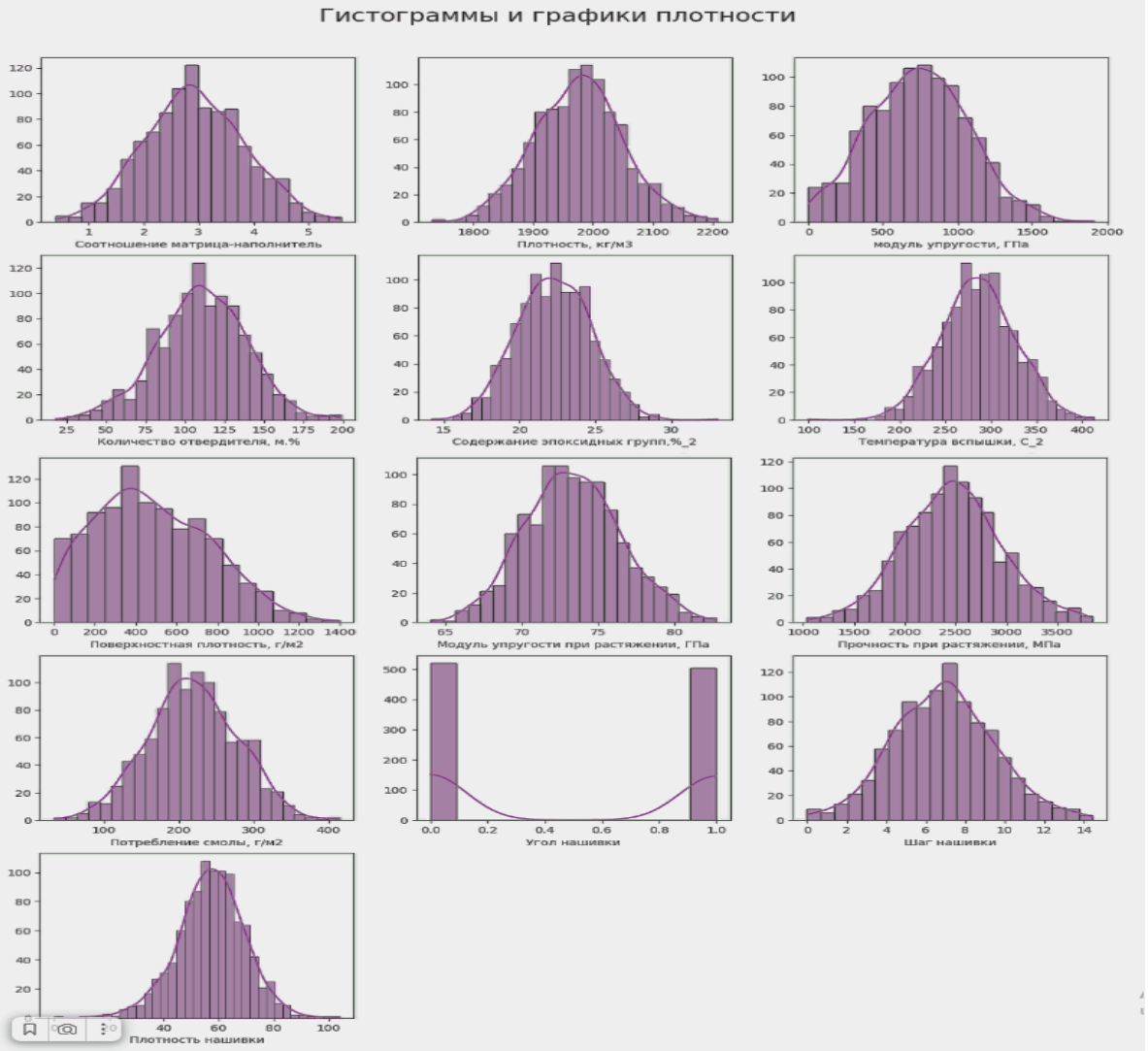

Рисунок 6. Гистограммы и графики плотности.

Анализ графиков показывает, что распределения всех параметров (кроме «Угла нашивки» и «Поверхностной плотности, г/м2») стремятся к нормальному. Параметр «Угол нашивки» имеет два значения: 0 и 1, что соответствует величинам 0 и 90 градусов. Т.к. другие значения данного признака в датасете не встречаются, будем считать его бинарным и категориальным.

Диаграмма «ящик с усами» – график, использующийся в описательной статистике, компактно изображающий одномерное распределение вероятностей. Такой вид диаграммы в удобной форме показывает медиану, нижний и верхний квантили, минимальное и максимальное значение выборки и выбросы.

Выброс (в статистике) – это измерительная точка данных, которая значительно выделяется из общей выборки. Выбросы могут быть вызваны вариативностью измерений или указывать на экспериментальную ошибку; в последнем случае они иногда исключаются из набора данных. Выброс может вызвать серьезные проблемы при статистическом анализе.

В boxplot можно считать нижний и верхний усики как о границы распределения данных. Любые точки данных, которые показывают выше или ниже усов, могут считаться выбросами или аномальными.

Из диаграмм «ящик с усами» (рисунок 7) видно, что выбросы имеют все характеристики, кроме параметра «Угол нашивки».

Влияние выбросов на модель:

- Данные оказались в искаженном формате.
- Изменяет общее статистическое распределение данных с точки зрения среднего значения, дисперсии и т.д.
- Приводит к искажению уровня точности модели.

Из-за вышеуказанных причин необходимо обнаружить и избавиться от выбросов до моделирования набора данных.

```
In [30]: # 3. Диаграммы "ящик с усами"
         n_subplots = len(df.columns)                          # считаем количество графиков
         n_cols = 4                                            # задаем количество столбцов
         n_rows = (n_subplots + n_cols - 1) // n_cols          # вычисляем количество строк

         plt.figure(figsize=(20,25))
         plt.suptitle('Диаграммы "ящик с усами"', fontsize=20,  y=0.92)
         for i, col in enumerate(df.columns, 1):
             plt.subplot(n_rows, n_cols, i)
             sns.boxplot(data=df[col], color='mediumorchid')
             plt.title(df[col].name, size=12)
         plt.show()
```

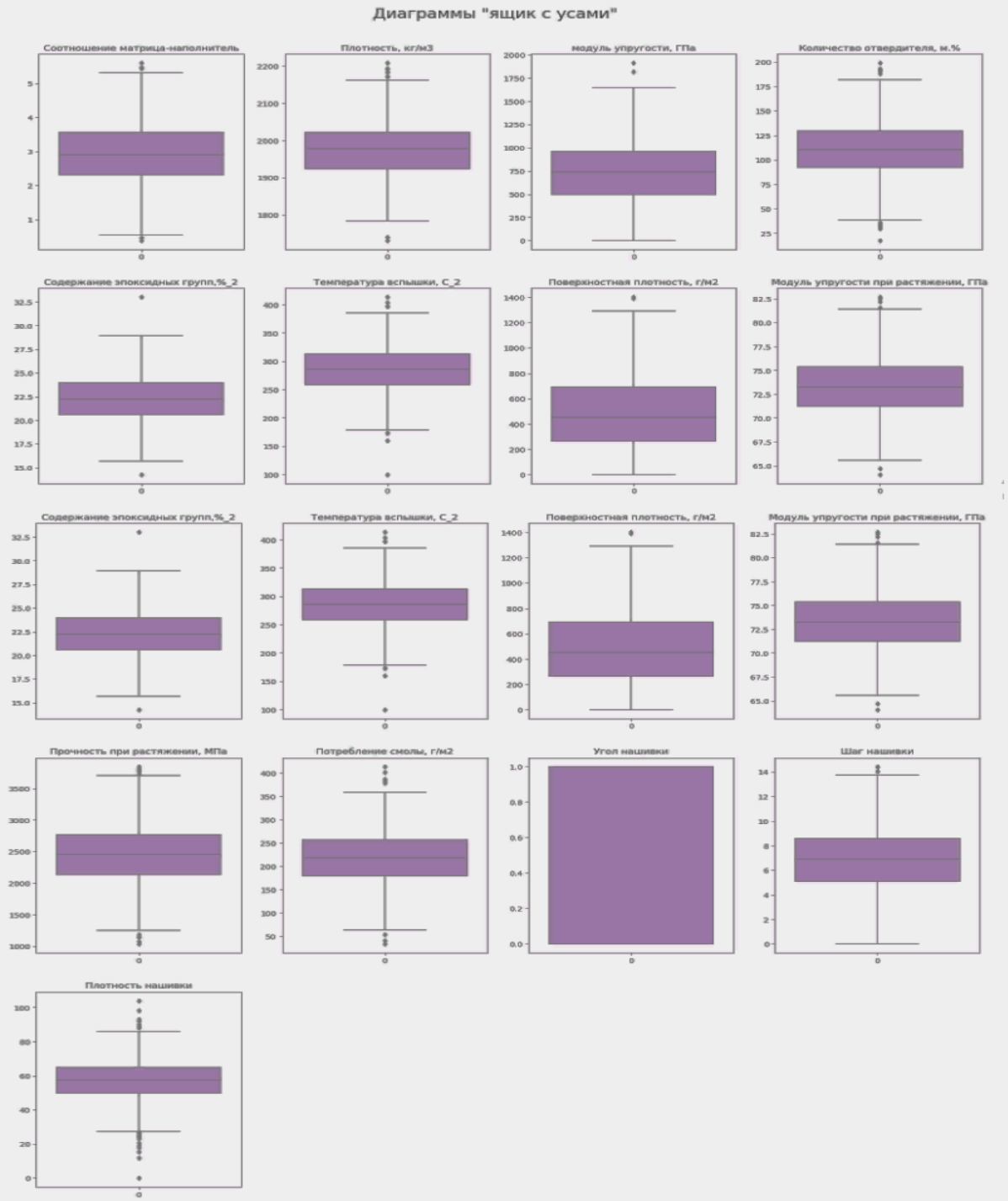

Рисунок 7. Диаграммы «ящик с усами».

Матричная диаграмма рассеяния (рисунок 8) представляет собой все возможные попарные диаграммы рассеяния, представленные в виде большой квадратной матрицы. Диагональные элементы матрицы являются графиками ядерной оценки плотности распределения вероятности каждой из переменных. А остальные элементы – это диаграммы рассеяния переменных относительно друг друга.

```
In [31]: # 4. Попарные графики рассеяния точек с помощью функции sns.pairplot()
         plt.figure(figsize=(20,20))
         sns.set(style='white', palette='RdPu_r')
         sns.pairplot(df)
         plt.show()

         <Figure size 2000x2000 with 0 Axes>
```

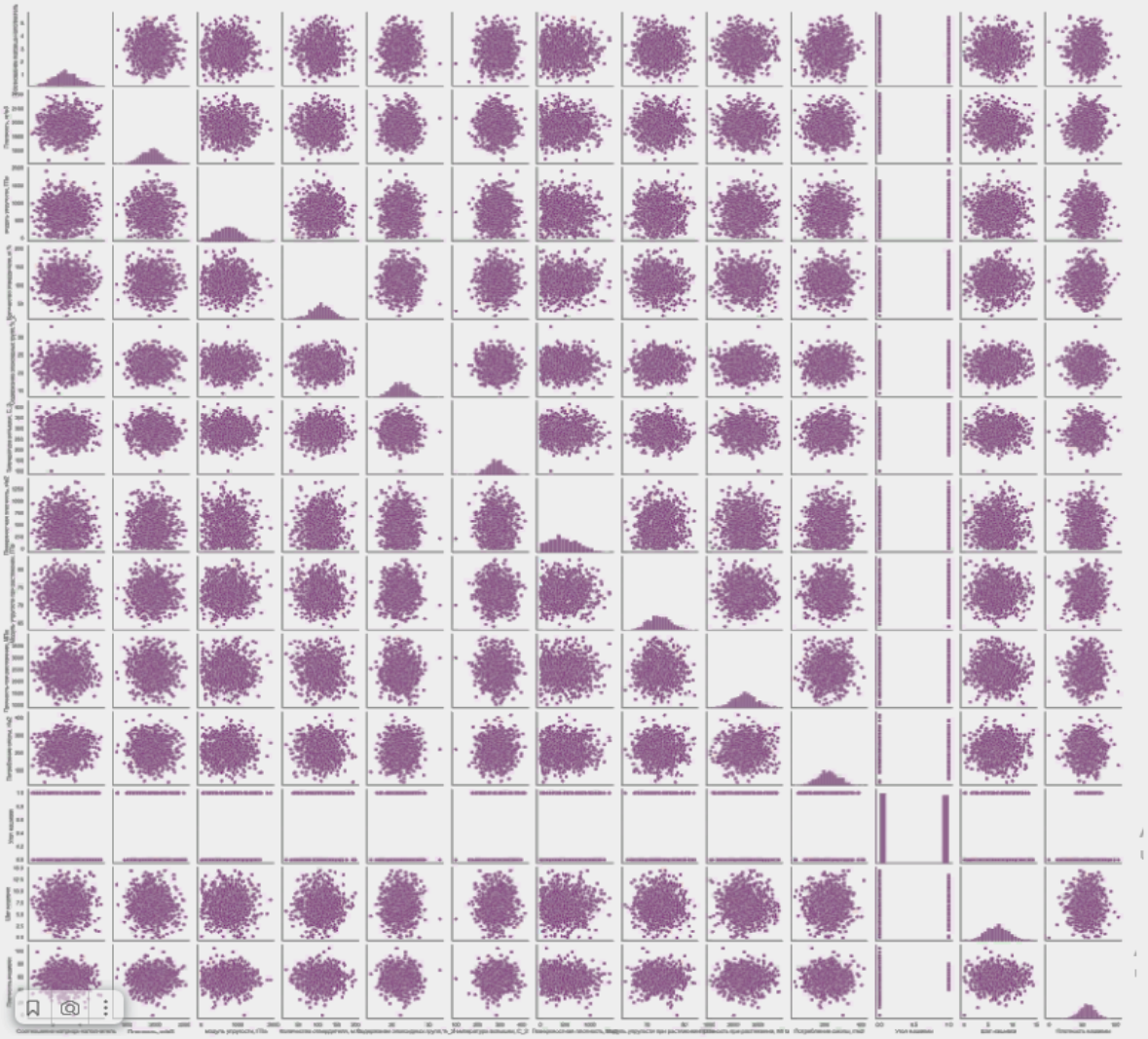

Рисунок 8. Попарные графики рассеяния точек с помощью функции sns.pairplot().

Приведенные выше графики свидетельствуют об отсутствии линейной зависимости между характеристиками композитных материалов.

Далее необходимо установить взаимосвязи (корреляции) между переменными.

Корреляционная зависимость – это согласованные изменения двух (парная корреляционная связь) или большего количества признаков (множественная корреляционная связь). Суть ее заключается в том, что при изменении значения одной переменной происходит закономерное изменение (уменьшение или увеличение) другой(-их) переменной(-ых).

Коэффициент корреляции – двумерная описательная статистика, количественная мера взаимосвязи (совместной изменчивости) двух переменных. Значение коэффициента корреляции может варьироваться от -1 до 1, где -1 указывает на полную отрицательную связь, 0 указывает на отсутствие связи и 1 указывает на полную положительную связь.

В зависимости от того в каких шкалах измерены переменные, для установления наличия корреляционной связи используют следующие коэффициенты корреляции:

1. Коэффициент корреляции Пирсона (рисунок 9): используется, когда переменные измерены в шкале интервалов или отношений, эмпирические данные подчиняются нормальному закону распределения, число данных по каждому признаку одинаково.

   Недостатки коэффициента Пирсона:

   - для распределений, отличных от нормального, перестаёт быть эффективной оценкой коэффициента корреляции;
   - служит мерой только линейной взаимосвязи;
   - чувствителен к выбросам.

```
In [25]: # Проведем анализ зависимостей между переменнными
         # Для этого рассчитаем значения коэффициентов корреляции
         # 1. Коэффициент корреляции Пирсона
         df.corr(method = 'pearson')
```

Out[25]:

	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
Соотношение матрица-наполнитель	1.000000	0.003841	0.031700	-0.006445	0.019766	-0.004776	-0.006272	-0.008411	0.024148	0.072531
Плотность, кг/м3	0.003841	1.000000	-0.009647	-0.035911	-0.008278	-0.020695	0.044930	-0.017602	-0.069981	-0.015937
модуль упругости, ГПа	0.031700	-0.009647	1.000000	0.024049	-0.006804	0.031174	-0.005306	0.023267	0.041868	0.001840
Количество отвердителя, м.%	-0.006445	-0.035911	0.024049	1.000000	-0.000684	0.095193	0.055198	-0.065929	-0.075375	0.007446
Содержание эпоксидных групп,%_2	0.019766	-0.008278	-0.006804	-0.000684	1.000000	-0.009769	-0.012940	0.056828	-0.023899	0.015165
Температура вспышки, С_2	-0.004776	-0.020695	0.031174	0.095193	-0.009769	1.000000	0.020121	0.028414	-0.031763	0.059954
Поверхностная плотность, г/м2	-0.006272	0.044930	-0.005306	0.055198	-0.012940	0.020121	1.000000	0.036702	-0.003210	0.015692
Модуль упругости при растяжении, ГПа	-0.008411	-0.017602	0.023267	-0.065929	0.056828	0.028414	0.036702	1.000000	-0.009009	0.050938
Прочность при растяжении, МПа	0.024148	-0.069981	0.041868	-0.075375	-0.023899	-0.031763	-0.003210	-0.009009	1.000000	0.028602
Потребление смолы, г/м2	0.072531	-0.015937	0.001840	0.007446	0.015165	0.059954	0.015692	0.050938	0.028602	1.000000
Угол нашивки	-0.031073	-0.068474	-0.025417	0.038570	0.008052	0.020695	0.052299	0.023003	0.023398	-0.015334
Шаг нашивки	0.036437	-0.061015	-0.009875	0.014887	0.003022	0.025795	0.038332	-0.029468	-0.059547	0.013394
Плотность нашивки	-0.004652	0.080304	0.056346	0.017248	-0.039073	0.011391	-0.049923	0.006476	0.019604	0.012239

Рисунок 9. Коэффициенты корреляции Пирсона.

2. **Коэффициент корреляции рангов Спирмена** (рисунок 10): используется для измерения взаимозависимости между признаками, значения которых могут быть упорядочены или проранжированы по степени убывания (или возрастания).

```
In [26]: # 2. Коэффициент корреляции Спирмена
         df.corr(method = 'spearman')
```

Out[26]:

	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
Соотношение матрица-наполнитель	1.000000	-0.005542	0.031823	0.002496	0.016358	-0.013635	-0.003896	-0.006459	0.017554	0.054084
Плотность, кг/м3	-0.005542	1.000000	-0.010097	-0.032907	-0.012287	-0.030447	0.055071	-0.032446	-0.069816	-0.025729
модуль упругости, ГПа	0.031823	-0.010097	1.000000	0.033397	0.002016	0.031247	-0.001318	0.007705	0.033750	0.008010
Количество отвердителя, м.%	0.002496	-0.032907	0.033397	1.000000	-0.001652	0.086970	0.050751	-0.064127	-0.067707	-0.006371
Содержание эпоксидных групп,%_2	0.016358	-0.012287	0.002016	-0.001652	1.000000	-0.003641	-0.011232	0.061930	-0.019193	0.014230
Температура вспышки, С_2	-0.013635	-0.030447	0.031247	0.086970	-0.003641	1.000000	0.025441	0.024625	-0.028283	0.051763
Поверхностная плотность, г/м2	-0.003896	0.055071	-0.001318	0.050751	-0.011232	0.025441	1.000000	0.035319	-0.008221	-0.005344
Модуль упругости при растяжении, ГПа	-0.006459	-0.032446	0.007705	-0.064127	0.061930	0.024625	0.035319	1.000000	-0.009970	0.052450
Прочность при растяжении, МПа	0.017554	-0.069816	0.033750	-0.067707	-0.019193	-0.028283	-0.008221	-0.009970	1.000000	0.020831
Потребление смолы, г/м2	0.054084	-0.025729	0.008010	-0.006371	0.014230	0.051763	-0.005344	0.052450	0.020831	1.000000
Угол нашивки	-0.026190	-0.063073	-0.038800	0.030222	0.005714	0.021886	0.055637	0.027458	0.025226	-0.002940
Шаг нашивки	0.032510	-0.045941	-0.011851	0.008440	-0.007421	0.043075	0.037675	-0.014210	-0.071915	0.009288
Плотность нашивки	0.004570	0.080001	0.072628	0.025265	-0.032182	0.008160	-0.032404	-0.003225	0.014950	0.016236

Рисунок 10. Коэффициенты корреляции Спирмена.

3. **Коэффициент корреляции Кендалла** (рисунок 11): применяется, когда переменные измерены в ранговой шкале, но размер выборки мал. Число данных по каждому признаку должно быть одинаковым, не допускается использование равных рангов.

```
In [27]: # 3. Коэффициент корреляции Кендалла
         df.corr(method = 'kendall')
```

Out[27]:

	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
Соотношение матрица-наполнитель	1.000000	-0.003135	0.021247	0.001410	0.010180	-0.009480	-0.002060	-0.004157	0.011614	0.035145
Плотность, кг/м3	-0.003135	1.000000	-0.008059	-0.021963	-0.007758	-0.019947	0.037302	-0.021151	-0.047426	-0.017079
модуль упругости, ГПа	0.021247	-0.008059	1.000000	0.022382	0.002351	0.021028	-0.000442	0.005458	0.022959	0.005169
Количество отвердителя, м.%	0.001410	-0.021963	0.022382	1.000000	0.000010	0.059034	0.033110	-0.043140	-0.046507	-0.003677
Содержание эпоксидных групп,%_2	0.010180	-0.007758	0.002351	0.000010	1.000000	-0.002170	-0.006859	0.041994	-0.013441	0.009756
Температура вспышки, С_2	-0.009480	-0.019947	0.021028	0.059034	-0.002170	1.000000	0.017196	0.016481	-0.019106	0.035313
Поверхностная плотность, г/м2	-0.002060	0.037302	-0.000442	0.033110	-0.006859	0.017196	1.000000	0.024051	-0.005099	-0.004446
Модуль упругости при растяжении, ГПа	-0.004157	-0.021151	0.005458	-0.043140	0.041994	0.016481	0.024051	1.000000	-0.006599	0.034814
Прочность при растяжении, МПа	0.011614	-0.047426	0.022959	-0.046507	-0.013441	-0.019106	-0.005099	-0.006599	1.000000	0.013580
Потребление смолы, г/м2	0.035145	-0.017079	0.005169	-0.003677	0.009756	0.035313	-0.004446	0.034814	0.013580	1.000000
Угол нашивки	-0.021395	-0.051525	-0.031695	0.024690	0.004668	0.017880	0.045452	0.022431	0.020609	-0.002402
Шаг нашивки	0.022723	-0.031220	-0.008305	0.006232	-0.004539	0.029552	0.025514	-0.010024	-0.048049	0.005962
Плотность нашивки	0.002788	0.052935	0.049347	0.016607	-0.021968	0.005268	-0.022320	-0.002600	0.009821	0.010792

Рисунок 11. Коэффициенты корреляции Кендалла.

Коэффициенты корреляции Спирмена и Кендалла используются как меры взаимозависимости между рядами рангов, а не как меры связи между самими переменными.

Коэффициенты Спирмена и Кендалла обладают примерно одинаковыми свойствами, но коэффициент корреляции Кендалла в случае многих рангов, а также при введении дополнительных объектов в ходе исследования имеет определенные вычислительные преимущества.

Для визуализации матриц корреляции можно воспользоваться тепловой картой (рисунок 12).

```
In [35]: # 6. Визуализируем матрицу корреляции в виде тепловой карты
         #sns.set(style='white', palette='RdPu_r')
         f, ax = plt.subplots(figsize=(12,10))
         sns.heatmap(df.corr(), annot=True, ax=ax, square = True, cmap='RdPu')
         plt.show()
```

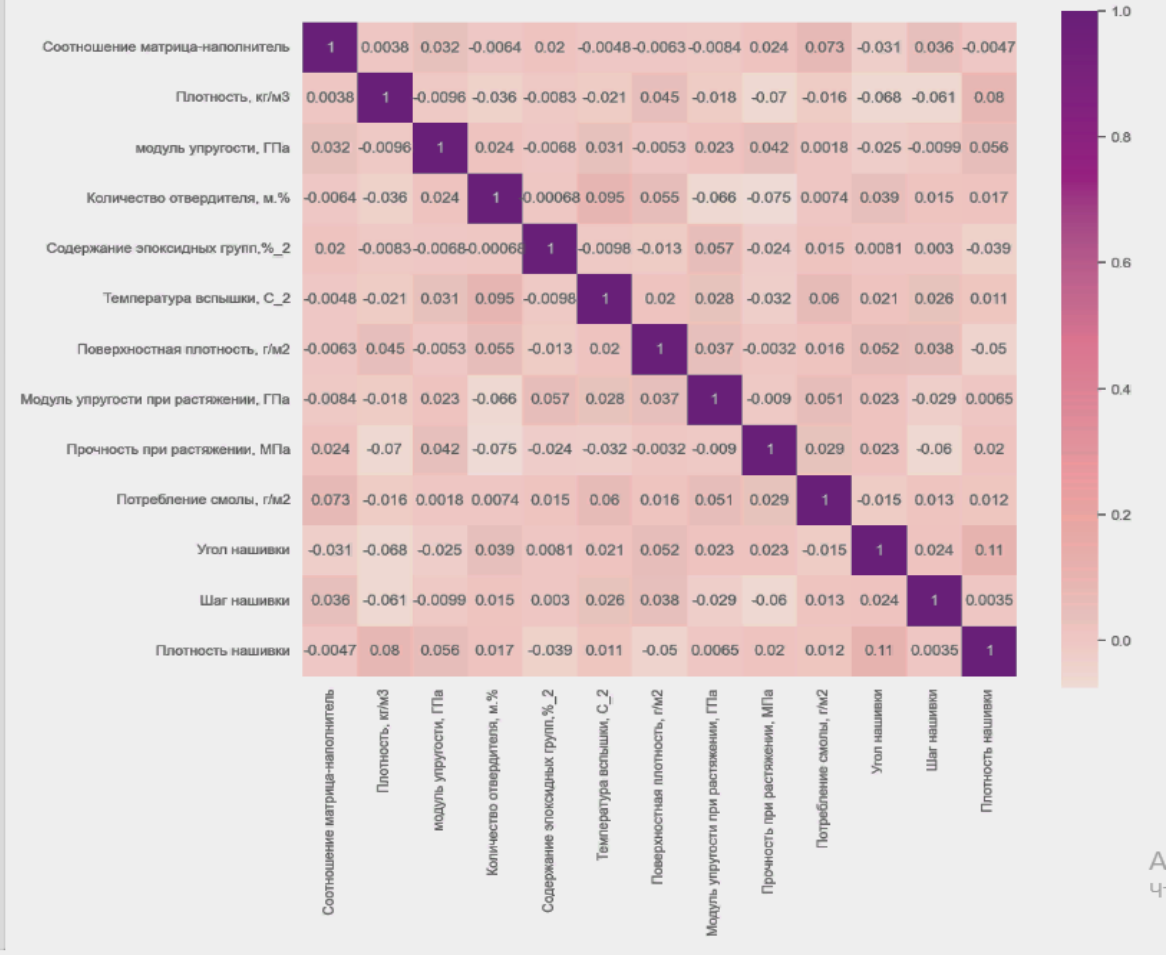

Рисунок 12. Тепловая карта.

Из приведенных выше матриц корреляции и тепловой карты видно, что наибольшая корреляция наблюдается между «Плотностью нашивки» и «Углом нашивки» и равняется 0,11. Другие коэффициенты корреляции близки к 0 и, соответственно, корреляционная зависимость между переменными крайне слабая.

2. Практическая часть.

2.1. Предобработка данных.

В области науки о данных и машинного обучения предварительная обработка значений данных является важным шагом. Первым шагом предварительной обработкой рассмотрим удаление всех выбросов из данных до моделирования.

Для поиска выбросов воспользуемся двумя методами:

1) Стандартное отклонение (правило трех сигм);
2) Метод межквартильного диапазона (IQR - Inter Quartile Range).

Метод 1. Стандартное отклонение (правило трех сигм) (рисунок 13): когда данные подчиняются нормальному распределению, 99,7% значений должны находиться в пределах 3 стандартных отклонений от среднего; когда значение превышает это расстояние, это можно считать выбросом.

Для реализации этого метода применяют z-оценку. Z-оценка показывает количество стандартных отклонений данного значения от среднего. Формула (1) для расчета z-показателя:

$$z = (X - \mu) / \sigma \qquad (1)$$

где: X – это одно необработанное значение данных;

μ – среднее значение;

σ – стандартное отклонение.

Наблюдение можно идентифицировать как выброс (формула (2)), если его z-оценка меньше -3 или больше 3.

Выбросы = наблюдения с z-показателями > 3 или < -3 (2)

```
In [37]: # 1. Поиск аномалий в признаках
         # Метод 1 - Стандартное отклонение (правило трех сигм)
         # Для этого для каждого значения всех параметров необходимо получить его z-оценку

         # Создадим функцию для поиска выбросов
         def detect_outliers_zscore(data):
             outliers = []                          # создаем пустой список, куда будем записывать выбросы
             mean = np.mean(data)
             std = np.std(data)
             for i in data:
                 z_score = (i-mean)/std
                 if (np.abs(z_score) > 3):
                     outliers.append(i)
             return outliers

         sum_of_outliers = 0                        # счетчик количества выбросов
         for col in df.columns:
             df_outliers_zscore = detect_outliers_zscore(df[col])
             sum_of_outliers += len(df_outliers_zscore)
             print("Количество выбросов в столбце ", df[col].name, ": ", len(df_outliers_zscore))
         print("Всего выбросов при использовании метода трех сигм: ", sum_of_outliers)

         Количество выбросов в столбце  Соотношение матрица-наполнитель :  0
         Количество выбросов в столбце  Плотность, кг/м3 :  3
         Количество выбросов в столбце  модуль упругости, ГПа :  2
         Количество выбросов в столбце  Количество отвердителя, м.% :  2
         Количество выбросов в столбце  Содержание эпоксидных групп,%_2 :  2
         Количество выбросов в столбце  Температура вспышки, С_2 :  3
         Количество выбросов в столбце  Поверхностная плотность, г/м2 :  2
         Количество выбросов в столбце  Модуль упругости при растяжении, ГПа :  1
         Количество выбросов в столбце  Прочность при растяжении, МПа :  0
         Количество выбросов в столбце  Потребление смолы, г/м2 :  3
         Количество выбросов в столбце  Угол нашивки :  0
         Количество выбросов в столбце  Шаг нашивки :  0
         Количество выбросов в столбце  Плотность нашивки :  7
         Всего выбросов при использовании метода трех сигм:  25
```

Рисунок 13. Метод 1 – Стандартное отклонение (правило трех сигм).

Метод 2. Метод межквартильного диапазона (IQR - Inter Quartile Range) (рисунок 14): Межквартильный размах (IQRб) – это разница между 75-м процентилем и 25-м процентилем в наборе данных (формула (2)). Он измеряет разброс средних 50% значений.

Другими словами, IQR – это первый квартиль (Q1), вычтенный из третьего квартиля (Q3); эти квартили можно четко увидеть на диаграмме «ящик с усами».

$$IQR = Q3 - Q1 \qquad (2)$$

Наблюдение можно рассматривать как выброс (формула (3), если оно в 1,5 раза превышает межквартильный размах, превышающий третий квартиль (Q3), или в 1,5 раза превышает межквартильный размах, меньше первого квартиля (Q1).

$$\text{Выбросы} = \text{наблюдения} > Q3 + 1{,}5IQR \text{ или } Q1 - 1{,}5IQR \qquad (3)$$

```
In [38]: # Метод 2 - Метод межквартильного диапазона (IQR - Inter Quartile Range):
def detect_outliers_IQR(data):
    outliers = []                          # создаем пустой список, куда будем записывать выбросы
    Q1 = data.quantile(0.25)
    Q3 = data.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR           # нижняя граница
    upper_bound = Q3 + 1.5 * IQR           # верхняя граница
    for i in data:
        if (i <= lower_bound) | (i >= upper_bound):
            outliers.append(i)
    return outliers

sum_of_outliers = 0                        # счетчик количества выбросов
for col in df.columns:
    df_outliers_IQR = detect_outliers_IQR(df[col])
    sum_of_outliers += len(df_outliers_IQR)
    print("Количество выбросов в столбце ", df[col].name, ": ", len(df_outliers_IQR))
print("Всего выбросов при использовании метода межквартильного диапазона: ", sum_of_outliers)

Количество выбросов в столбце  Соотношение матрица-наполнитель :  6
Количество выбросов в столбце  Плотность, кг/м3 :  9
Количество выбросов в столбце  модуль упругости, ГПа :  2
Количество выбросов в столбце  Количество отвердителя, м.% :  14
Количество выбросов в столбце  Содержание эпоксидных групп,%_2 :  2
Количество выбросов в столбце  Температура вспышки, С_2 :  8
Количество выбросов в столбце  Поверхностная плотность, г/м2 :  2
Количество выбросов в столбце  Модуль упругости при растяжении, ГПа :  6
Количество выбросов в столбце  Прочность при растяжении, МПа :  11
Количество выбросов в столбце  Потребление смолы, г/м2 :  8
Количество выбросов в столбце  Угол нашивки :  0
Количество выбросов в столбце  Шаг нашивки :  4
Количество выбросов в столбце  Плотность нашивки :  21
Всего выбросов при использовании метода межквартильного диапазона:  93
```

Рисунок 14. Метод 2 - Метод межквартильного диапазона (IQR).

При использовании метода трех сигм было выявлено 25 выбросов, при использовании метода межквартильного диапазона - 93 выброса.

Подход к поиску выбросов в интервале между квартилями является наиболее часто используемым и наиболее надежным подходом, применяемым в области исследований. Воспользуемся результатами второго метода.

Выбросы из данных удалены после трех этапов очистки (рисунок 15). Размерность датасета стала (922, 13).

```
In [45]: # Визуализируем очищенный датасет с помощью диаграмм "ящик с усами"
         n_subplots = len(df_clean.columns)                    # считаем количество графиков
         n_cols = 4                                            # задаем количество столбцов
         n_rows = (n_subplots + n_cols - 1) // n_cols          # вычисляем количество строк

         plt.figure(figsize=(20,25))
         plt.suptitle('Диаграммы "ящик с усами"', fontsize=20,  y=0.92)
         for i, col in enumerate(df_clean.columns, 1):
             plt.subplot(n_rows, n_cols, i)
             sns.boxplot(data=df_clean[col], color='mediumorchid')
             plt.title(df_clean[col].name, size=12)
         plt.show()
```

Рисунок 15. Диаграмма «ящик с усами» после удаления выбросов.

Тестирование данных на нормальность часто является первым этапом их анализа, так как большое количество статистических методов исходит из предположения нормальности распределения изучаемых данных.

Многие алгоритмы машинного обучения построены на предположении о Гауссовом (нормальном) распределении входных данных. Поэтому для качественной работы таких моделей, обязательна проверка данных на нормальность, а при необходимости – приведение их к распределению, близкому к нормальному.

Проверку выборки на нормальность можно производить несколькими способами:

- построение гистограммы признака,
- построение QQ-графика,
- проведение теста на нормальность.

Тест Шапиро-Уилка (рисунок 16) используется для определения того, соответствует ли выборка нормальному распределению. Если p-значение ниже определенного уровня значимости (0,05), то у нас есть достаточно доказательств, чтобы сказать, что данные выборки не получены из нормального распределения.

```
In [64]: # Проведем тест Шапиро-Уилка на нормальность
         for col in df_clean.columns:
             print(df_clean[col].name, shapiro(df_clean[col]))

         Соотношение матрица-наполнитель ShapiroResult(statistic=0.9971880316734314, pvalue=0.1090434491634368
         9)
         Плотность, кг/м3 ShapiroResult(statistic=0.997011661529541, pvalue=0.08352091163396835)
         модуль упругости, ГПа ShapiroResult(statistic=0.995254397392273, pvalue=0.0058130305260419846)
         Количество отвердителя, м.% ShapiroResult(statistic=0.9966756105422974, pvalue=0.05000615119934082)
         Содержание эпоксидных групп,%_2 ShapiroResult(statistic=0.9977133870124817, pvalue=0.2354103326797485
         4)
         Температура вспышки, С_2 ShapiroResult(statistic=0.9971266984939575, pvalue=0.09941922873258591)
         Поверхностная плотность, г/м2 ShapiroResult(statistic=0.9776217937469482, pvalue=1.0688074730813568e-
         10)
         Модуль упругости при растяжении, ГПа ShapiroResult(statistic=0.9955782890319824, pvalue=0.00941655971
         1098671)
         Прочность при растяжении, МПа ShapiroResult(statistic=0.9973730444908142, pvalue=0.14375117421150208)
         Потребление смолы, г/м2 ShapiroResult(statistic=0.9955164790153503, pvalue=0.008584197610616684)
         Угол нашивки ShapiroResult(statistic=0.6364129781723022, pvalue=2.8589291269154918e-40)
         Шаг нашивки ShapiroResult(statistic=0.9980173110961914, pvalue=0.3559962809085846)
         Плотность нашивки ShapiroResult(statistic=0.9967383742332458, pvalue=0.05505112186074257)
```

Рисунок 16. Тест Шапиро-Уилка до нормализации.

В данном случае нулевая гипотеза отвергается (p-value < 0.05) в случаях:

- модуль упругости, ГПа
- Поверхностная плотность, г/м2
- Модуль упругости при растяжении, ГПа
- Потребление смолы, г/м2
- Угол нашивки

что подразумевает, что распределения не является нормальными.

Однако, проверка нормальности статистическими тестами является очень строгой, т.к. идет сравнение с идеальным распределением. Поэтому несмотря на то, что статистический тест говорит о ненормальности распределения, стоит посмотреть на гистограмму распределения (рисунок 17).

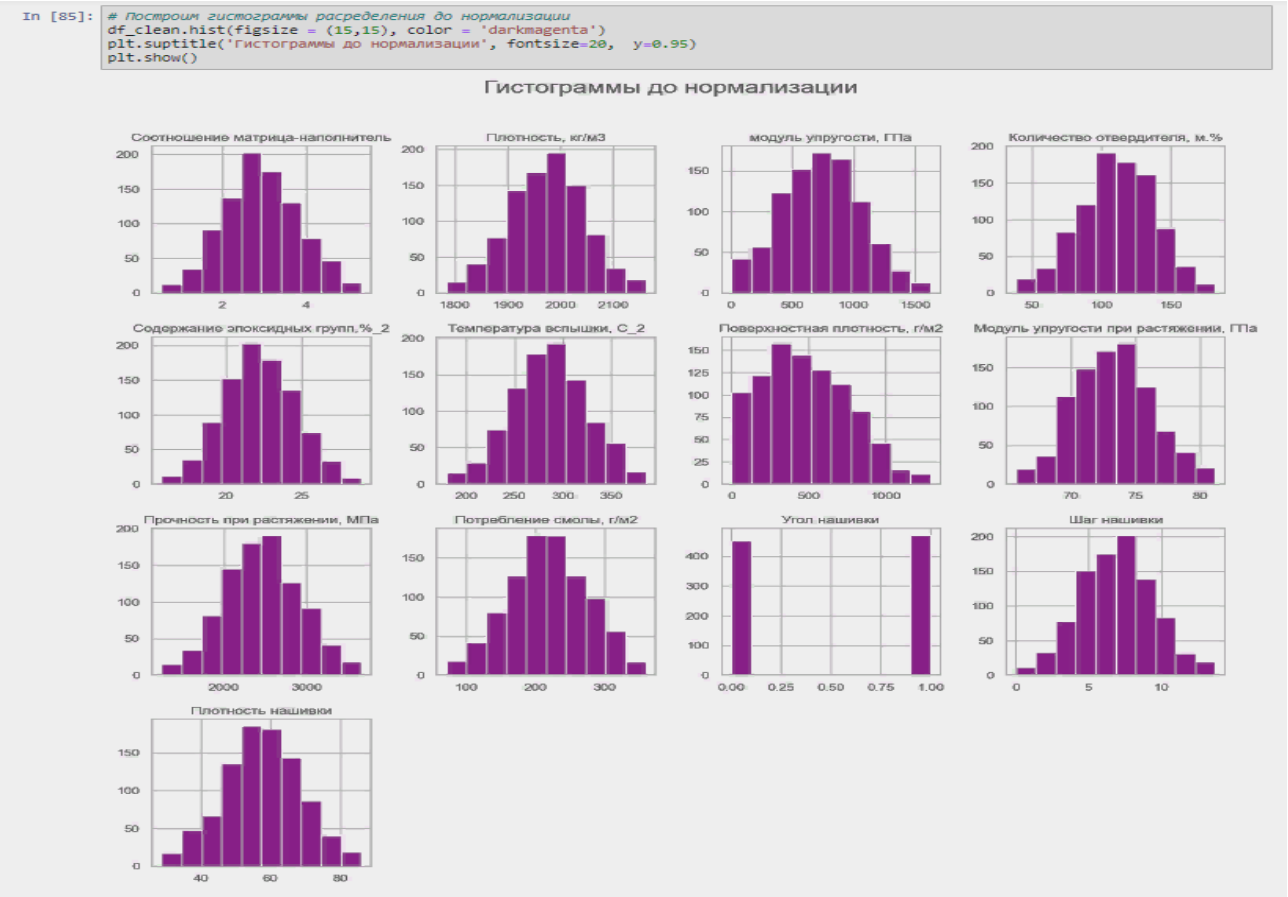

Рисунок 17. Гистограмма распределения до нормализации.

Анализ графиков показывает, что распределения всех параметров (кроме «Угла нашивки» и «Поверхностной плотности, г/м2») стремятся к нормальному. Параметр «Угол нашивки» имеет два значения: 0 и 1, что соответствует величинам 0 и 90 градусов.

График QQ, сокращение от графика «квантиль-квантиль» (рисунок 18), используется для оценки того, потенциально ли набор данных получен из некоторого теоретического распределения.

В большинстве случаев этот тип графика используется для определения того, соответствует ли набор данных нормальному распределению. Если данные распределены нормально, точки на графике QQ будут лежать на прямой диагональной линии. И наоборот, чем больше точки на графике значительно отклоняются от прямой диагональной линии, тем менее вероятно, что набор данных следует нормальному распределению.

```
In [86]: # Построим QQ-графики до нормализации
         n_subplots = len(df_clean.columns)                    # считаем количество графиков
         n_cols = 4                                            # задаем количество столбцов
         n_rows = (n_subplots + n_cols - 1) // n_cols          # вычисляем количество строк

         plt.figure(figsize=(20,25))
         plt.suptitle('Распределение до нормализации', fontsize=20,  y=0.92)
         for i, col in enumerate(df_clean.columns, 1):
             plt.subplot(n_rows, n_cols, i)
             stats.probplot(df_clean[col], dist='norm', plot=plt)
             plt.title(df_clean[col].name, size=12)
         plt.show()
```

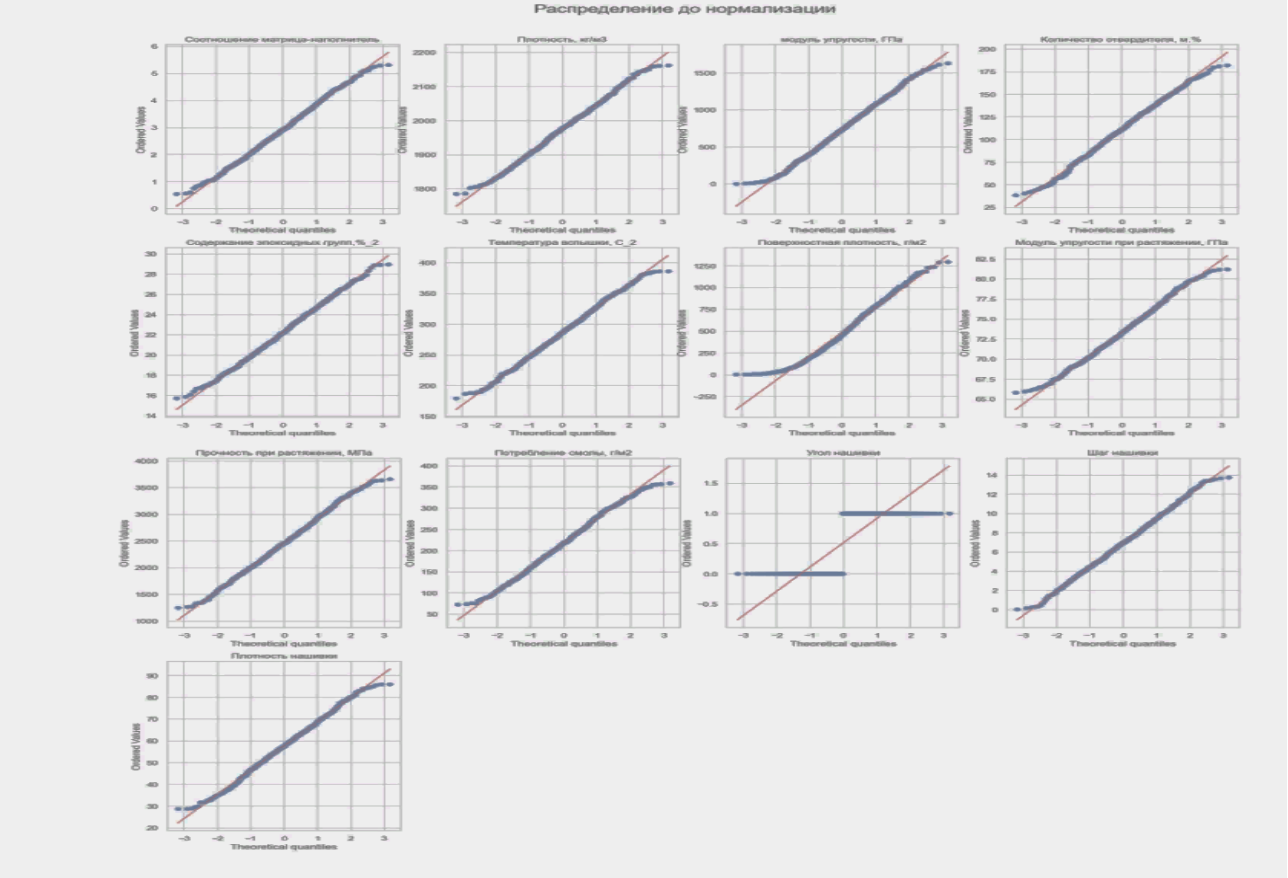

Рисунок 18. QQ-графики до нормализации.

Из графиков видно, что большинство данных близки к нормальному распределению, которое на данном графике представляет красная линия.

Алгоритмы машинного обучения, как правило, работают лучше или сходятся быстрее, когда различные переменные примерно одинаковый масштаб и близки к нормальному распределению. Поэтому общепринятой практикой является нормализация и стандартизация данных перед обучением на них моделей машинного обучения. Кроме того, в данном датасете присутствуют численные признаки разных масштабов, поэтому необходимо отмасштабировать признаки (рисунок 19).

```
In [47]: # Построим графики распределения всех параметров
         plt.figure(figsize=(15,8))
         sns.set(context='notebook', style='whitegrid')
         sns.kdeplot(data=df_clean)
         plt.show()
```

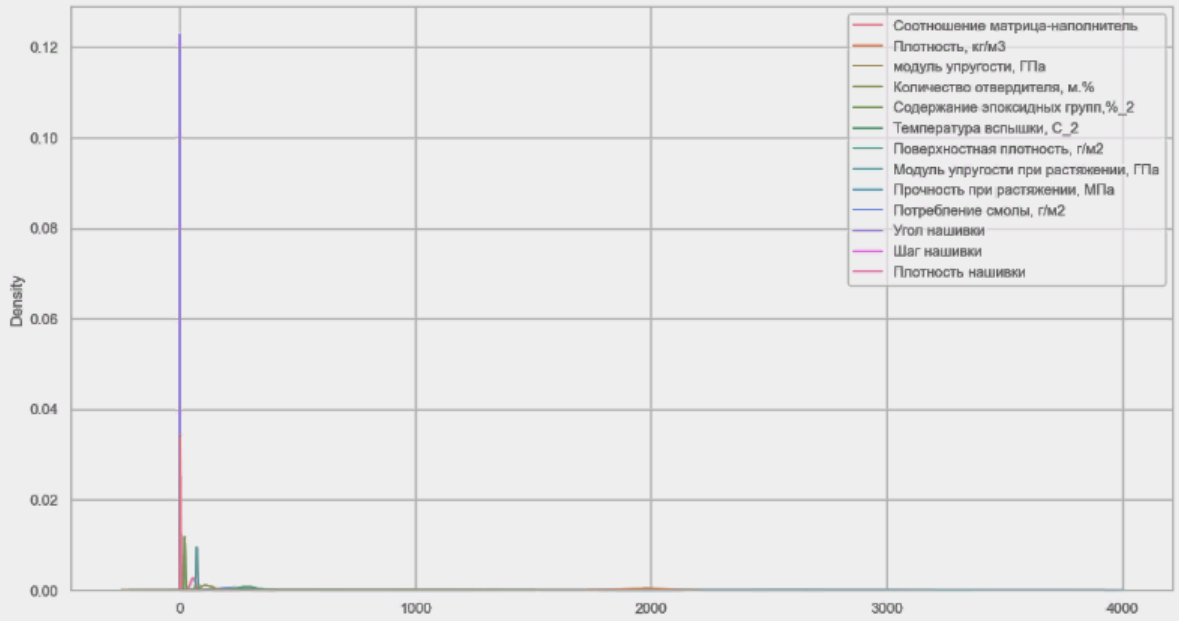

Рисунок 19. Распределение переменных датасета.

Нормализация – процедура предобработки входной информации (обучающих, тестовых и валидационных выборок, а также реальных данных), при которой значения признаков во входном векторе приводятся к некоторому заданному диапазону, например, [0...1] или [-1...1].

Ключевая цель нормализации – приведение данных в самых разных единицах измерения и диапазонных значениях к единому виду, который позволит сравнить данные между собой и использовать для расчета схожести объектов в выборке. До начала обучения модели необходимо привести все признаки к равному влиянию друг на друга. В Python-библиотеке Scikit-learn есть для этого классы MinMaxScaler и RobustScaler.

Стандартизация – приведения данных к определенному формату и представлению, которые обеспечивают их корректное применение в многомерном анализе. Стандартизация позволяет устранить возможное влияние отклонений по

каждому признаку и приводит все исходные значения в датасете к набору значений к нормальному распределению с математическим ожиданием равным 0 и стандартным отклонением равным 1. В результате получается стандартизированная шкала, которая определяет место каждого значения в наборе данных и измеряет его отклонение от среднего в единицах стандартного отклонения.

Недостатком стандартизации является возможность присутствия в этих шкалах отрицательных значений, что может привести к потере логики анализа данных. Отрицательные значения могут исключаться путем дополнительных преобразований.

Для стандартизации данных в Scikit-learn есть класс StandardScaler, который применяет к каждому из атрибутов следующее: вычитает из значений среднее и делит полученную разность на стандартное отклонение.

Для работы с датасетом был выбран метод PowerTransformer. Преобразование удаляет сдвиг из распределения данных, чтобы сделать распределение более нормальным (гауссовским). Популярными примерами являются лог-преобразование (положительные значения) или обобщенные версии, такие как преобразование Бокса-Кокса (положительные значения) или преобразование Йео-Джонсона (положительные и отрицательные значения).

После форматирования данных с помощью функции MinMaxScaler все параметры имеют одинаковый относительный масштаб (рисунки 20-21). Относительные пробелы между значениями каждого объекта были сохранены. Максимальное значение признака равно 1, минимальное - 0.

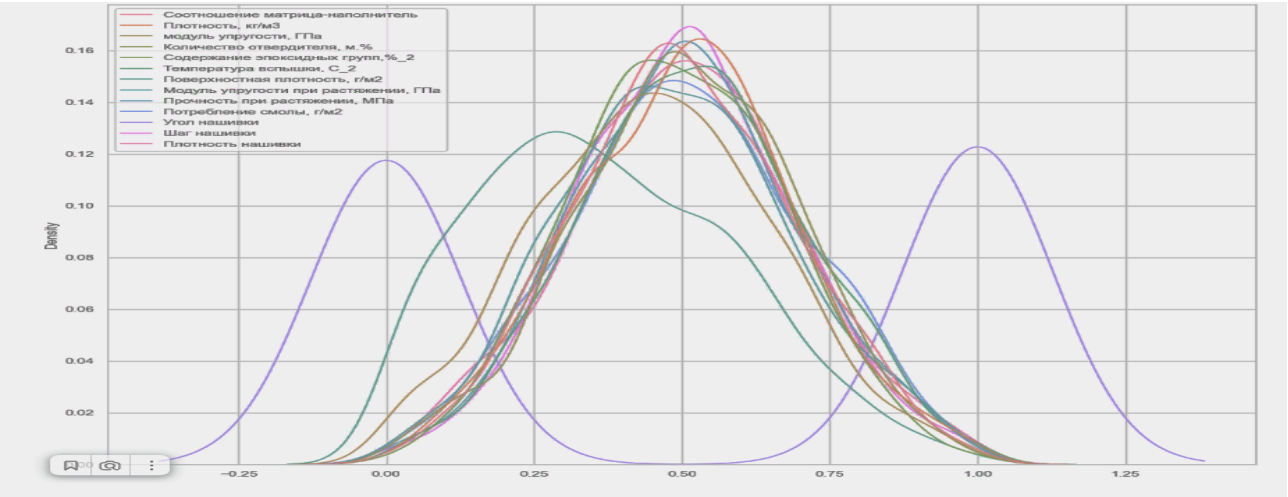

Рисунок 20. Распределение переменных датасета после масштабирования при помощи функции MinMaxScaler().

Рисунок 21. Гистограмма распределения после масштабирования при помощи функции MinMaxScaler().

2.2. Разработка и обучение модели.

Разработка модели машинного обучения для прогнозирования таких характеристик композитных материалов, как модуль упругости при растяжении и прочность при растяжении, включает следующие этапы:

- Разделение нормализованных данных на обучающую и тестовую выборку: 30% данных оставлено на тестирование моделей, на остальных происходит обучение моделей.

```
In [64]: # Разделим датасет на тренировочную и тестовую выборки.
         # При построении модели 30% данных оставим на тестирование модели, на остальных происходит обучение моделей.
         X_train_elastic, X_test_elastic, y_train_elastic, y_test_elastic = train_test_split(
                                                                            df_norm.drop(['Модуль упругости при растяжении, ГПа', 'Прочнос
                                                                            df_norm[['Модуль упругости при растяжении, ГПа']],
                                                                            test_size = 0.3,
                                                                            random_state = 42,
                                                                            shuffle = True
                                                                            )

In [66]: # Посмотрим размерность тренировочной и тестовой выборок
         print('Размер тренировочного датасета: {}\nРазмер тестового датасета:{}'.format(X_train_elastic.shape, X_test_elastic.shape))

         Размер тренировочного датасета: (645, 11)
         Размер тестового датасета:(277, 11)
```

Рисунок 21-1. Разделение данных на обучающую и тестовую выборку.

- Анализ работы различных моделей на стандартных параметрах. Для задачи прогнозирования модуля упругости при растяжении (рисунки 22-31) и прочности при растяжении используем описанные ранее модели:

1. **Метод К-ближайших соседей (рисунок 22).**

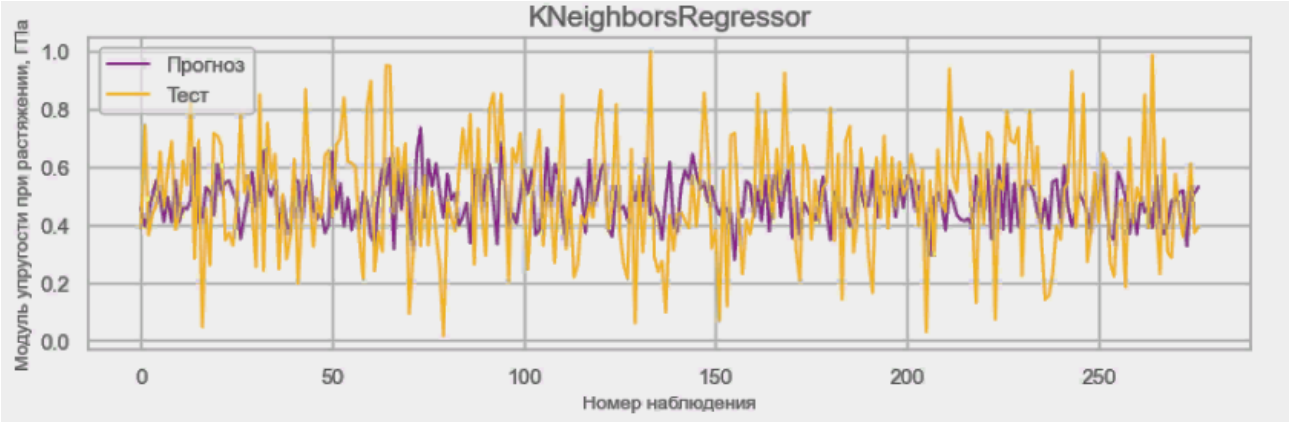

Рисунок 22. Прогнозные и тестовые результаты для KNeighborsRegressor().

2. **Метод опорных векторов.**

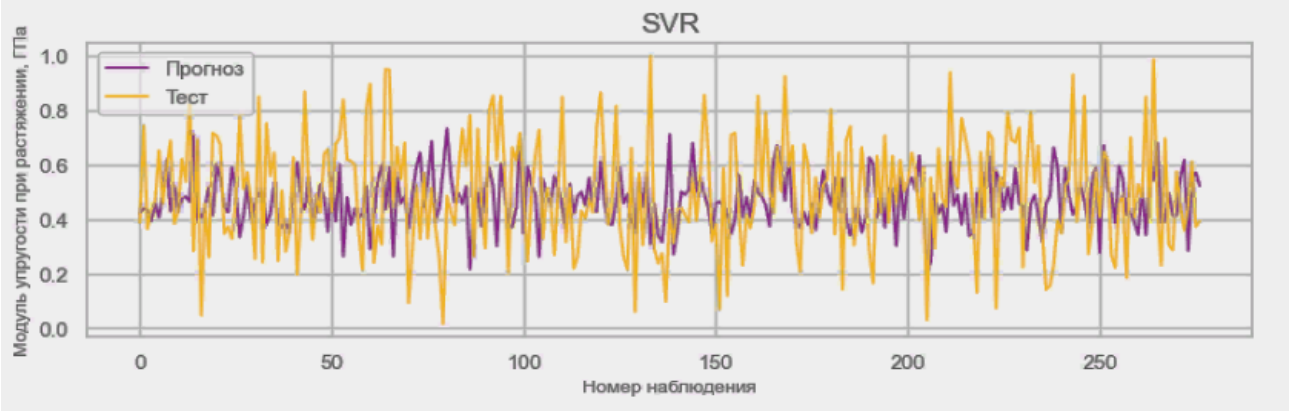

Рисунок 23. Прогнозные и тестовые результаты для SVR().

3. **Линейная регрессия.**

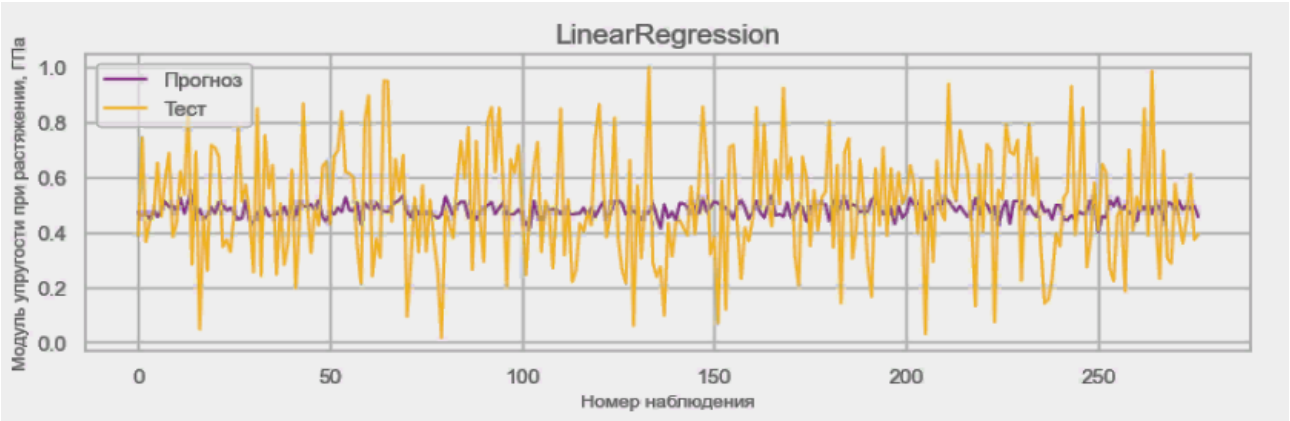

Рисунок 24. Прогнозные и тестовые результаты для LinearRegression().

4. **Дерево решений.**

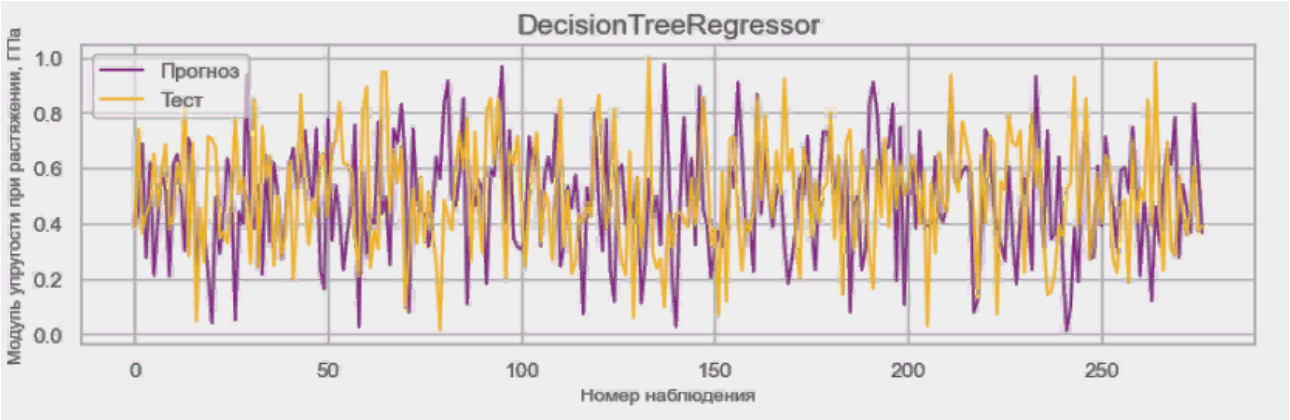

Рисунок 25. Прогнозные и тестовые результаты для DecisionTreeRegressor().

5. **AdaBoost.**

Рисунок 26. Прогнозные и тестовые результаты для AdaBoostRegressor

6. **Градиентный бустинг.**

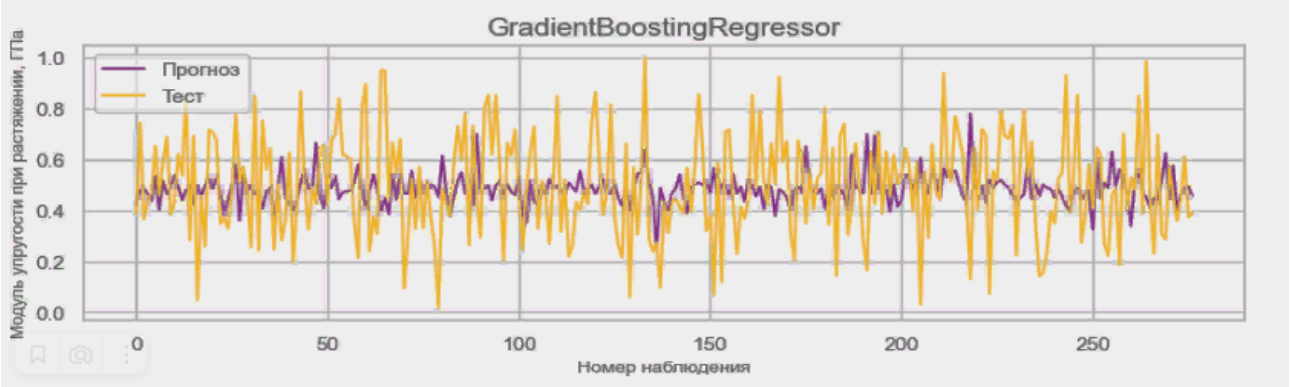

Рисунок 27. Прогнозные и тестовые результаты для GradientBoostingRegressor()

7. **XGBoost.**

Рисунок 28. Прогнозные и тестовые результаты для XGBRegressor().

8. **Случайный лес.**

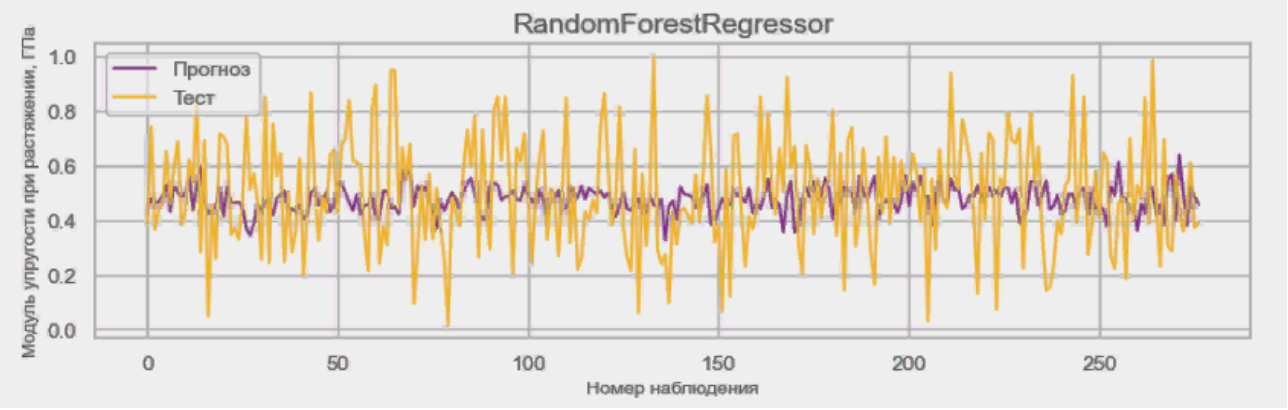

Рисунок 29. Прогнозные и тестовые результаты для RandomForestRegressor().

9. **Стохастический градиентный спуск.**

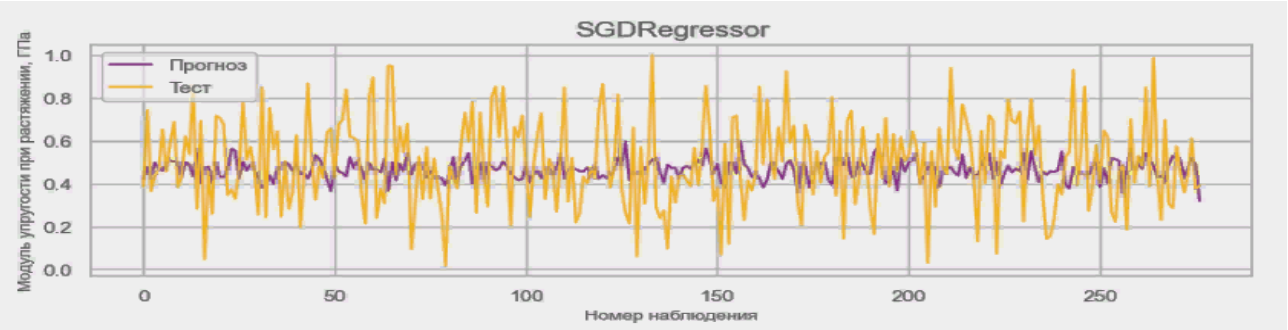

Рисунок 30. Прогнозные и тестовые результаты для SGDRegressor().

10.Метод регрессии «Lasso».

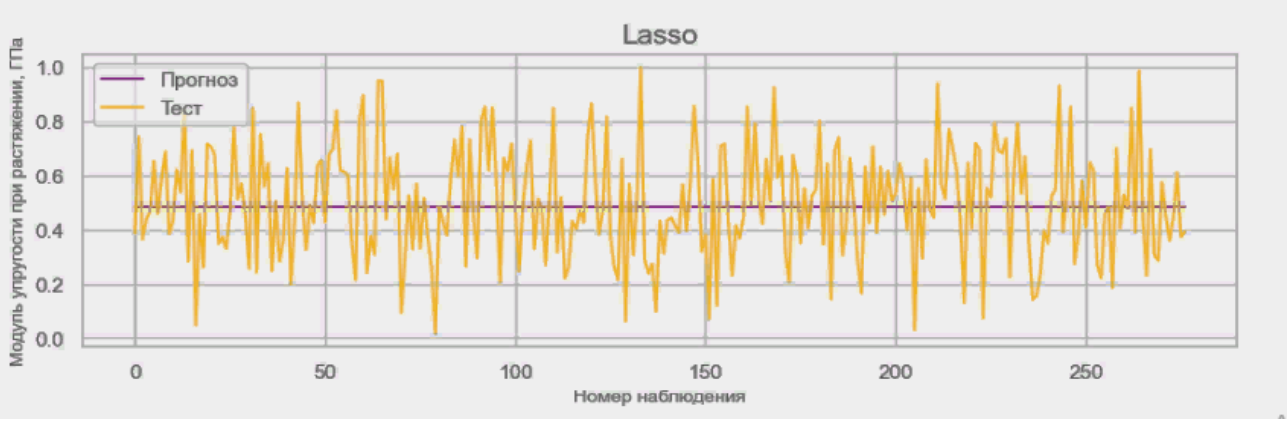

Рисунок 31. Прогнозные и тестовые результаты для Lasso().

- Сравнение моделей по следующим метрикам: коэффициент детерминации R2, средняя абсолютная ошибка (MAE), средний квадрат ошибки (MSE) и среднеквадратичная ошибка (RMSE) (рисунок 32).

```
In [70]: # Рассчитаем метрики для следующих моделей:
         KNR = KNeighborsRegressor()
         SVReg = SVR()
         LR = LinearRegression()
         DTR = DecisionTreeRegressor()
         Abr = AdaBoostRegressor(base_estimator=DecisionTreeRegressor())
         Gbr = GradientBoostingRegressor()
         XgbR = XGBRegressor()
         RFR = RandomForestRegressor(n_estimators = 20, max_depth = 10, random_state = 42)
         SGDR = SGDRegressor()
         LassoR = Lasso(alpha=0.1)

         Model_Comparision_Train_Test([KNR, SVReg, LR, DTR, Abr, Gbr, XgbR, RFR, SGDR, LassoR], X_train_elastic, np.ravel(y_train_elastic
```

Out[70]:

Model	R2_score	MAE	MSE	RMSE	MAPE
KNeighborsRegressor	-0.1435 (0.197)	0.18 (0.14)	0.05 (0.03)	0.22 (0.17)	0.65 (2088968184903.12)
SVR	-0.2554 (0.4649)	0.18 (0.11)	0.05 (0.02)	0.23 (0.14)	0.66 (1409875462373.38)
LinearRegression	-0.0245 (0.0172)	0.17 (0.15)	0.04 (0.04)	0.21 (0.19)	0.65 (3530183592566.66)
DecisionTreeRegressor	-0.8755 (1.0)	0.22 (0.0)	0.08 (0.0)	0.28 (0.0)	0.72 (0.0)
AdaBoostRegressor	-0.095 (0.9999)	0.17 (0.0)	0.05 (0.0)	0.21 (0.0)	0.65 (0.0)
GradientBoostingRegressor	-0.1196 (0.4548)	0.17 (0.11)	0.05 (0.02)	0.22 (0.14)	0.68 (1828505402718.78)
XGBRegressor	-0.1728 (0.9989)	0.18 (0.0)	0.05 (0.0)	0.22 (0.01)	0.7 (28797473498.31)
RandomForestRegressor	-0.0506 (0.6056)	0.17 (0.1)	0.04 (0.01)	0.21 (0.12)	0.65 (1298604470286.13)
SGDRegressor	-0.0632 (-0.0241)	0.17 (0.16)	0.04 (0.04)	0.21 (0.19)	0.64 (3547015966533.43)
Lasso	-0.0089 (0.0)	0.17 (0.16)	0.04 (0.04)	0.21 (0.19)	0.65 (3362407323544.2)

Рисунок 32. Метрики для тестовых и тренировочных данных – тренировочные метрики находятся в ().

Отрицательные значение коэффициента детерминации означают слабую обобщающую способность моделей. Если R2 отрицательна, то модель работает хуже, чем простой подсчет среднего.

Лучшие показатели R2 и MAE на тестовой выборке у алгоритма регрессии «Lasso», на тренировочной выборке – у алгоритмов DecisionTreeRegressor и AdaBoostRegressor.

- Поиск гиперпараметров моделей с помощью поиска по сетке с перекрестной проверкой.

Задачу выбора модели усложняет тот факт, что у многих типов моделей существуют разные вариации, особые параметры. Гиперпараметр модели – это численное значение, которое влияет на работу модели, но не подбирается в процессе обучения. Они задаются при определении модели и должны оставаться неизменными до схождения алгоритма обучения.

Поиск по сетке – полный перебор всех комбинаций значений гиперпараметров для поиска оптимальных значений. Для его организации надо задать список гиперпараметров и их конкретных значений (рисунок 33). Поиск по сетке имеет экспоненциальную сложность. Чем больше параметров и значений задать, тем лучше получится модель, но дольше поиск. Рекомендуется использовать кросс-валидацию. По умолчанию используется оценка модели, встроенная в сам объект модели через метод score, то есть точность (accuracy) для классификации и коэффициент детерминации (R2) для регрессии (рисунок 34).

```
# Создаем словарь с наборами гиперпараметров всех моделей
all_params = {'kneighborsregressor':       {'kneighborsregressor__n_neighbors': [i for i in range(1, 201, 2)],
                                            'kneighborsregressor__weights':     ['uniform', 'distance'],
                                            'kneighborsregressor__algorithm':   ['auto', 'ball_tree', 'kd_tree', 'brute']
                                            },
              'svr':                       {'svr__kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
                                            'svr__C':      [0.01, 0.1, 1],
                                            'svr__gamma':  [0.01, 0.1, 1]
                                            },
              'linearregression':          {'linearregression__fit_intercept': [True, False]},
              'decisiontreeregressor':     {'decisiontreeregressor__max_depth':         [3, 5, 7, 9, 11, 13, 15],
                                            'decisiontreeregressor__min_samples_leaf':  [1, 2, 5, 10, 20, 50, 100, 150, 200],
                                            'decisiontreeregressor__min_samples_split': [200, 250, 300],
                                            'decisiontreeregressor__max_features':      ['auto', 'sqrt', 'log2']
                                            },
              'adaboostregressor':         {'adaboostregressor__base_estimator__max_depth':        [i for i in range(2, 11, 2)],
                                            'adaboostregressor__base_estimator__min_samples_leaf': [5,  10],
                                            'adaboostregressor__n_estimators':                     [10, 50, 100, 250, 1000],
                                            'adaboostregressor__learning_rate':                    [0.01, 0.05, 0.1, 0.5]
                                            },
              'gradientboostingregressor': {'gradientboostingregressor__learning_rate': [0.01,  0.02, 0.03, 0.04],
                                            'gradientboostingregressor__subsample'    : [0.9, 0.5, 0.2, 0.1],
                                            'gradientboostingregressor__n_estimators' : [100, 500, 1000, 1500],
                                            'gradientboostingregressor__max_depth'    : [4, 6, 8, 10]
                                            },
              'xgbregressor':              {'xgbregressor__learning_rate':    [0.05, 0.10, 0.15],
                                            'xgbregressor__max_depth':        [ 3, 4, 5, 6, 8],
                                            'xgbregressor__min_child_weight': [ 1, 3, 5, 7],
                                            'xgbregressor__gamma':            [ 0.0, 0.1, 0.2],
                                            'xgbregressor__colsample_bytree': [ 0.3, 0.4]
                                            },
              'randomforestregressor':     {'randomforestregressor__n_estimators':     [30, 100, 200, 300, 500],
                                            'randomforestregressor__max_depth':        [1, 2, 3, 4, 5, 6, 7, 8],
                                            'randomforestregressor__min_samples_leaf': [1, 2],
                                            'randomforestregressor__max_features':     ['auto', 'sqrt', 'log2']
                                            },
              'sgdregressor':              {'sgdregressor__penalty': ['l2', 'l1', 'elasticnet', None],
                                            'sgdregressor__alpha':   [0.0001, 0.001, 0.01, 0.1]
                                            },
              'lasso':                     {'lasso__alpha': [0.01, 0.02, 0.1, 0.2, 0.03, 0.3, 0.05, 0.5, 0.07, 0.7, 1]}
             }
```

Рисунок 33. Набор гиперпараметров всех моделей.

```
    # Выполняем обучение и оценку качества с помощью кросс-валидации,
    model = GridSearchCV(estimator = pipe, param_grid = current_params,
                         scoring = 'r2', cv = 10, n_jobs = -1)
    model.fit(x_train, y_train)

    print('{} Лучшее значение R2 на тренировочной выборке: {:.4f}'.format(all_models[regressor], model.score(x_train, y_train)))
    print('{} Лучшее значение R2 на тестовой выборке {:.4f}'.format(all_models[regressor], model.score(x_test, y_test)))
    print('{} Лучшее значение R2 на перекрестной проверке: {:.4f}'.format(all_models[regressor], model.best_score_))
    print('{} Лучшие параметры модели: {}'.format(all_models[regressor], model.best_params_))
    print('\n_____________________________________________________________________________________________\n')
    if model.score(x_test, y_test) > best_score:
            best_score = model.score(x_test, y_test)
            best_model = model.best_estimator_
            best_regressor = regressor
print('Регрессор с лучшим значением R2 = {:.4f} на тестовой выборке: {}\n'.format(best_score, all_models[best_regressor]))
print('Лучший алгоритм:\n{}\n'.format(best_model))
```

```
In [75]: Model_Selection(X_train_elastic, np.ravel(y_train_elastic), X_test_elastic, np.ravel(y_test_elastic))
```

```
KNeighborsRegressor() Лучшее значение R2 на тренировочной выборке: 0.0091
KNeighborsRegressor() Лучшее значение R2 на тестовой выборке -0.0105
KNeighborsRegressor() Лучшее значение R2 на перекрестной проверке: -0.0021
KNeighborsRegressor() Лучшие параметры модели: {'kneighborsregressor__algorithm': 'auto', 'kneighborsregressor__n_neighbors':
199, 'kneighborsregressor__weights': 'uniform'}

_____________________________________________________________________________________________

SVR() Лучшее значение R2 на тренировочной выборке: 0.0326
SVR() Лучшее значение R2 на тестовой выборке -0.0146
SVR() Лучшее значение R2 на перекрестной проверке: -0.0031
SVR() Лучшие параметры модели: {'svr__C': 0.01, 'svr__gamma': 1, 'svr__kernel': 'rbf'}

_____________________________________________________________________________________________

LinearRegression() Лучшее значение R2 на тренировочной выборке: 0.0172
LinearRegression() Лучшее значение R2 на тестовой выборке -0.0245
LinearRegression() Лучшее значение R2 на перекрестной проверке: -0.0307
LinearRegression() Лучшие параметры модели: {'linearregression__fit_intercept': True}
```

Рисунок 34. Вывод лучших гиперпараметров для каждой модели.

Лучшим алгоритмом для прогнозирования модуля упругости при растяжении при использовании функции GridSearchCV() выбран регрессор AdaBoostRegressor со значением R2 = -0.0009 на тестовой выборке (рисунок 35).

```
Регрессор с лучшим значением R2 = -0.0009 на тестовой выборке: AdaBoostRegressor(base_estimator=DecisionTreeRegressor())

Лучший алгоритм:
Pipeline(steps=[('adaboostregressor',
                 AdaBoostRegressor(base_estimator=DecisionTreeRegressor(max_depth=2,
                                                                        min_samples_leaf=10),
                                   learning_rate=0.01, n_estimators=10))])
```

Рисунок 35. Вывод лучшего алгоритма прогнозирования параметров.

2.3. Тестирование модели.

После определения лучших параметров для каждой модели произведено тестирование моделей на тренировочном и тестовом наборе данных.

Для сравнения моделей будем использовать коэффициент детерминации R2 и средняя абсолютная ошибка MAE, которые показывают, насколько модель улавливает изменение объясняемой переменной и среднее абсолютное отклонение предсказанных значений от реальных.

Коэффициенты детерминации (R2) отрицательные у всех моделей прогнозирования модуля упругости при растяжении (рисунок 36). Это означает, что прогнозы моделей хуже, чем предсказание среднего значения. Модели плохо обучаются на тренировочной выборке и, соответственно, плохо предсказывают значения для тестовой выборки. Коэффициент MAE одинаков у всех моделей.

In [108]:
```
# Рассчитаем метрики для моделей с их лучшими параметрами:
KNR_best = KNeighborsRegressor(algorithm='auto', n_neighbors=199, weights='uniform')
SVReg_best = SVR(C=0.01, gamma=1, kernel='rbf')
LR_best = LinearRegression(fit_intercept=True)
DTR_best = DecisionTreeRegressor(max_depth=9, max_features='sqrt', min_samples_leaf=200, min_samples_split=250)
Abr_best = AdaBoostRegressor(base_estimator=DecisionTreeRegressor(max_depth=2, min_samples_leaf=10), learning_rate=0.01, n_estima
Gbr_best = GradientBoostingRegressor(learning_rate=0.01, max_depth=4, n_estimators=100, subsample=0.5)
XgbR_best = XGBRegressor(colsample_bytree=0.3, gamma=0.2, learning_rate=0.05, max_depth=3, min_child_weight=5)
RFR_best = RandomForestRegressor(max_depth=1, max_features='log2', min_samples_leaf=2, n_estimators=100)
SGDR_best = SGDRegressor(alpha=0.1, penalty='l1')
LassoR_best = Lasso(alpha=0.01)

Model_Comparision_Train_Test([KNR_best, SVReg_best, LR_best, DTR_best, Abr_best, Gbr_best, XgbR_best, RFR_best, SGDR_best, LassoR
```

Out[108]:

Model	R2_score	MAE	MSE	RMSE	MAPE
KNeighborsRegressor	-0.0105 (0.0091)	0.17 (0.15)	0.04 (0.04)	0.21 (0.19)	0.65 (3324567063626.71)
SVR	-0.0146 (0.0326)	0.17 (0.15)	0.04 (0.04)	0.21 (0.19)	0.65 (3293971898525.6)
LinearRegression	-0.0245 (0.0172)	0.17 (0.15)	0.04 (0.04)	0.21 (0.19)	0.65 (3530183592566.66)
DecisionTreeRegressor	0.006 (0.0061)	0.17 (0.15)	0.04 (0.04)	0.2 (0.19)	0.65 (3479792181714.39)
AdaBoostRegressor	-0.0068 (0.0171)	0.17 (0.15)	0.04 (0.04)	0.2 (0.19)	0.66 (3400849732982.94)
GradientBoostingRegressor	-0.0104 (0.1557)	0.17 (0.14)	0.04 (0.03)	0.21 (0.18)	0.65 (2976673273693.22)
XGBRegressor	-0.0218 (0.0918)	0.17 (0.15)	0.04 (0.03)	0.21 (0.18)	0.66 (3274354675248.97)
RandomForestRegressor	-0.0079 (0.0184)	0.17 (0.15)	0.04 (0.04)	0.21 (0.19)	0.65 (3443052806421.78)
SGDRegressor	-0.01 (-0.0)	0.17 (0.16)	0.04 (0.04)	0.21 (0.19)	0.65 (3354326516915.18)
Lasso	-0.0089 (0.0)	0.17 (0.16)	0.04 (0.04)	0.21 (0.19)	0.65 (3362407323544.2)

Рисунок 36. Метрики для тестовых и тренировочных данных после подбора гиперпараметров – тренировочные метрики находятся в () – для прогнозирования модуля упругости при растяжении.

Все использованные модели плохо справились с поставленной задачей прогнозирования модуля упругости при растяжении. Получен неудовлетворительный результат.

При разработке модели машинного обучения для прогнозирования прочности при растяжении также получены модели со слабой обобщающей способностью (рисунок 37). Полученный результат не решает поставленную задачу.

```
In [112]: # Рассчитаем метрики для моделей с их лучшими параметрами:
KNR_best = KNeighborsRegressor(algorithm='auto', n_neighbors=159, weights='distance')
SVReg_best = SVR(C=0.01, gamma=0.1, kernel='poly')
LR_best = LinearRegression(fit_intercept=True)
DTR_best = DecisionTreeRegressor(max_depth=15, max_features='sqrt', min_samples_leaf=100, min_samples_split=250)
Abr_best = AdaBoostRegressor(base_estimator=DecisionTreeRegressor(max_depth=2, min_samples_leaf=10), learning_rate=0.01, n_estima
Gbr_best = GradientBoostingRegressor(learning_rate=0.01, max_depth=8, n_estimators=100, subsample=0.1)
XgbR_best = XGBRegressor(colsample_bytree=0.3, gamma=0.2, learning_rate=0.05, max_depth=3, min_child_weight=5)
RFR_best = RandomForestRegressor(max_depth=1, max_features='log2', min_samples_leaf=2, n_estimators=30)
SGDR_best = SGDRegressor(alpha=0.1, penalty='l1')
LassoR_best = Lasso(alpha=0.01)

Model_Comparision_Train_Test([KNR_best, SVReg_best, LR_best, DTR_best, Abr_best, Gbr_best, XgbR_best, RFR_best, SGDR_best, LassoR
```

Out[112]:

Model	R2_score	MAE	MSE	RMSE	MAPE
KNeighborsRegressor	-0.002 (1.0)	0.15 (0.0)	0.04 (0.0)	0.19 (0.0)	8176160847752.77 (0.0)
SVR	-0.0012 (0.0012)	0.15 (0.15)	0.04 (0.03)	0.19 (0.19)	8151410332325.56 (0.69)
LinearRegression	-0.0035 (0.017)	0.15 (0.15)	0.04 (0.03)	0.19 (0.19)	9069348467609.18 (0.68)
DecisionTreeRegressor	-0.0371 (0.0234)	0.16 (0.15)	0.04 (0.03)	0.2 (0.18)	8702601068544.95 (0.68)
AdaBoostRegressor	-0.0124 (0.0594)	0.16 (0.14)	0.04 (0.03)	0.19 (0.18)	8793829166673.36 (0.67)
GradientBoostingRegressor	-0.0008 (0.1549)	0.15 (0.14)	0.04 (0.03)	0.19 (0.17)	8607548068569.25 (0.64)
XGBRegressor	0.0005 (0.1155)	0.15 (0.14)	0.04 (0.03)	0.19 (0.18)	8709751775321.21 (0.65)
RandomForestRegressor	0.0018 (0.0209)	0.15 (0.15)	0.04 (0.03)	0.19 (0.18)	8367939336339.68 (0.68)
SGDRegressor	-0.0002 (-0.0)	0.15 (0.15)	0.04 (0.03)	0.19 (0.19)	8168241775487.68 (0.69)
Lasso	-0.0001 (0.0)	0.15 (0.15)	0.04 (0.03)	0.19 (0.19)	8183534682026.43 (0.69)

Рисунок 37. Метрики для тестовых и тренировочных данных после подбора гиперпараметров – тренировочные метрики находятся в () – для прогнозирования прочности при растяжении.

Лучшим алгоритмом для прогнозирования прочности при растяжении при использовании функции GridSearchCV() выбран регрессор XGBRegressor со значением R2 = 0.0005 на тестовой выборке .

С учетом полученных неудовлетворительных результатов в качестве прогноза для модуля упругости при растяжении и прочности при растяжении можно использовать среднее значение признака.

2.4. Нейронная сеть для рекомендации соотношения «матрица – наполнитель».

Нейронная сеть – это метод в искусственном интеллекте, который учит компьютеры обрабатывать данные таким же способом, как и человеческий мозг. Это тип процесса машинного обучения, называемый глубоким обучением, который использует взаимосвязанные узлы или нейроны в слоистой структуре, напоминающей человеческий мозг. Он создает адаптивную систему, с помощью которой компьютеры учатся на своих ошибках и постоянно совершенствуются. Таким образом, искусственные нейронные сети пытаются решать сложные задачи с более высокой точностью.

Часто архитектуры нейронных сетей строят в виде последовательности слоев, начиная с входного и заканчивая выходным. Теоретически, число скрытых слоев может быть сколь угодно большим. Для описания такой модели, как раз применяется класс Sequential, который используется в нейронной сети для рекомендации соотношения «матрица – наполнитель».

Объект оболочки Keras для использования в качестве регрессионной оценки называется KerasRegressor. Создадим экземпляр и передадим ему как имя функции для создания модели нейронной сети, так и некоторые параметры для дальнейшей их передачи в функции компиляции и обучения модели. Далее с помощью функции GridSearchCV() произведем сравнение моделей нейронной сети (рисунок 38). Выбор модели осуществляется по лучшему значению коэффициента KerasRegressor.score(), который возвращает коэффициент детерминации прогноза (также известный как оценка R2) (рисунок 39).

```
In [118]: # Создадим функцию для генерации слоев нейронной сети
          def create_NN_model(layers, activation, drop, opt):
                  model= Sequential()
                  for i, neurons in enumerate(layers):
                      if i==0:
                          model.add(Dense(neurons, input_dim=X_train_matrix.shape[1], activation = activation))  # входной слой
                      else:
                          model.add(Dense(neurons, activation))                                                  # добавляем полносвязный
                          model.add(Dropout(drop))                                                               # исключаем переобучения
                      model.add(Dense(1))                                                                        # выходной слой

                      # Компиляция модели: определяем метрики и алгоритм оптимизации
                      model.compile(loss = 'mse', optimizer = opt, metrics = ['mae'])

                      return model

In [119]: # Построим нейронную сети с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 5 (cv = 5)
          # Воспользуемся методом GridSearchCV()

          reg = KerasRegressor(model = create_NN_model, layers = [128], activation = 'relu', drop = 0.1, opt = 'Adam', verbose = 2)

          # Зададим параметры для модели
          param_grid = {'activation': ['relu', 'softmax', 'sigmoid'],
                        'layers': [[128, 64, 16], [128, 128, 64, 32], [128, 128, 64, 16]],
                          'opt': ['Adam', 'SGD'],
                         'drop': [0.0, 0.1, 0.2],
                     'batch_size': [10, 20, 40],
                        'epochs': [10, 50, 100]
                     }

          # Произведем поиск лучших параметров
          grid = GridSearchCV(estimator = reg,
                              param_grid = param_grid,
                              cv = 5,
                              verbose = 0,
                              n_jobs = -1)

          grid_result = grid.fit(X_train_matrix, np.ravel(y_train_matrix))
```

Рисунок 38. Построение нейронной сети с помощью поиска по сетке с перекрестной проверкой.

```
In [120]: print('Лучший коэффициент R2: {:.4f} при использовании модели с параметрами {} \n'.format(grid_result.best_score_, grid_result.be

          Лучший коэффициент R2: -0.0021 при использовании модели с параметрами {'activation': 'softmax', 'batch_size': 10, 'drop': 0.0,
          'epochs': 10, 'layers': [128, 128, 64, 32], 'opt': 'SGD'}

In [121]: # Создадим модель с полученными значениями
          best_model = Sequential()
          best_model.add(Dense(128, input_dim = X_train_matrix.shape[1], activation = 'softmax'))      # входной слой
          best_model.add(Dense(128, activation = 'softmax'))                                           # добавляем полносвязный слой
          best_model.add(Dropout(0.0))                                                                 # исключаем переобучения
          best_model.add(Dense(64, activation = 'softmax'))                                            # добавляем полносвязный слой
          best_model.add(Dropout(0.0))                                                                 # исключаем переобучения
          best_model.add(Dense(32, activation = 'softmax'))                                            # добавляем полносвязный слой
          best_model.add(Dropout(0.0))                                                                 # исключаем переобучения
          best_model.add(Dense(1))                                                                     # выходной слой

          # Компиляция модели: определяем метрики и алгоритм оптимизации
          best_model.compile(loss = 'mse',
                             optimizer = 'SGD',
                             metrics = ['mae'])

          # Обучение модели
          best_history = best_model.fit(X_train_matrix, np.ravel(y_train_matrix),
                                        epochs=10,
                                        batch_size=10,
                                        verbose=1,
                                        validation_split=0.2)
```

Рисунок 39. Выбор и построение лучшей модели.

Структура нейронной сети, выбранной с помощью функции KerasRegressor(), приведена на рисунке 40.

```
In [123]: # Структура нейронной сети
          best_model.summary()

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 128)               1664

 dense_3 (Dense)             (None, 128)               16512

 dropout (Dropout)           (None, 128)               0

 dense_4 (Dense)             (None, 64)                8256

 dropout_1 (Dropout)         (None, 64)                0

 dense_5 (Dense)             (None, 32)                2080

 dropout_2 (Dropout)         (None, 32)                0

 dense_6 (Dense)             (None, 1)                 33

=================================================================
Total params: 28,545
Trainable params: 28,545
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 40. Структура нейронной сети.

После выбора модели произведем ее обучение на тренировочном датасете. На рисунке 41 представлен график потерь на тренировочной и тестовой выборках, на рисунке 42 – визуализация прогнозных результатов для модели.

```
In [125]: # Построим график потерь на тренировочной и тестовой выборках
          model_loss_plot(best_history)
```

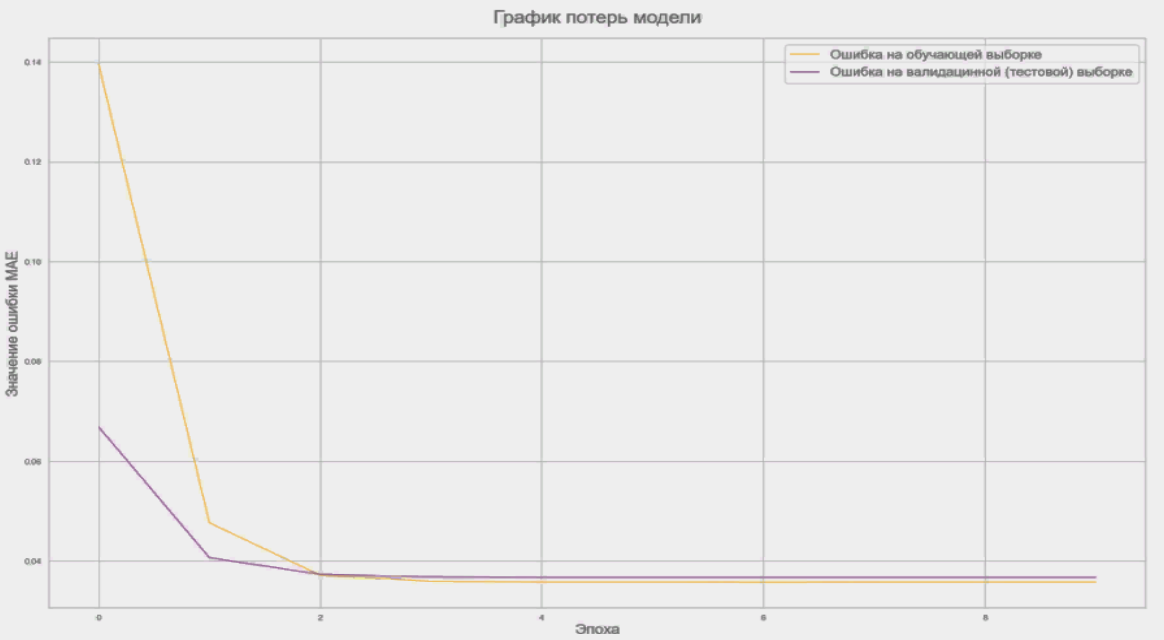

Рисунок 41. График потерь на тренировочной и тестовой выборках.

```
In [126]: # Создадим функцию для визуализации прогнозных результатов y_pred для модели
          def Comparision_Visualization_NN(y_test, y_pred):
              plt.figure(figsize=(15,7))
              plt.title(f'Тест и прогноз, Соотношение матрица-наполнитель', size=15)
              plt.plot(np.ravel(y_test), label = 'Тест', color = "gold")
              plt.plot(y_pred, label = 'Прогноз', color = "darkmagenta")
              plt.xlabel("Номер наблюдения", size=10)
              plt.ylabel("Соотношение матрица-наполнитель", size=10)
              plt.legend(loc='best')
              plt.show()

In [127]: Comparision_Visualization_NN(y_test_matrix, best_model.predict(X_test_matrix))

          9/9 [==============================] - 0s 2ms/step
```

Рисунок 42. Визуализация прогнозных результатов для модели.

Результат прогноза нейронной сети неудовлетворительный. Значение функции потерь – среднего квадрата ошибки (R2) – составило 0.0336, а средней абсолютной ошибки (MAE) – 0.1491 (рисунок 43). Полученная модель нейронной сети плохо справились с поставленной задачей прогнозирования соотношения «матрица-наполнитель».

```
In [91]: # Оценка получившейся модели
         best_model.evaluate(X_test_matrix, np.ravel(y_test_matrix), verbose = 1)

         9/9 [==============================] - 0s 4ms/step - loss: 0.0336 - mae: 0.1491

Out[91]: [0.0336499847471714, 0.14910170435905457]
```

Рисунок 43. Оценка работы модели.

2.5. Разработка приложения для прогнозирования соотношения «матрица – наполнитель».

Создадим два варианта приложения:

- веб-приложение;
- консольное приложение.

Разработка веб-приложения в фреймворке Flask включает следующие этапы:

1. Инициализация приложения Flask и загрузка модели машинного обучения, а также необходимых масштабаторов (т.к. при обучении модели были использованы нормализованные данные) (рисунок 44).

```
In [145]: # Инициализируем приложение Flask
          app = Flask(__name__)

In [133]: # Загружаем модель и масштабаторы
          nn_model = keras.models.load_model('C:/Users/Asus/Documents/УЧЕБА_Data Science/0. ВКР/Application//model_matrix/')
          scaler_x = load('C:/Users/Asus/Documents/УЧЕБА_Data Science/0. ВКР/Application//minmax_scl_x.pkl')
          scaler_y = load('C:/Users/Asus/Documents/УЧЕБА_Data Science/0. ВКР/Application//minmax_scl_y.pkl')
```

Рисунок 44. Создание Flask приложения.

2. Определение маршрута приложения для страницы веб-приложения по умолчанию: маршруты относятся к шаблонам URL-адресов приложения. @app.route('/') – это декоратор Python, который Flask предоставляет для простого назначения URL-адресов в создаваемом приложении функциям.

   Декоратор сообщает нашему @app, что всякий раз, когда пользователь посещает домен приложения (localhost: 5000 для локальных серверов) с заданным .route(), выполнять функцию home() (рисунок 44). Flask использует библиотеку шаблонов Jinja для визуализации шаблонов. В создаваемом приложении используются шаблоны для рендеринга HTML, который будет отображаться в браузере.
3. Перенаправление API для прогнозирования соотношения «матрица-наполнитель». Создатся новый маршрут приложения («/predict»),

который считывает ввод из формы «main.html» и при нажатии кнопки «Рассчитать» выводит результат с помощью render_template (рисунок 44).

4. Запуск сервера Flask вызывается app.run(), и веб-приложение размещается локально на [localhost: 5000]. «Debug = True» и «Use_reloader = False» гарантирует, что не нужно загружать и запускать созданное приложение каждый раз, когда будут внесены изменения, и дает возможность просто обновить нашу веб-страницу, чтобы увидеть изменения, пока сервер все еще работает (рисунок 45).

```
In [138]: # Определяем маршрут приложения для страницы веб-приложения по умолчанию
          @app.route('/')
          def home():
              return render_template('main.html')

In [139]: # Создаем новый маршрут приложения, который считывает ввод из формы «main.html»
          # и при нажатии кнопки "Рассчитать" выводит результат
          @app.route('/predict', methods = ['POST'])
          def predict():
              int_features = [float(x) for x in request.form.values()]
              X = scaler_x.transform(np.array(int_features).reshape(1,-1))
              prediction = nn_model.predict(X)
              output = scaler_y.inverse_transform(prediction)
              return render_template('main.html',
                                     prediction_text = 'Прогнозное значение соотношения "матрица - наполнитель": {}'.format(output[0][0]))

In [140]: # Запуск сервера Flask
          if __name__ == "__main__":
              app.run(debug=True, use_reloader=False)

           * Serving Flask app "__main__" (lazy loading)
           * Environment: production
             WARNING: This is a development server. Do not use it in a production deployment.
             Use a production WSGI server instead.
           * Debug mode: on

           * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
          127.0.0.1 - - [31/Mar/2023 11:00:17] "GET / HTTP/1.1" 200 -
```

Рисунок 45. Запуск Flask приложения.

Проект сохраняется в папке с именем «Application» включает следующее (рисунок 46):

model_matrix	30.03.2023 18:14	Папка с файлами	
templates	31.03.2023 10:39	Папка с файлами	
Application.ipynb	31.03.2023 11:25	Файл "IPYNB"	13 КБ
minmax_scl_x.pkl	30.03.2023 16:32	Файл "PKL"	2 КБ
minmax_scl_y.pkl	30.03.2023 16:32	Файл "PKL"	1 КБ

Рисунок 46. Содержание Flask приложения.

- Папка «model_matrix» с моделью нейронной сети;
- Папка «templates» с файлом main.html;

- minmax_scl_x.pkl, minmax_scl_y.pkl – сохранные нормализаторы MinMaxScaler();
- Application.ipynb – созданное приложение Flask.

При запуске приложения открывается локальный сервер на порту 5000 (рисунок 45). Рекомендуется сначала запустить приложение на локальном сервере и проверить его функциональность, прежде чем размещать его в интернете на облачной платформе.

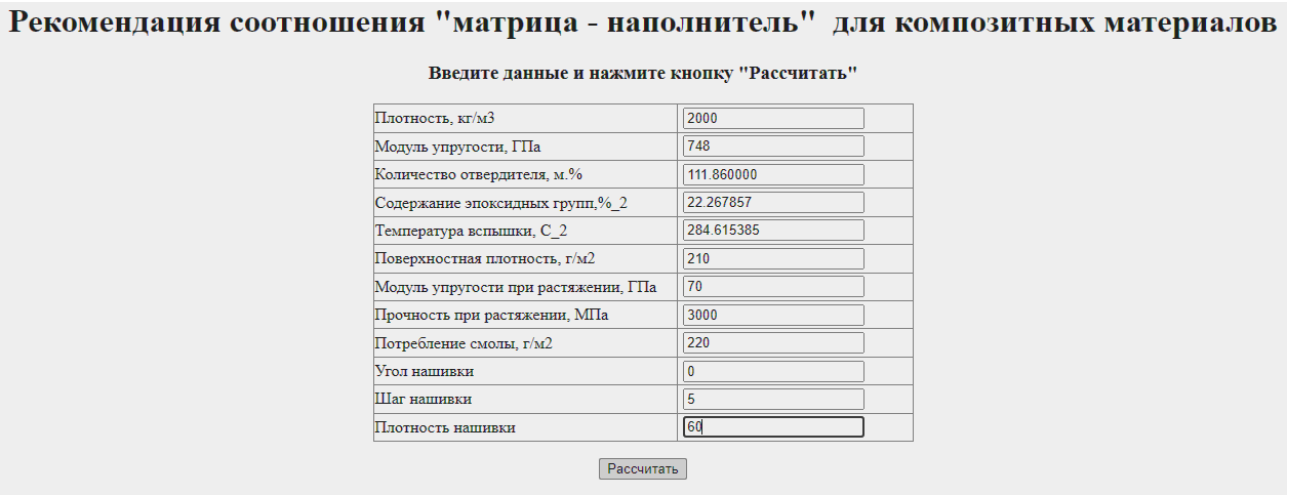

Рисунок 47. Приложение Flask.

На открывшейся html-странице (рисунок 47) необходимо ввести данные для прогноза соотношения «матрица-наполнитель» и нажать кнопку «Рассчитать». Результат прогноза представлен на рисунке 48.

Прогнозное значение соотношения "матрица - наполнитель": 2.914088010787964

Рисунок 48. Результат прогноза с помощью приложения Flask.

Второй варианта приложения – консольное. В этом случае данные для расчета вводятся непосредственно в рабочем файле (или отдельном для удобства использования) с расширением .ipynb (рисунок 49). Здесь же выводится и результат работы приложения (рисунок 50).

```
In [8]: # Создадим функцию для ввода данных
        def input_variable():
            x1 = float(input('Плотность, кг/м3: '))
            x2 = float(input('Модуль упругости, ГПа: '))
            x3 = float(input('Количество отвердителя, м.%: '))
            x4 = float(input('Содержание эпоксидных групп,%_2: '))
            x5 = float(input('Температура вспышки, С_2: '))
            x6 = float(input('Поверхностная плотность, г/м2: '))
            x7 = float(input('Модуль упругости при растяжении, ГПа: '))
            x8 = float(input('Прочность при растяжении, МПа: '))
            x9 = float(input('Потребление смолы, г/м2: '))
            x10 = float(input('Угол нашивки: '))
            x11 = float(input('Шаг нашивки: '))
            x12 = float(input('Плотность нашивки: '))
            return x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12

In [*]: # Создадим функцию для вызова приложения
        def app_model():
            # Загружаем модель и масштабаторы
            nn_model = load_model('model_matrix')
            scaler_x = load('minmax_scl_x.pkl')
            scaler_y = load('minmax_scl_y.pkl')

            print('Приложение прогнозирует соотношение "матрица-наполнитель"')
            for i in range(110):
                try:
                    print('Введите "1" для прогноза, "2" для выхода')
                    check = input()

                    if check == '1':
                        print('Введите данные для прогноза')
                        X = input_variable()
                        X =  scaler_x.transform(np.array(X).reshape(1,-1))
                        prediction = nn_model.predict(X)
                        output = scaler_y.inverse_transform(prediction)
                        print('Прогнозное значение соотношения "матрица-наполнитель": ')
                        print(output[0][0])

                    elif check == '2':
                        break
                    else:
                        print('Повторите выбор')

                except Exception as e:
                    print(e)
                    print('Введены некорректные данные. Пожалуйста, повторите операцию')

        # Запускаем приложение
        app_model()
```

Рисунок 49. Консольное приложение для прогноза соотношения «матрица-наполнитель».

```
Приложение прогнозирует соотношение "матрица-наполнитель"
Введите "1" для прогноза, "2" для выхода
1
Введите данные для прогноза
Плотность, кг/м3: 2000
Модуль упругости, ГПа: 748
Количество отвердителя, м.%: 111.860000
Содержание эпоксидных групп,%_2: 22.267857
Температура вспышки, С_2: 284.615385
Поверхностная плотность, г/м2: 210
Модуль упругости при растяжении, ГПа: 70
Прочность при растяжении, МПа: 3000
Потребление смолы, г/м2: 220
Угол нашивки: 0
Шаг нашивки: 5
Плотность нашивки: 60
1/1 [==============================] - 0s 62ms/step
Прогнозное значение соотношения "матрица-наполнитель":
2.914088
Введите "1" для прогноза, "2" для выхода
2
```

Рисунок 50. Консольное приложение для прогнозирования соотношения «матрица-наполнитель».

2.6. Создание удаленного репозитория и загрузка результатов работы.

Репозиторий создан на сайте github.com по адресу: https://github.com/wwwmyroot/DS_graduate (рисунки 51-52).

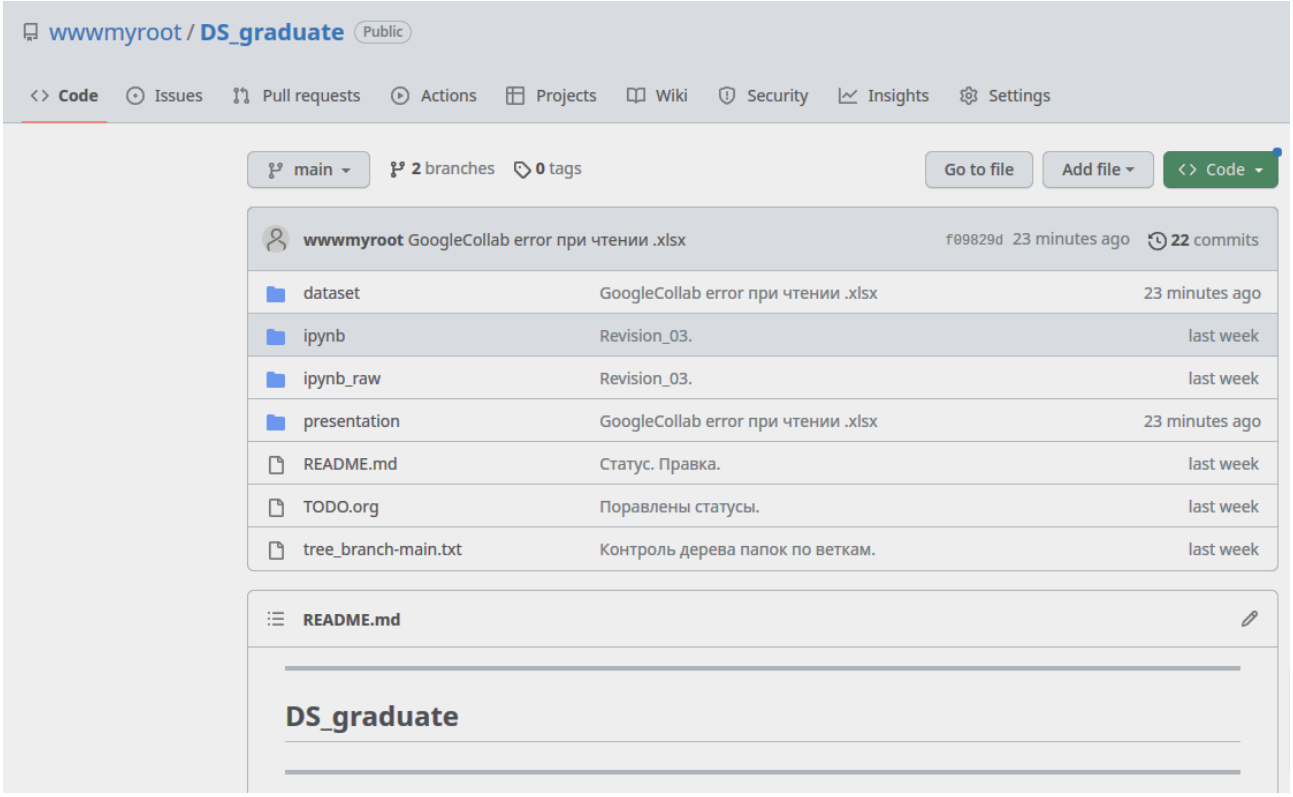

Рисунок 51. Репозиторий на сайте GitHub.com

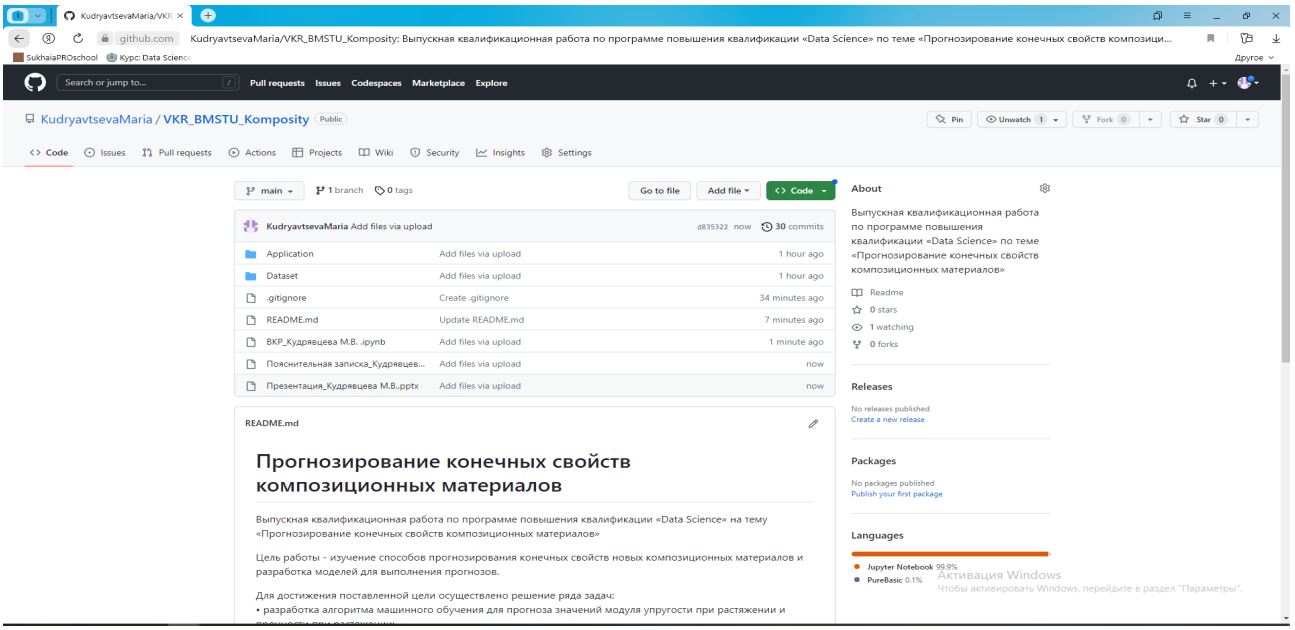

Рисунок 52. Содержание репозитория на сайте github.com.

Заключение

В результате выполнения выпускной квалификационной работы, цель которой – изучение способов прогнозирования конечных свойств новых композиционных материалов, были проанализированы характеристики композитных материалов, а также разработаны модели машинного обучения для выполнения прогнозов этих характеристик.

С использованием разработанных алгоритмов была проведена обработка экспериментальных данных модуля упругости при растяжении, прочности при растяжении и соотношения «матрица-наполнитель» с использованием языка программирования python.

Как показал анализ исходных данных, корреляционная зависимость между характеристиками композитов крайне слабая и стремится к нулю. Этот факт непосредственно повлиял на результат работы регрессионных моделей. Все использованные модели показали низкую прогнозирующую способность. Лучшим алгоритмом для прогноза модуля упругости при растяжении выбран AdaBoostRegressor, для прогнозирования прочности при растяжении – XGBRegressor.

Созданная для рекомендации соотношения «матрица-наполнитель» нейронная сеть также плохо справилась с поставленной задачей прогноза. Такие низкие показатели работы алгоритмов машинного обучения говорят о том, что прогнозирование свойств композиционных материалов – достаточно сложный процесс, требующий как знаний в области композиционных материалов, так и опыта в построении и использовании алгоритмов машинного обучения.

Полученный неудовлетворительный результат может также свидетельствовать о недостатках и ошибках в наборе исходных данных, недостаточно глубокой и детальной обработке данных, неточностях в выборе алгоритмов машинного обучения и их параметров.

Таким образом, для успешного решения задачи, поставленной в выпускной квалификационной работе, необходимы более глубокие знания в области материаловедения и технологии конструкционных материалов, математического анализа и статистики, а также в области решения задач машинного обучения и обработки данных. Более детальное изучение данных вопросов и консультация квалифицированных специалистов из указанных областей определенно положительно повлияют на уточнение подходов и оптимизацию алгоритмов для решения задачи прогнозирования конечных свойств композиционных материалов.

Список литературы

1 Абдуллин И. А., Автоматизированная информационная система прогнозирования свойств полимерных композиционных материалов на основе регрессионного анализа [Текст] / И. А. Абдуллин, А. Ф. Гумеров, Л. Н. Шафигуллин // Вестник Казанского технологического университета. – 2012. Режим доступа: https://cyberleninka.ru/article/n/avtomatizirovannaya-informatsionnaya-sistema-prognozirovaniya-svoystv-polimernyh-kompozitsionnyh-materialov-na-osnove. (дата обращения: 04.04.2023).

2 Бондалетова Л.И. Полимерные композиционные материалы (часть 1) [Текст] : учебное пособие / Л.И. Бондалетова, В.Г. Бондалетов. – Томск: Изд-во Томского политехнического университета, 2013. – 118 с.

3 Боршова И. Развитие композитов [Текст] / И. Боршова // Нефть и Жизнь. – 2022. – № 2. – с. 6–9.

4 Касенова Т. Композиты готовы к захвату рынков. Режим доступа: https://vmeste.severstal.com/expert/kompozity-gotovy-k-zakhvatu-rynkov/. (дата обращения: 04.04.2023).

5 Кербер М.Л. Полимерные композиционные материалы: структура, свойства, технология [Текст] : учеб. пособие / Кербер М.Л., Виноградов В.М., Головкин Г.С. и др.; под ред. А.А. Берлина. – СПб.: Профессия, 2008. – 560 с., ил.

6 Леонов В.В., Материаловедение и технология композиционных материалов [Текст] : Курс лекций / В.В. Леонов, О.А. Артемьева, Е.Д. Кравцова. – Красноярск: Федеральное государственное образовательное учреждение высшего профессионального образования «СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ», 2007. – 241 с.

7 Соловьев В.Г. Композиционные материалы в строительстве [Электронный ресурс] : учебное пособие для обучающихся по направлению

подготовки 08.03.01 Строительство / В.Г. Соловьев, В.Ф. Коровяков, О.А. Ларсен, Н.А. Гальцева; Министерство науки и высшего образования Российской Федерации, Национальный исследовательский Московский государственный строительный университет, кафедра технологии вяжущих веществ и бетонов. – Электрон. дан. и прогр. (3,8 Мб). – Москва: Издательство МИСИ – МГСУ, 2020. – Режим доступа: http://lib.mgsu.ru/. — Загл. с титул. экрана.

8 2.3. Разведочный анализ данных (рад) [Электронный ресурс] : – Режим доступа: https://studfile.net/preview/8858767/page:8/ (дата обращения: 02.04.2023).

9 4.2 Регуляризация, частные наименьшие квадраты и kNN-регрессия [Электронный ресурс] : – Режим доступа: https://ranalytics.github.io/data-mining/042-Regularization.html (дата обращения: 07.04.2023).

10 5 алгоритмов регрессии в машинном обучении, о которых вам следует знать [Электронный ресурс] : – Режим доступа: https://habr.com/ru/company/vk/blog/513842/ (дата обращения: 02.04.2023).

11 Apoorva, D. Регрессия (Regression) [Электронный ресурс] : – Режим доступа: https://www.helenkapatsa.ru/rieghriessiia/ (дата обращения: 07.04.2023).

12 CatBoost, XGBoost и выразительная способность решающих деревьев [Электронный ресурс] : – Режим доступа: https://habr.com/ru/company/ods/blog/645887/#3.3 (дата обращения: 04.04.2023).

13 Cheat code to find(MSE,RMSE,MAE)Mape [Электронный ресурс] : – Режим доступа: https://www.kaggle.com/code/udayreddie/cheat-code-to-find-mse-rmse-mae-mape (дата обращения: 11.04.2023).

14 Detect and Remove the Outliers using Python [Электронный ресурс] : – Режим доступа: https://www.geeksforgeeks.org/detect-and-remove-the-outliers-using-python/ (дата обращения: 07.04.2023).

15 Keras - последовательная модель Sequential [Электронный ресурс] : – Режим доступа: https://proproprogs.ru/tensorflow/keras-posledovatelnaya-model-sequential (дата обращения: 03.04.2023).

16 Maszański, A. Машинное обучение для начинающих: алгоритм случайного леса (Random Forest) [Электронный ресурс] : – Режим доступа: https://proglib.io/p/mashinnoe-obuchenie-dlya-nachinayushchih-algoritm-sluchaynogo-lesa-random-forest-2021-08-12 (дата обращения: 08.04.2023).

17 Maszański, A. Метод k-ближайших соседей (k-nearest neighbour) [Электронный ресурс] : – Режим доступа: https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19 (дата обращения: 09.04.2023).

18 SciKeras 0.9.0 documentation. Migrating from tf.keras.wrappers.scikit_learn [Электронный ресурс] : – Режим доступа: https://www.adriangb.com/scikeras/stable/migration.html (дата обращения: 11.04.2023).

19 Абрамов, И. Масштабирование характеристик в машинном обучении [Электронный ресурс] : – Режим доступа: https://procodings.ru/dev-ru/masshtabirovanie-harakteristik-v-mashinnom-obuchenii/ (дата обращения: 12.04.2023).

20 Алгоритм AdaBoost [Электронный ресурс] : – Режим доступа: https://habr.com/ru/company/otus/blog/503888/ (дата обращения: 05.04.2023).

21 Алгоритм классификации Random Forest на Python [Электронный ресурс] : – Режим доступа: https://pythonru.com/uroki/sklearn-random-forest (дата обращения: 12.04.2023).

22 Вятский государственный университет, Корреляционный анализ MySQL [Электронный ресурс] : – Режим доступа:

https://e.vyatsu.ru/pluginfile.php/462616/mod_resource/content/3/Теоретический%20материал_корреляционный%20анализ.pdf (дата обращения: 05.04.2023).

23 Гришкина, Т.Е. Корреляционный анализ [Электронный ресурс] : – Режим доступа: https://irbis.amursu.ru/DigitalLibrary/AmurSU_Edition/11630.pdf (дата обращения: 06.04.2023).

24 Документация по библиотеке scikit-learn: 1.4.2. Регрессия [Электронный ресурс] : – Режим доступа: https://scikit-learn.ru/1-4-support-vector-machines/#regression (дата обращения: 14.04.2023).

25 Документация по библиотеке scikit-learn: 1.5. Стохастический градиентный спуск [Электронный ресурс] : – Режим доступа: https://scikit-learn.ru/1-5-stochastic-gradient-descent/ (дата обращения: 12.04.2023).

26 Документация по библиотеке scikit-learn: 1.10. Деревья решений [Электронный ресурс] : – Режим доступа: https://scikit-learn.ru/1-10-decision-trees/ (дата обращения: 12.04.2023).

27 Ильина, Т. 9 ключевых алгоритмов машинного обучения простым языком [Электронный ресурс] : – Режим доступа: https://habr.com/ru/post/509472/ (дата обращения: 02.04.2023).

28 Интеграция машинного обучения в веб-приложения с помощью Flask [Электронный ресурс] : – Режим доступа: https://digitrain.ru/articles/193378/ (дата обращения: 14.04.2023).

29 Коротеев М., Диагностика систем машинного обучения [Электронный ресурс] : – Режим доступа: https://koroteev.site/text/ml4/?ysclid=lfgw3x6jgu88803263#метрики-эффективности-для-регрессии (дата обращения: 11.04.2023).

30 Корреляционный анализ. Академия НАФИ. Москва [Электронный ресурс] : – Режим доступа: https://nafi.ru/upload/spss/Lection_6.pdf (дата обращения: 04.04.2023).

31 Масштабирование, стандартизация или нормализация с помощью Scikit-Learn [Электронный ресурс] : – Режим доступа: https://datascience.xyz/theory/masshtabirovanie-standartizaciya-ili-normalizaciya-s-pomoshhju-scikit-learn.html (дата обращения: 09.04.2023).

32 Масштабирование функций с помощью Scikit-Learn для науки о данных [Электронный ресурс] : – Режим доступа: https://digitrain.ru/articles/80191/ (дата обращения: 09.04.2023).

33 Машинное обучение с Python в розничной торговле : метрики качества в задачах регрессии [Электронный ресурс] : – Режим доступа: https://statrpy2020.blogspot.com/2022/03/python_20.html (дата обращения: 11.04.2023).

34 Модели для классификации: Метод опорных векторов (SVM) [Электронный ресурс] : – Режим доступа: https://python-school.ru/blog/svm_classifier/ (дата обращения: 14.04.2023).

35 Подбор коэффициентов линейной регрессии методом наименьших квадратов (Ordinary Least Squares) [Электронный ресурс] : – Режим доступа: https://python-school.ru/blog/linearregression-ols/ (дата обращения: 08.04.2023).

36 Проверка на нормальность [Электронный ресурс] : – Режим доступа: http://datascientist.one/proverka-na-normalnost/ (дата обращения: 08.04.2023).

37 Редакция Кодкампа, Как выполнить тест Шапиро-Уилка в Python [Электронный ресурс] : – Режим доступа: https://www.codecamp.ru/blog/shapiro-wilk-test-python/ (дата обращения: 08.04.2023).

38 Редакция Кодкампа, Как использовать графики QQ для проверки нормальности [Электронный ресурс] : – Режим доступа: https://www.codecamp.ru/blog/q-q-plot-normality/ (дата обращения: 10.04.2023).

39 Редакция Кодкампа, Как удалить выбросы в Python [Электронный ресурс] : – Режим доступа: https://www.codecamp.ru/blog/remove-outliers-python/ (дата обращения: 02.04.2023).

40 Редакция Кодкампа, Корреляции в Стате: Пирсон, Спирмен и Кендалл [Электронный ресурс] : – Режим доступа: https://www.codecamp.ru/blog/correlations-stata/ (дата обращения: 04.04.2023).

41 Рябенко, Е. Прикладной статистический анализ данных. 6. Анализ зависимостей. [Электронный ресурс] : – Режим доступа: http://www.machinelearning.ru/wiki/images/e/e7/Psad_corr.pdf (дата обращения: 05.04.2023).

42 Способы обнаружения и удаления выбросов [Электронный ресурс] : – Режим доступа: https://questu.ru/articles/434373/ (дата обращения: 07.04.2023).

43 Сравнительное изучение алгоритмов классического машинного обучения [Электронный ресурс] : – Режим доступа: https://machinelearningmastery.ru/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222/ (дата обращения: 04.04.2023).

44 Что такое нейронная сеть? [Электронный ресурс] : – Режим доступа: https://aws.amazon.com/ru/what-is/neural-network/ (дата обращения: 09.04.2023).