

**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»

Тема: «Прогнозирование конечных свойств композиционных материалов»

Слушатель

Харин Сергей Семенович

Москва, 2023

Содержание

Введение.....	3
1. Аналитическая часть.....	5
1.1. Постановка задачи.....	5
1.2. Описание используемых методов	6
1.3. Разведочный анализ данных.....	10
2. Практическая часть.....	18
2.1. Предобработка данных.....	18
2.2. Разработка и обучение моделей	22
2.3. Нейронная сеть для рекомендации соотношения матрица – наполнитель ...	35
2.4. Параллельная разработка приложения.....	43
2.5. Создание репозитория на GitHub.....	48
Заключение.....	49
Список литературы.....	50

Введение

Композитный материал - это многокомпонентный материал, изготовленный (искусственно или естественным путем) из двух или более компонентов со значительно отличающимися физическими и/или химическими свойствами, которые объединяются для получения нового материала со свойствами, отличными от свойств отдельных компонентов, а не просто наложенными друг на друга. В композитных материалах принято различать матрицу/матрицу и наполнитель/наполнитель, последний выполняет армирующую функцию (по аналогии с армированием в композитных строительных материалах, таких как железобетон). Наполнителем в композитных материалах обычно являются углеродные или стеклянные волокна, а матрицей - полимер. Комбинация различных компонентов улучшает свойства материала, делая его одновременно легким и прочным. В то же время отдельные компоненты остаются нетронутыми в структуре композита, что отличает его от смесей и затвердевших строительных растворов. Изменяя состав, соотношение и ориентацию матрицы и наполнителя, можно получать различные материалы с требуемыми свойствами. Многие композитные материалы превосходят по своим механическим свойствам обычные материалы и сплавы. Использование композитных материалов обычно позволяет снизить вес конструкции при сохранении или улучшении ее механических свойств.

Разработка композитных материалов имеет очень высокий уровень сложности из-за огромного количества компонентов и условий обработки, так что изменение одного элемента может иметь множество непредвиденных последствий из-за их взаимодействия. Традиционные подходы к разработке новых материалов - это метод проб и ошибок, который приводит к проблемам низкой эффективности, высокой стоимости

и неустойчивым результатам. С другой стороны, многочисленные эксперименты и вычислительные испытания накопили огромное количество многомерных, сложных и плохо изученных данных, которые могут скрывать важные правила структура-свойства. Машинное обучение может помочь выявить такие взаимосвязи и обеспечить возможность тестирования и оптимизации нескольких метрик одновременно, ускоряя открытие и оптимизацию дизайна материалов и прогнозируя композиции с желаемыми и уникальными свойствами.

1. Аналитическая часть

1.1. Постановка задачи

Наша задача заключается в том, чтобы разработать модели для прогнозирования модуля упругости при растяжении, прочности при растяжении и соотношения матрица-наполнитель. У нас есть два файла с наборами данных.

Цель нашей работы - продемонстрировать наши знания и навыки в области Data Science, машинного обучения, баз данных, нейросетей и больших данных, которые мы получили в процессе обучения на квалификационном курсе. Для достижения этой цели мы должны выполнить следующие задачи:

- Провести разведочный анализ предложенных данных и предварительно обработать данные.
- Обучить несколько моделей для прогнозирования модуля упругости при растяжении и прочности при растяжении. Оценить точность моделей на тренировочных и тестовых наборах данных.
- Написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель. Оценить точность прогноза на тренировочном и тестовом наборах данных.
- Разработать приложение с интерфейсом командной строки, которое будет предоставлять один или два прогноза.

Для дальнейшей работы два файла с датасетами были объединены в один. Полученный датасет содержит 1023 строки и 13 столбцов (рисунок 1). Это означает, что часть данных (а именно 17 строк из датасета

dataset_nup) была удалена из таблицы и исключена из дальнейшего исследования.

Объединение датасетов по индексу, тип объединения INNER

```
[3] # объединяем датасеты
df = pd.merge(X_bo, X_nup, how = 'inner')
df.drop(['Unnamed: 0'], axis=1, inplace=True)
# 13 столбцов и 1023 строки
print('Размер датасета: {}'.format(df.shape))
df.head()
```

Размер датасета: (1023, 13)

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп, %_2	Температура вспышки, C_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

Рисунок 1. Объединенный датасет для работы

1.2. Описание используемых методов

Всего в работе используется 7 моделей машинного обучения :

- Метод К-ближайших соседей
- Линейная регрессия
- Случайный лес
- Метод опорных векторов
- Градиентный бустинг
- Дерево решений
- Lasso регрессия.

Метод К-ближайших соседей заключается в том, что мы можем классифицировать образ, определив к какому классу он принадлежит на основе ближайшего к нему соседа. Для этого мы определяем k ближайших соседей для образа x и затем присваиваем ему класс, к которому относится наибольшее число образов из этой группы. Преимущества этого алгоритма заключаются в том, что он прост и понятен, легко обучается на новых данных, может работать с любым количеством категорий в задачах классификации, принимает два параметра - k и метрику расстояния (как

правило, это евклидово расстояние) и имеет низкую чувствительность к выбросам. Однако, его минусы состоят в том, что он имеет высокую стоимость вычислений, так как вам нужно обработать весь объем данных, и не работает так хорошо с категорическими параметрами.

Линейная регрессия является алгоритмом обучения с учителем в машинном обучении, который требует указания как входных, так и заранее подготовленных выходных данных для обучения модели. Эти данные вместе называются обучающей выборкой. Преимуществом линейной регрессии является ее простота в реализации. Однако, следует учитывать, что выбросы могут сильно повлиять на результаты регрессии.

Случайный лес - это алгоритм классификации, который состоит из многих деревьев решений (ансамбль решающих деревьев) и использует бэггинг и случайность признаков при построении каждого дерева, чтобы создать некоррелированный лес, прогноз которого точнее, чем у отдельного дерева. Преимущества этого метода в том, что он имеет высокую точность предсказания и практически не чувствителен к выбросам, масштабированию значений признаков и монотонным преобразованиям, не требует тщательной настройки параметров и хорошо работает «из коробки». Недостатки заключаются в том, что результаты случайного леса сложнее интерпретировать, алгоритм работает хуже многих линейных методов в случае большого количества разреженных признаков, склонен к переобучению на некоторых задачах, особенно на зашумленных данных, и предвзят в пользу признаков с большим количеством уровней, что может привести к неправильным предсказаниям на данных, содержащих категориальные переменные с различным количеством уровней.

Метод опорных векторов применяется для задач классификации и регрессии, а не только для регрессии, как было указано в вашем сообщении.

SVM ищет оптимальную гиперплоскость в p -мерном пространстве, которая максимально разделяет два класса объектов, или предсказывает целевую переменную в регрессии. SVM работает путем поиска оптимальной разделяющей гиперплоскости, которая максимизирует зазор между двумя классами или уменьшает ошибку предсказания целевой переменной в регрессии. SVM имеет ряд гиперпараметров, которые могут быть настроены для достижения оптимальной производительности. Основные преимущества SVM включают в себя высокую точность предсказания, возможность работать с высокоразмерными данными, возможность настройки гиперпараметров для достижения оптимальной производительности и хорошую обобщающую способность. Однако, недостатки SVM заключаются в его чувствительности к выбросам и шуму в данных, а также в высокой вычислительной сложности для больших наборов данных.

Градиентный бустинг - это метод машинного обучения, основанный на последовательном построении нескольких базовых классификаторов, каждый из которых компенсирует ошибки предыдущего, чтобы получить лучший финальный классификатор. Он имеет ряд преимуществ, таких как обучение на ошибках, выбор наблюдений на основе ошибки, простая настройка и легкая интерпретация. Однако, чтобы избежать переобучения, необходимо тщательно выбирать критерии остановки, и наблюдения с наибольшей ошибкой могут появляться чаще. Кроме того, он менее гибок, чем нейронные сети.

Метод **дерева решений** использует древовидную структуру для принятия решений на основе выбранных критериев. Алгоритм начинается с корневого узла, который представляет всю выборку, и на каждом шаге выбирает лучший признак для разделения выборки на подмножества.

Процесс продолжается рекурсивно для каждого подмножества, пока не будет достигнут критерий остановки, например, когда все объекты в подмножестве относятся к одному классу или достигнуто максимальное число уровней дерева. Метод дерева решений имеет ряд преимуществ, таких как легкая интерпретируемость, возможность обработки данных различных типов и работа с отсутствующими данными. Кроме того, он может использоваться как для задач классификации, так и для задач регрессии. Однако метод дерева решений также имеет некоторые недостатки. Например, при построении больших деревьев может возникнуть переобучение, когда модель хорошо подстраивается под обучающую выборку, но плохо обобщается на новые данные. Кроме того, он может быть неустойчив к малым изменениям в данных, что может привести к значительным изменениям в структуре дерева.


Метод **Lasso регрессия** - это метод регуляризации, используемый для выбора наиболее значимых признаков в модели линейной регрессии. Он основан на добавлении штрафа L1-нормы к функции потерь модели, что приводит к уменьшению коэффициентов признаков до нуля, тем самым выбирая только наиболее значимые признаки. Для обучения модели Lasso регрессии используется метод наименьших квадратов, при этом функция потерь модели выглядит как сумма квадратов разностей между предсказанными значениями и истинными значениями целевой переменной, умноженных на коэффициент регуляризации α , умноженный на L1-норму вектора весов модели. Таким образом, функция потерь Lasso регрессии выглядит следующим образом: $\text{Loss} = \text{Sum of squared differences between predicted and true values} + \alpha * \text{L1-norm of weights}$. Преимущества метода Lasso регрессии включают возможность выбора наиболее значимых признаков в модели, что может привести к более простой и интерпретируемой модели. Кроме того, Lasso регрессия

может использоваться для снижения эффекта мультиколлинеарности между признаками, что может улучшить качество модели. Однако, метод Lasso регрессии также имеет некоторые недостатки. Например, если признаки взаимосвязаны, то Lasso регрессия может выбрать только один из них, что может привести к потере информации и снижению качества модели. Кроме того, подбор оптимального значения параметра α может быть сложным и требовать значительного времени и ресурсов.

1.3. Разведочный анализ данных

Разведочный анализ данных является необходимой процедурой для получения представления о распределении данных и оценки качества данных, включая пропуски и выбросы. В ходе разведочного анализа учитываются и сравниваются множество признаков и закономерностей. Без понимания характера данных и их качества трудно работать с ними, поэтому важно изучить набор данных перед удалением или изменением каких-либо значений. Загрузка данных и вывод некоторых статистик позволяют получить необходимое представление о данных.

▼ Анализ и предобработка датасета

✓ 0.1  df.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1023 entries, 0 to 1022
Data columns (total 13 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Соотношение матрица-наполнитель          1023 non-null   float64
1   Плотность, кг/м3                          1023 non-null   float64
2   модуль упругости, ГПа                     1023 non-null   float64
3   Количество отвердителя, м.%               1023 non-null   float64
4   Содержание эпоксидных групп, %_2         1023 non-null   float64
5   Температура вспышки, C_2                  1023 non-null   float64
6   Поверхностная плотность, г/м2             1023 non-null   float64
7   Модуль упругости при растяжении, ГПа      1023 non-null   float64
8   Прочность при растяжении, МПа             1023 non-null   float64
9   Потребление смолы, г/м2                   1023 non-null   float64
10  Угол нашивки, град                         1023 non-null   int64
11  Шаг нашивки                               1023 non-null   float64
12  Плотность нашивки                         1023 non-null   float64
dtypes: float64(12), int64(1)
memory usage: 111.9 KB
```

Рисунок 2. Информация о столбцах и значениях датасета

Действительно, в разведочном анализе применяется множество методов для оценки характеристик датасета, включая гистограммы распределения, диаграммы ящика с усами, попарные графики рассеяния точек, тепловую карту, а также анализ и удаление выбросов, пропусков и дубликатов. Эти методы помогают понять, как данные распределены, выявить потенциальные проблемы в данных, такие как отсутствие значений, необычные выбросы или ошибки в данных. После проведения разведочного анализа можно сделать более информированные выводы о данных и выбрать соответствующие методы анализа и моделирования для достижения нужных целей.

Поиск пропусков в датасетах

```
[5] df.isnull().sum()
```

Соотношение матрица-наполнитель	0
Плотность, кг/м3	0
модуль упругости, ГПа	0
Количество отвердителя, м.%	0
Содержание эпоксидных групп,%_2	0
Температура вспышки, C_2	0
Поверхностная плотность, г/м2	0
Модуль упругости при растяжении, ГПа	0
Прочность при растяжении, МПа	0
Потребление смолы, г/м2	0
Угол нашивки, град	0
Шаг нашивки	0
Плотность нашивки	0
dtype: int64	

Рисунок 3. Поиск пропусков в датасете

Поиск дубликатов

```
[7] df.duplicated().sum()
```

0

Рисунок 4. Поиск дубликатов в датасете

Поиск уникальных значений

```
df.nunique()

Соотношение матрица-наполнитель      1014
Плотность, кг/м3                      1013
модуль упругости, ГПа                 1020
Количество отвердителя, м.%           1005
Содержание эпоксидных групп,%_2      1004
Температура вспышки, С_2              1003
Поверхностная плотность, г/м2        1004
Модуль упругости при растяжении, ГПа  1004
Прочность при растяжении, МПа        1004
Потребление смолы, г/м2              1003
Угол нашивки, град                    2
Шаг нашивки                          989
Плотность нашивки                     988
dtype: int64
```

Рисунок 5. Поиск уникальных значений в датасете

Просмотр числовых статистик датасета

```
[9] 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, град	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	44.252199	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	45.015793	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	90.000000	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	90.000000	14.440522	103.988901

Рисунок 6. Описательная статистика датасета

```
[ ] df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Рисунок 7. Описательная статистика датасета (2)

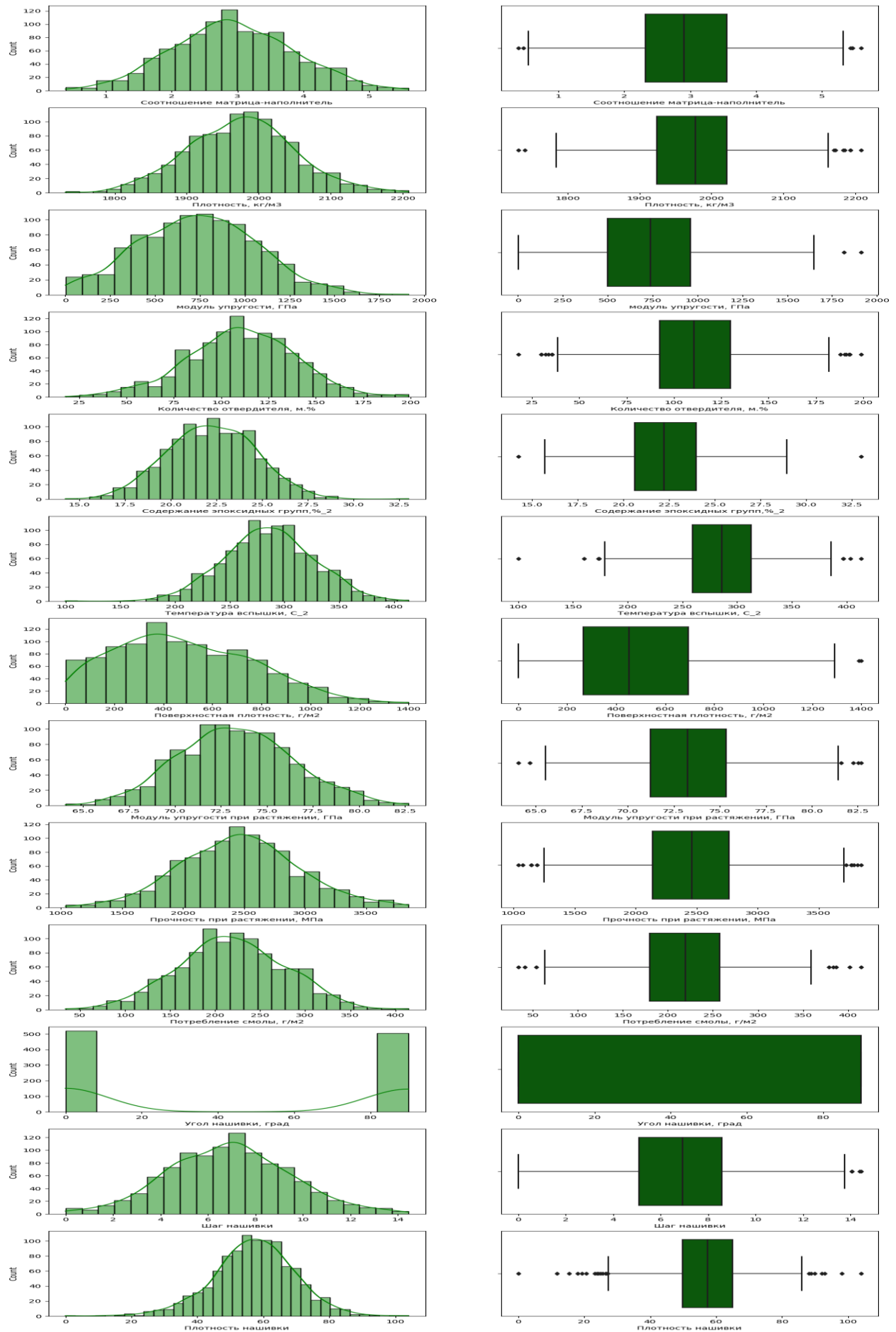


Рисунок 8. Гистограммы и ящики с усами

Гистограмма является удобным инструментом визуализации, который используется для представления распределения одной или нескольких переменных. Она строится путем разбиения значений на интервалы и подсчета количества наблюдений, попавших в каждый интервал. Гистограмма позволяет оценить различные характеристики распределения, такие как центральную тенденцию, дисперсию, форму и симметрию. Она также может использоваться для оценки нормальности эмпирического распределения. Кроме того, на гистограмму можно накладывать кривую распределения, чтобы визуально сравнить эмпирическое распределение с теоретическим распределением. Это помогает оценить, насколько хорошо данные соответствуют определенному теоретическому распределению и использовать эту информацию для выбора соответствующей модели.

Исследование графиков показало, что большинство параметров в датасете имеют нормальное распределение, за исключением «Угла нашивки» и «Поверхностной плотности, г/м²». Поскольку значения «Угла нашивки» ограничены двумя значениями (0 и 1), он считается бинарным и категориальным признаком.

"Ящик с усами" - это диаграмма, используемая в статистическом анализе для компактного представления одномерного распределения вероятностей. Она показывает медиану, нижний и верхний квартили, минимальное и максимальное значения и выбросы.

Выбросы - это точки данных, значительно отличающиеся от общей выборки и могут вызывать проблемы при статистическом анализе. На диаграмме "ящик с усами" выбросы выделяются как точки, находящиеся за

границами нижнего и верхнего усов. В данном случае, выбросы были обнаружены во всех параметрах, кроме "Угла нашивки".

Выбросы могут исказить данные и повлиять на статистические характеристики модели, такие как среднее значение и дисперсия. Они также могут привести к потере точности модели. Поэтому необходимо обнаружить и удалить выбросы перед моделированием данных.

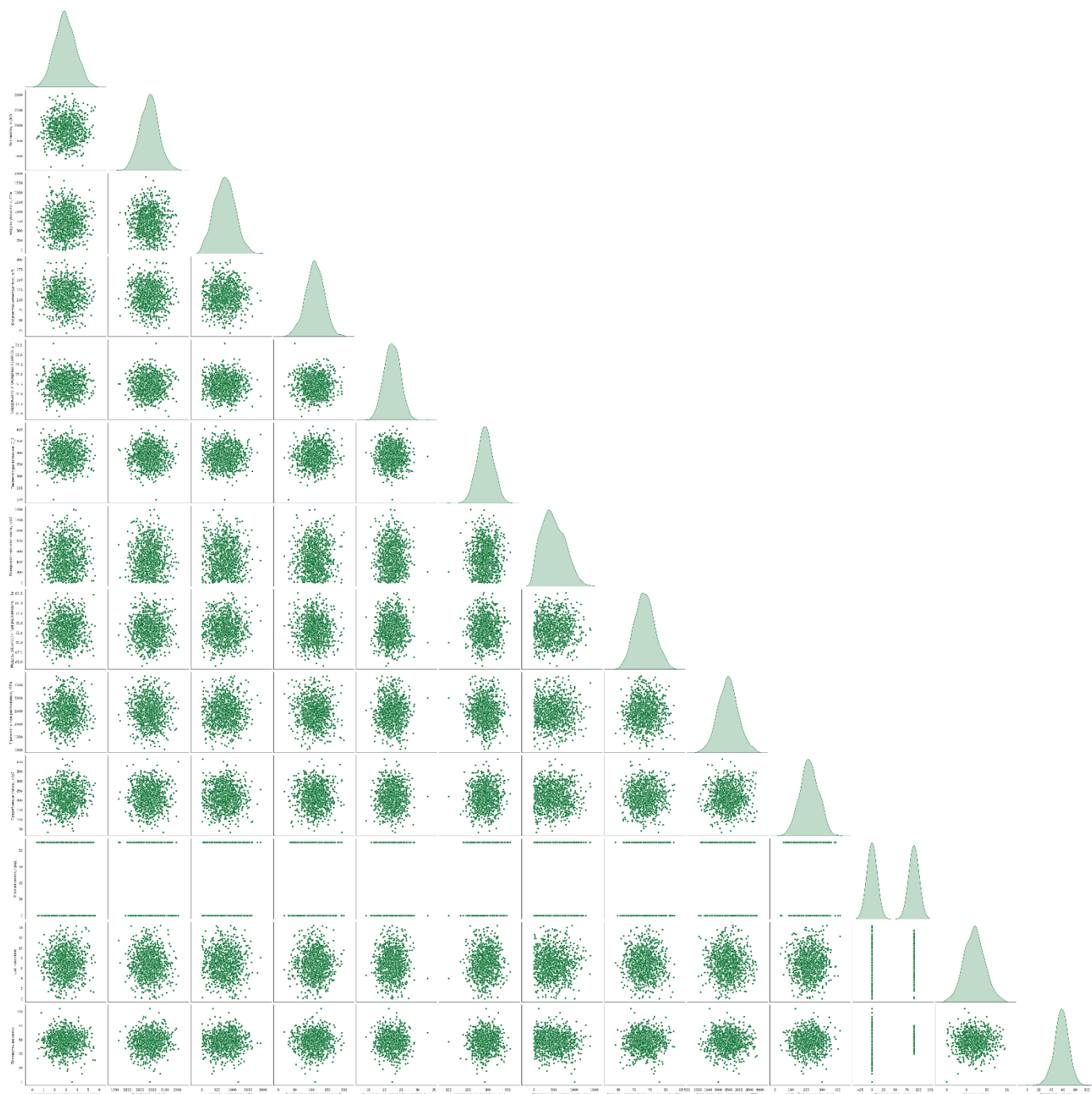


Рисунок 9. Гистограммы и ящики с усами

Согласно представленным графикам, не обнаружено линейной зависимости между характеристиками композитных материалов. Однако, необходимо провести анализ корреляций между переменными. Корреляционная связь подразумевает согласованные изменения между двумя или более признаками, где изменение одной переменной может вызвать закономерное изменение другой(-их) переменной(-ых).

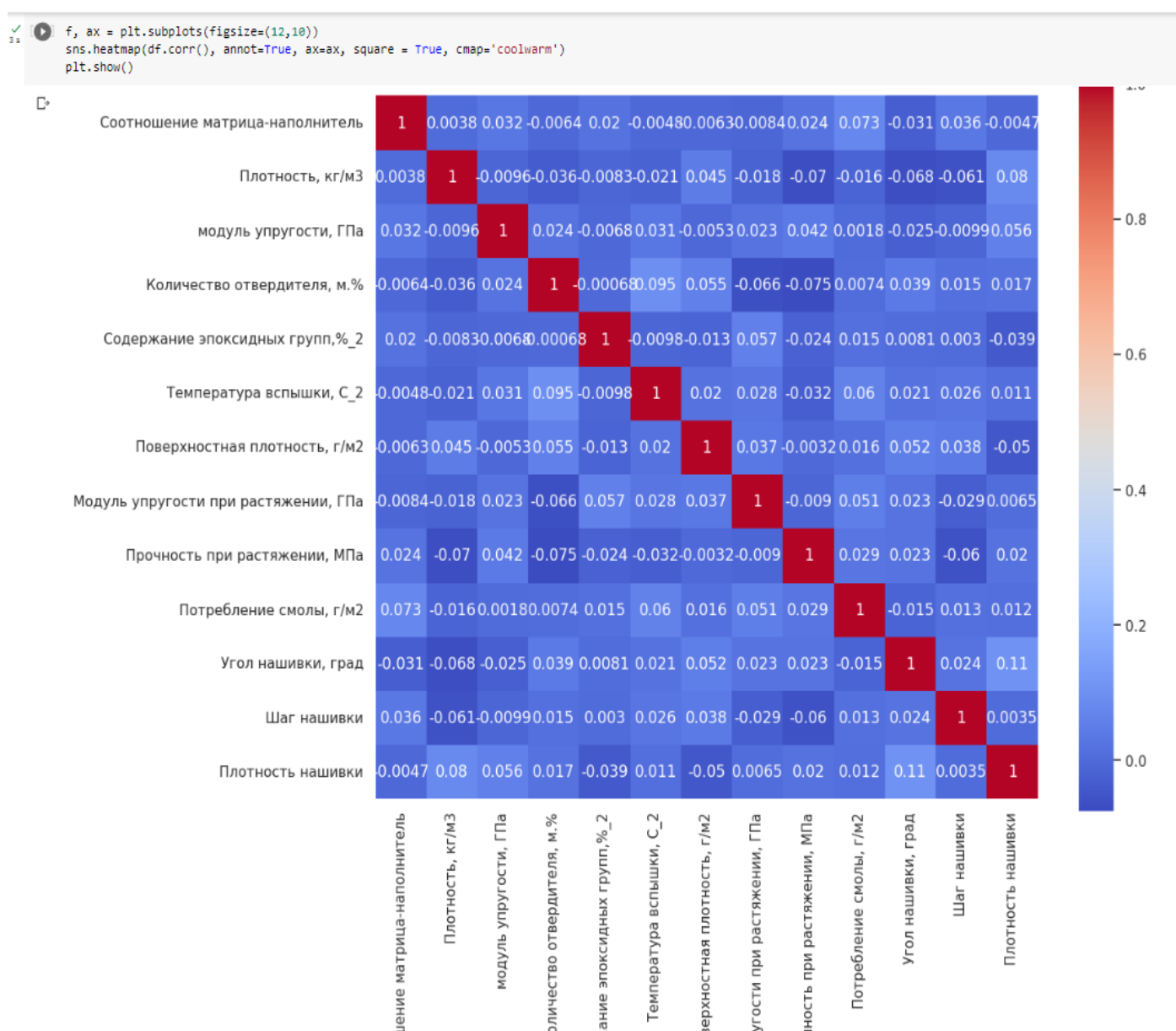


Рисунок 10. Тепловая карта

2. Практическая часть

2.1. Предобработка данных

Первым шагом предварительной обработкой рассмотрим удаление всех выбросов из данных до моделирования.

Удаление выбросов

```
✓ [13] df_clean = df.copy()
      for col in df_clean.columns:
          q75,q25 = np.percentile(df_clean.loc[:,col],[75,25])
          intr_qr = q75-q25

          max = q75+(1.5*intr_qr)
          min = q25-(1.5*intr_qr)

          df_clean.loc[df_clean[col] < min,col] = np.nan
          df_clean.loc[df_clean[col] > max,col] = np.nan
```

```
✓ [14] df_clean.isnull().sum()
0 2
```

Соотношение матрица-наполнитель	6
Плотность, кг/м3	9
модуль упругости, ГПа	2
Количество отвердителя, м.%	14
Содержание эпоксидных групп,%_2	2
Температура вспышки, С_2	8
Поверхностная плотность, г/м2	2
Модуль упругости при растяжении, ГПа	6
Прочность при растяжении, МПа	11
Потребление смолы, г/м2	8
Угол нашивки, град	0
Шаг нашивки	4
Плотность нашивки	21
dtype: int64	

Рисунок 11. Удаление выбросов

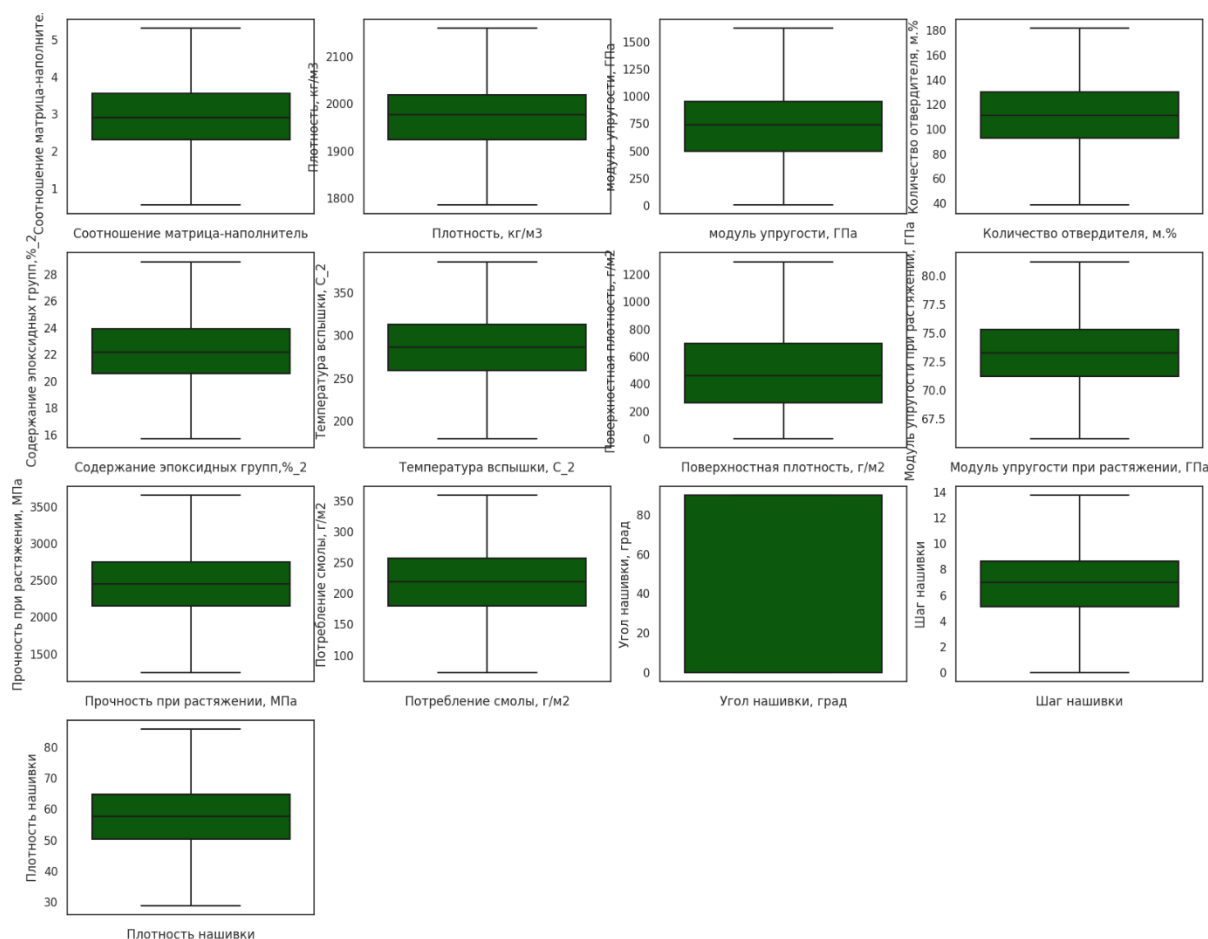


Рисунок 12. Чистый датасет после трёх чисток

Тест Шапиро-Уилка позволяет определить, является ли распределение выборки нормальным. Если значение p -уровня значимости (обычно 0,05) ниже указанного порога, то можно утверждать, что данные выборки не имеют нормальное распределение. В данном случае, значение p -уровня значимости оказалось меньше 0,05 для следующих характеристик: модуль упругости в ГПа, поверхностная плотность в г/м², модуль упругости при растяжении в ГПа, потребление смолы в г/м² и угол нашивки. Это говорит о том, что распределение для данных характеристик не является нормальным.

Проверка на нормальность (тест Шапиро-Уилка)

```
from scipy.stats import shapiro
for col in df_clean_3.columns:
    print(df_clean_3[col].name, shapiro(df_clean_3[col]))
```

Соотношение матрица-наполнитель ShapiroResult(statistic=0.9971880316734314, pvalue=0.10904344916343689)
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=0.23541033267974854)
Температура вспышки, C_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=0.009416559711098671)
Прочность при растяжении, МПа ShapiroResult(statistic=0.9973730444908142, pvalue=0.14375117421150208)
Потребление смолы, г/м2 ShapiroResult(statistic=0.9955164790153503, pvalue=0.008584197610616684)
Угол нашивки, град 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 13. Тест Шапиро-Уилка

Нормализация - это процедура предварительной обработки входных данных (например, обучающих, тестовых, валидационных выборок или реальных данных), в которой значения признаков во входном векторе приводятся к определенному заданному диапазону, такому как $[0...1]$ или $[-1...1]$. Главная цель нормализации - приведение данных, измеренных в разных единицах и диапазонах, к единому виду, который позволяет сравнивать данные и использовать их для расчета схожести объектов в выборке. Перед обучением модели все признаки должны быть приведены к одинаковому влиянию друг на друга. В библиотеке Scikit-learn на Python для этого доступны классы `MinMaxScaler` и `RobustScaler`.

Стандартизация представляет собой процесс приведения данных к определенному формату и представлению, что обеспечивает возможность их правильного использования в многомерном анализе. Она позволяет исключить влияние отклонений по каждому из признаков и привести все значения в датасете к нормальному распределению с математическим ожиданием, равным нулю, и стандартным отклонением, равным единице. Это приводит к созданию стандартизированной шкалы, которая определяет

положение каждого значения в наборе данных и измеряет его отклонение от среднего в единицах стандартного отклонения.

Функция MinMaxScaler была использована для форматирования данных, что позволило привести все параметры к одинаковому относительному масштабу. При этом сохранены относительные различия между значениями каждого объекта. Значения каждого признака были масштабированы таким образом, что максимальное значение признака стало равным 1, а минимальное - 0.

Нормализация данных

1) Нормализация с помощью MinMaxScaler

```
[ ] min_max_scaler = MinMaxScaler()
df_norm_minmax = pd.DataFrame(min_max_scaler.fit_transform(df_clean_3), columns = df_clean_3.columns, index=df_clean_3.index)
```

```
[ ] df_norm_minmax.describe()
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп, %_2	Температура вспышки, C_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
count	922.000000	922.000000	922.000000	922.000000	922.000000	922.000000	922.000000	922.000000	922.000000	922.000000	922.000000	922.000000	922.000000
mean	0.499412	0.502904	0.451341	0.506200	0.490578	0.516739	0.373295	0.487343	0.503776	0.507876	0.510846	0.503426	0.503938
std	0.187858	0.188395	0.201534	0.186876	0.180548	0.190721	0.217269	0.196386	0.188668	0.199418	0.500154	0.183587	0.193933
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.371909	0.368184	0.305188	0.378514	0.366571	0.386228	0.204335	0.353512	0.373447	0.374647	0.000000	0.372844	0.376869
50%	0.495189	0.511396	0.451377	0.506332	0.488852	0.516931	0.354161	0.483718	0.501481	0.510143	1.000000	0.506414	0.504310
75%	0.629774	0.624719	0.587193	0.638735	0.623046	0.646553	0.538397	0.617568	0.624299	0.642511	1.000000	0.626112	0.630842
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Рисунок 14. Описание датасета после преобразования с MinMaxScaler

Визуализируем полученный результат

```
[ ] plt.figure(figsize=(15, 6))
sns.set(context='notebook', style='whitegrid')
sns.kdeplot(data=df_norm_minmax)
plt.show()
```

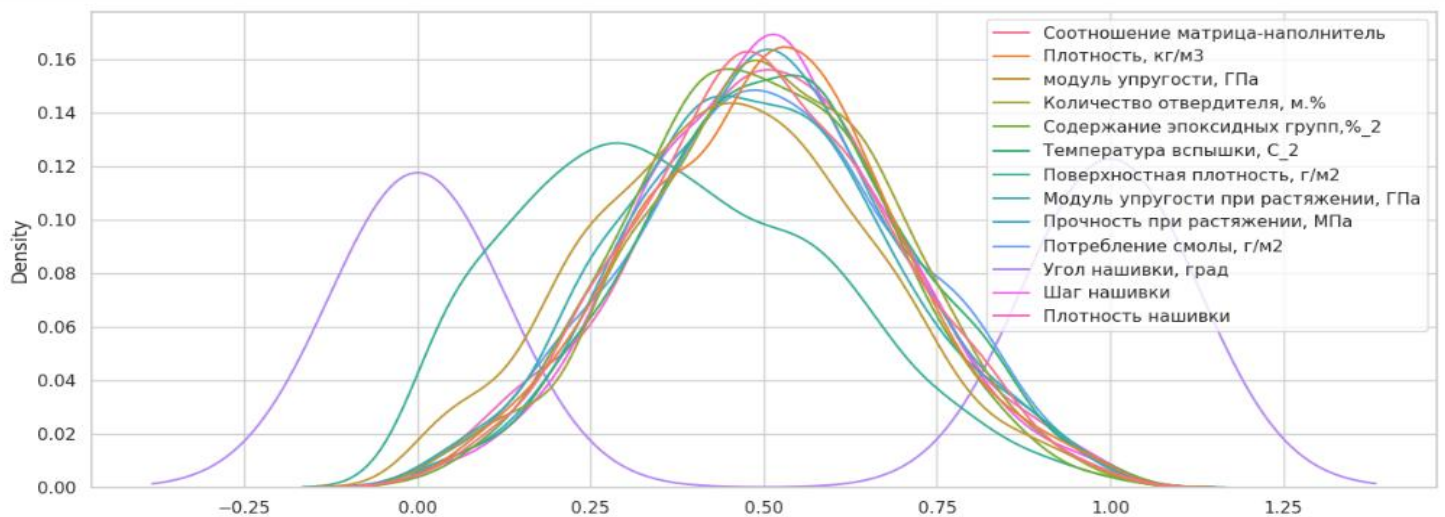


Рисунок 15. Визуализация после преобразования с MinMaxScaler

2.2. Разработка и обучение моделей

Для создания модели машинного обучения, которая может прогнозировать характеристики композитных материалов, такие как модуль упругости при растяжении и прочность при растяжении, необходимо выполнить следующие этапы:

1. Нормализовать данные, чтобы гарантировать их однородность и сопоставимость.

2. Разделить нормализованные данные на обучающую и тестовую выборки, где 30% данных отводится для тестирования модели, а на оставшихся данных модель обучается.

Для моделей используем датасет, преобразованный с помощью MinMaxScaler.

Разбиваем на тестовую, тренировочную выборки

```
[ ] #1) Выборка для прогноза Прочность при растяжении, МПа
X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(
    df_norm.loc[:, df_norm.columns != 'Прочность при растяжении, МПа'],
    df_norm[['Прочность при растяжении, МПа']],
    test_size = 0.3,
    random_state = 42)

#2) Выборка для прогноза Модуль упругости при растяжении, ГПа
X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(
    df_norm.loc[:, df_norm.columns != 'Модуль упругости при растяжении, ГПа'],
    df_norm[['Модуль упругости при растяжении, ГПа']],
    test_size = 0.3,
    random_state = 42)
```

Рисунок 16. Разбивка на тестовую и обучающую выборки

- Метод К-ближайших соседей :

1) K-nearest neighbors

```
[ ] # 1) Прочность при растяжении, МПа
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train_1, y_train_1)
y_pred_knn = knn.predict(X_test_1)
mae_knr = mean_absolute_error(y_pred_knn, y_test_1)
mse_knn_elast = mean_squared_error(y_test_1, y_pred_knn)
Score_KNN_1 = knn.score(X_test_1, y_test_1)

# 2) Модуль упругости при растяжении, ГПа
knn2 = KNeighborsRegressor(n_neighbors=5)
knn2.fit(X_train_2, y_train_2)
y_pred_knn2 = knn2.predict(X_test_2)
mae_knr2 = mean_absolute_error(y_pred_knn2, y_test_2)
mse_knn_elast2 = mean_squared_error(y_test_2, y_pred_knn2)
Score_KNN_2 = knn2.score(X_test_2, y_test_2)
```

```
K-nearest neighbors train data results:
Train score: 0.23
K-nearest neighbors test data results:
MAE: 0.17
MAPE: 6526587660746.40
MSE: 0.04
RMSE: 0.21
Test score: -0.18
=====
K-nearest neighbors train data results:
Train score: 0.18
K-nearest neighbors test data results:
MAE: 0.18
MAPE: 0.68
MSE: 0.05
RMSE: 0.23
Test score: -0.22
```

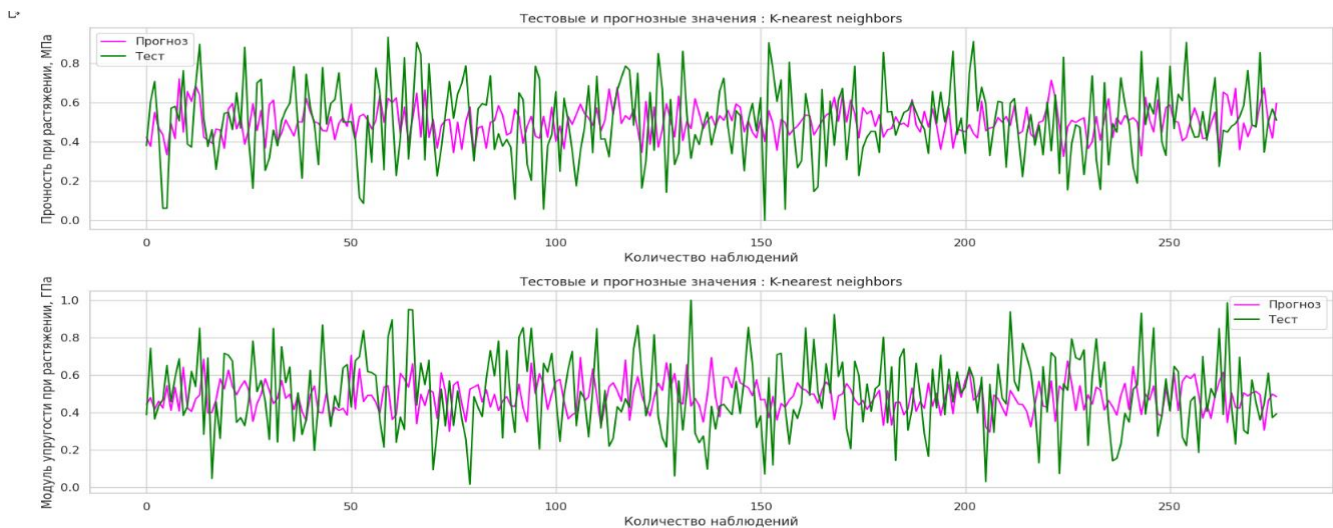


Рисунок 17. K-NN results

- Linear Regression

2) Линейная регрессия

```
[ ] # 1) Прочность при растяжении, МПа
lr = LinearRegression()
lr.fit(X_train_1, y_train_1)
y_pred_lr = lr.predict(X_test_1)
mae_lr = mean_absolute_error(y_pred_lr, y_test_1)
mse_lin_elast = mean_squared_error(y_test_1, y_pred_lr)
Score_LR_1 = lr.score(X_test_1, y_test_1)

# 2) Модуль упругости при растяжении, ГПа
lr2 = LinearRegression()
lr2.fit(X_train_2, y_train_2)
y_pred_lr2 = lr2.predict(X_test_2)
mae_lr2 = mean_absolute_error(y_pred_lr2, y_test_2)
mse_lin_elast2 = mean_squared_error(y_test_2, y_pred_lr2)
Score_LR_2 = lr2.score(X_test_2, y_test_2)
```

Linear Regression train data results:

Train score: 0.02

Linear Regression test data results:

MAE: 0.16

MAPE: 9232724774322.15

MSE: 0.04

RMSE: 0.19

Test score: -0.01

=====

Linear Regression train data results:

Train score: 0.02

Linear Regression test data results:

MAE: 0.17

MAPE: 0.65

MSE: 0.04

RMSE: 0.21

Test score: -0.03

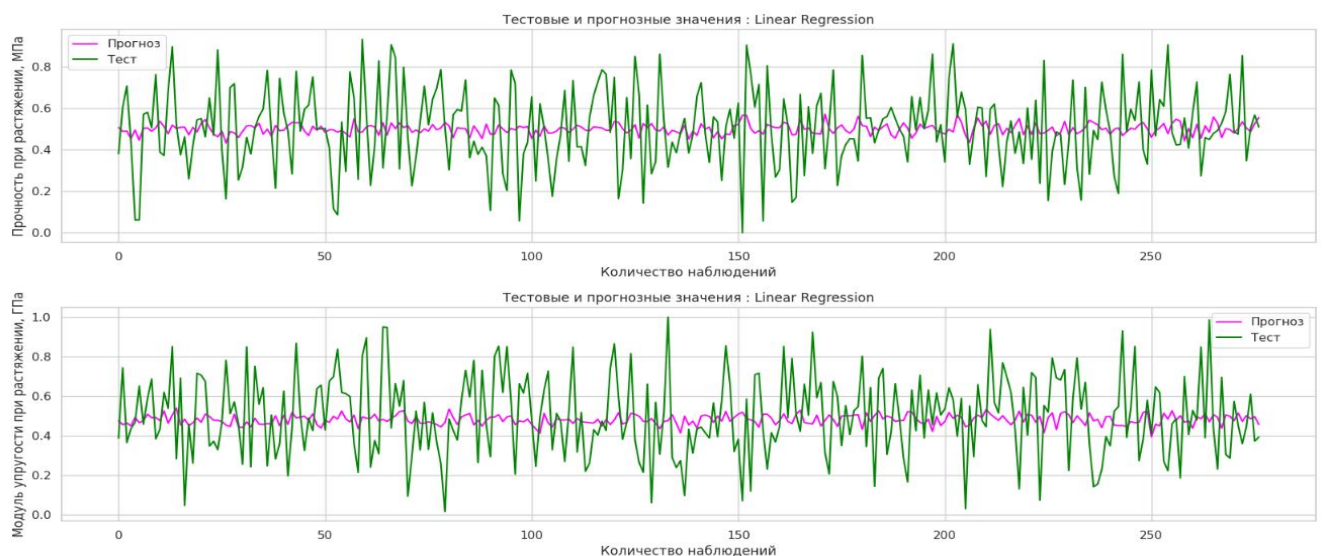


Рисунок 18. Linear Regression results

- Random Forest

3) Случайный лес

```
[ ] # 1) Прочность при растяжении, МПа
rfr = RandomForestRegressor(n_estimators=15,max_depth=7, random_state=33)
rfr.fit(X_train_1, y_train_1.values)
y_pred_forest = rfr.predict(X_test_1)
mae_rfr = mean_absolute_error(y_pred_forest, y_test_1)
mse_rfr_elast = mean_squared_error(y_test_1,y_pred_forest)
Score_RF_1 = rfr.score(X_test_1, y_test_1)

# 2) Модуль упругости при растяжении, ГПа
rfr2 = RandomForestRegressor(n_estimators = 15,max_depth = 7, random_state = 33)
rfr2.fit(X_train_2, y_train_2.values)
y2_pred_forest = rfr2.predict(X_test_2)
mae_rfr2 = mean_absolute_error(y2_pred_forest, y_test_2)
mse_rfr_elast2 = mean_squared_error(y_test_2, y2_pred_forest)
Score_RF_2 = rfr2.score(X_test_2, y_test_2)
```

Random Forest train data results:

Train score: 0.48

Random Forest test data results:

MAE: 0.16

MAPE: 8769553567468.01

MSE: 0.04

RMSE: 0.20

Test score: -0.09

=====

Random Forest train data results:

Train score: 0.39

Random Forest test data results:

MAE: 0.17

MAPE: 0.67

MSE: 0.04

RMSE: 0.21

Test score: -0.06

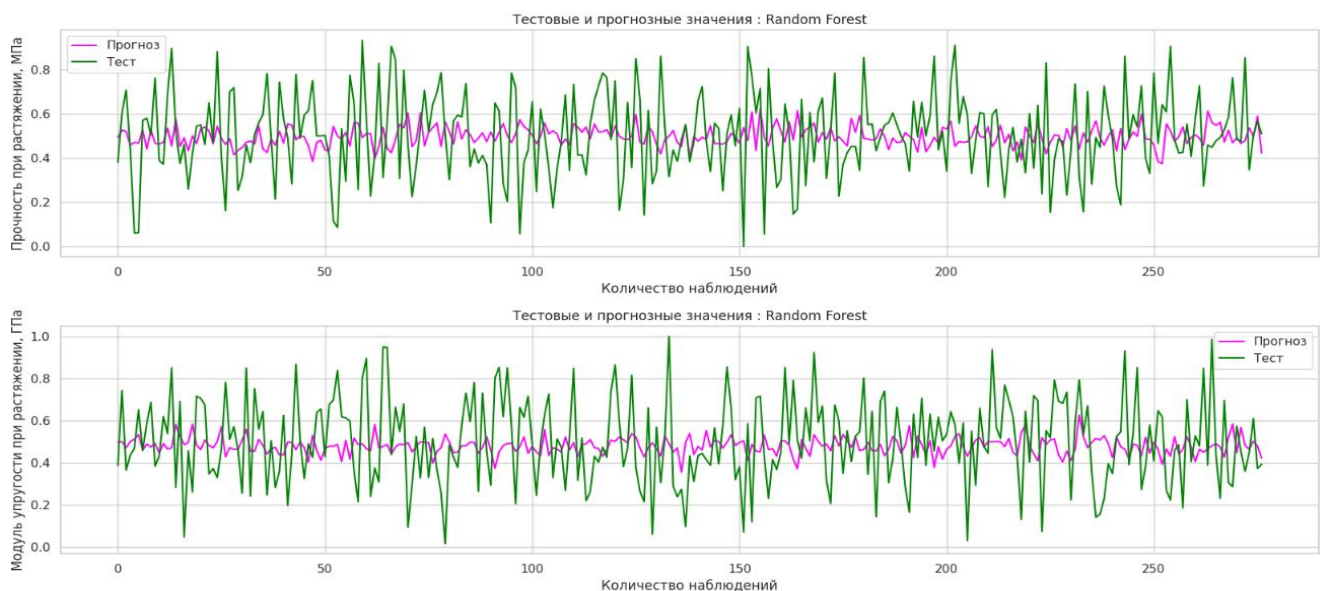


Рисунок 19. Random Forest results

- Support Vector

4) Support Vector

```
[ ] # 1) Прочность при растяжении, МПа
svr = make_pipeline(StandardScaler(), SVR(kernel = 'rbf', C = 500.0, epsilon = 1.0))
#обучаем модель
svr.fit(X_train_1, np.ravel(y_train_1))
#вычисляем коэффициент детерминации
y_pred_svr=svr.predict(X_test_1)
mae_svr = mean_absolute_error(y_pred_svr, y_test_1)
v = mean_squared_error(y_test_1,y_pred_svr)
score_SVR_1 = svr.score(X_test_1, y_test_1)
mse_svr_elast = mean_squared_error(y_test_1, y_pred_svr)

# 2) Модуль упругости при растяжении, ГПа
svr2 = make_pipeline(StandardScaler(), SVR(kernel = 'rbf', C = 500.0, epsilon = 1.0))
#обучаем модель
svr2.fit(X_train_2, np.ravel(y_train_2))
#вычисляем коэффициент детерминации
y_pred_svr2 = svr2.predict(X_test_2)
mae_svr2 = mean_absolute_error(y_pred_svr2, y_test_2)
mse_svr_elast2 = mean_squared_error(y_test_2, y_pred_svr2)
score_SVR_2 = svr2.score(X_test_2, y_test_2)
```

Support Vector train data results:

Train score: -0.00

Support Vector test data results:

MAE: 0.15

MAPE: 8176304369191.76

MSE: 0.04

RMSE: 0.19

Test score: -0.00

=====

Support Vector train data results:

Train score: -0.01

Support Vector test data results:

MAE: 0.17

MAPE: 0.68

MSE: 0.04

RMSE: 0.20

Test score: -0.00

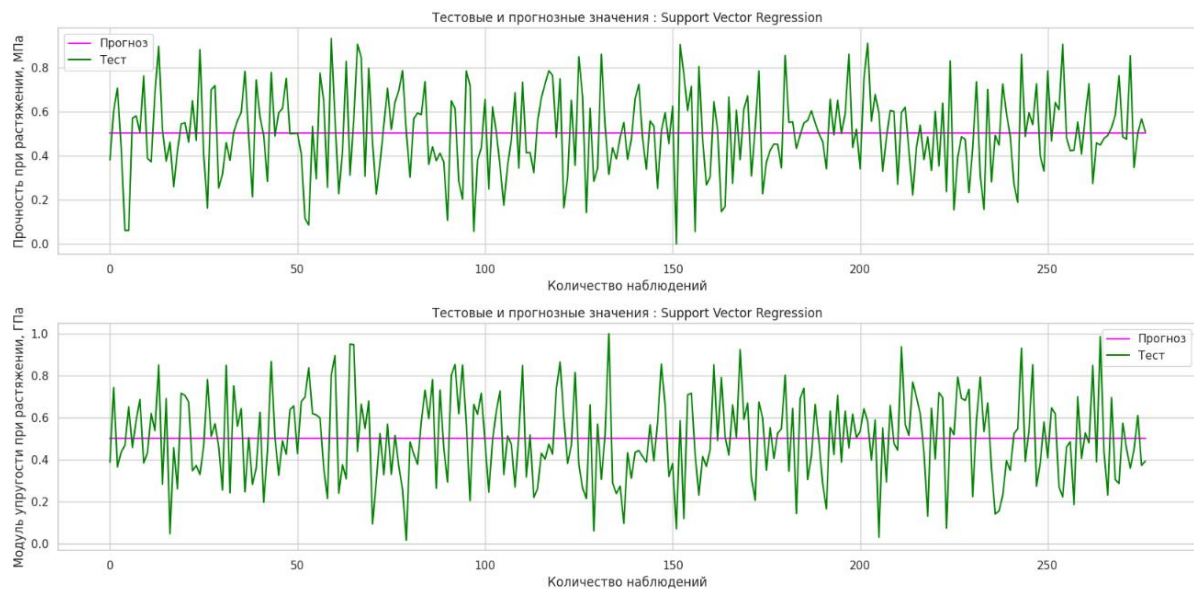


Рисунок 20. Support Vector results

- Gradient Boosting

5) Gradient Boosting

```
[ ] # 1) Прочность при растяжении, МПа
gbr = make_pipeline(StandardScaler(), GradientBoostingRegressor())
gbr.fit(X_train_1, np.ravel(y_train_1))
y_pred_gbr = gbr.predict(X_test_1)
mae_gbr = mean_absolute_error(y_pred_gbr, y_test_1)
mse_gbr_elast = mean_squared_error(y_test_1, y_pred_gbr)
Score_GB_1 = gbr.score(X_test_1, y_test_1)

# 2) Модуль упругости при растяжении, ГПа
gbr2 = make_pipeline(StandardScaler(), GradientBoostingRegressor())
gbr2.fit(X_train_2, np.ravel(y_train_2))
y_pred_gbr2 = gbr2.predict(X_test_2)
mae_gbr2 = mean_absolute_error(y_pred_gbr2, y_test_2)
mse_gbr_elast2 = mean_squared_error(y_test_2, y_pred_gbr2)
Score_GB_2 = gbr2.score(X_test_2, y_test_2)
```

```
Gradient Boosting train data results:
Train score: 0.51
Gradient Boosting test data results:
MAE: 0.16
MAPE: 8395674120401.24
MSE: 0.04
RMSE: 0.20
Test score: -0.05
=====
Gradient Boosting train data results:
Train score: 0.49
Gradient Boosting test data results:
MAE: 0.17
MAPE: 0.68
MSE: 0.05
RMSE: 0.21
Test score: -0.09
```



Рисунок 21. Gradient Boosting results

- Decision Tree Regressor

6) Decision Tree Regressor

```
[ ] # 1) Прочность при растяжении, МПа
dtr = DecisionTreeRegressor()
dtr.fit(X_train_1, y_train_1.values)
y_pred_dtr = dtr.predict(X_test_1)
mae_dtr = mean_absolute_error(y_pred_dtr, y_test_1)
mse_dtr_elast = mean_squared_error(y_test_1, y_pred_dtr)
Score_DT_1 = dtr.score(X_test_1, y_test_1)

# 2) Модуль упругости при растяжении, ГПа
dtr2 = DecisionTreeRegressor()
dtr2.fit(X_train_2, y_train_2.values)
y_pred_dtr2 = dtr2.predict(X_test_2)
mae_dtr2 = mean_absolute_error(y_pred_dtr2, y_test_2)
mse_dtr_elast2 = mean_squared_error(y_test_2, y_pred_dtr2)
Score_DT_2 = dtr2.score(X_test_2, y_test_2)
```

Decision Tree train data results:

Train score: 1.00

Decision Tree test data results:

MAE: 0.2

MAPE: 8198168168923.65

MSE: 0.06

RMSE: 0.25

Test score: -0.75

=====

Decision Tree train data results:

Train score: 1.00

Decision Tree test data results:

MAE: 0.25

MAPE: 0.83

MSE: 0.09

RMSE: 0.30

Test score: -1.18

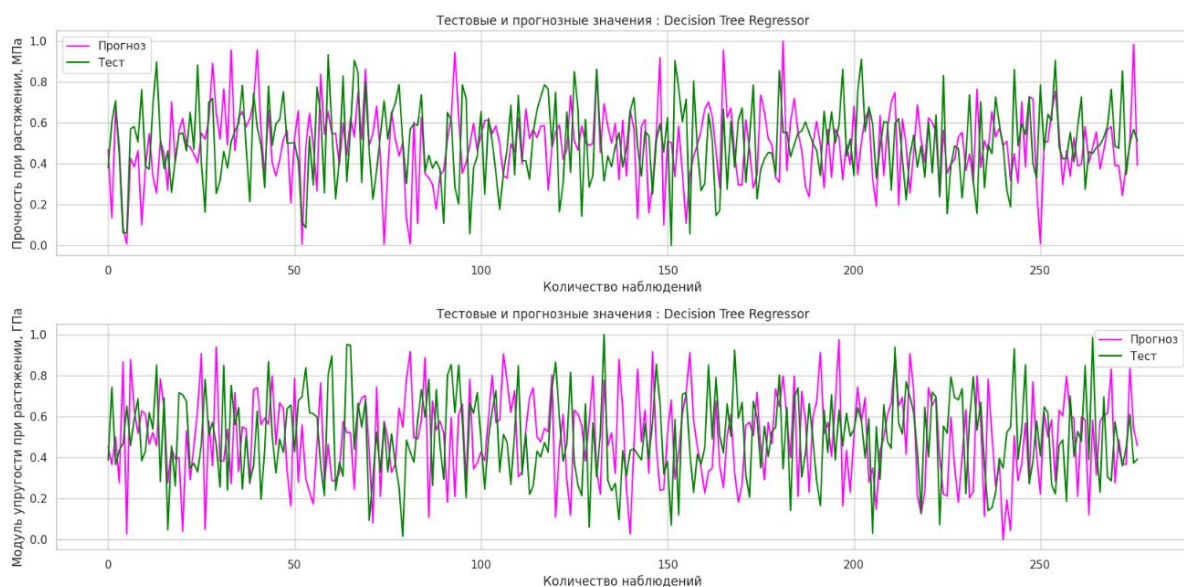


Рисунок 22. Decision Tree Regressor

- The Lasso

```
# 1) Прочность при растяжении, МПа
clf = linear_model.Lasso(alpha=0.1)
clf.fit(X_train_1, y_train_1)
y_pred_clf = clf.predict(X_test_1)
mae_clf = mean_absolute_error(y_pred_clf, y_test_1)
mse_clf_elast = mean_squared_error(y_test_1, y_pred_clf)
Score_LAS_1 = clf.score(X_test_1, y_test_1)

# 2) Модуль упругости при растяжении, ГПа
clf2 = linear_model.Lasso(alpha = 0.1)
clf2.fit(X_train_2, y_train_2)
y_pred_clf2 = clf2.predict(X_test_2)
mae_clf2 = mean_absolute_error(y_pred_clf2, y_test_2)
mse_clf_elast2 = mean_squared_error(y_test_2, y_pred_clf2)
Score_LAS_2 = clf2.score(X_test_2, y_test_2)
```

The Lasso train data results:

Train score: 0.00

The Lasso test data results:

MAE: 0.15

MAPE: 8183534682026.43

MSE: 0.04

RMSE: 0.19

Test score: -0.00

=====

The Lasso train data results:

Train score: 0.00

The Lasso test data results:

MAE: 0.17

MAPE: 0.65

MSE: 0.04

RMSE: 0.21

Test score: -0.01

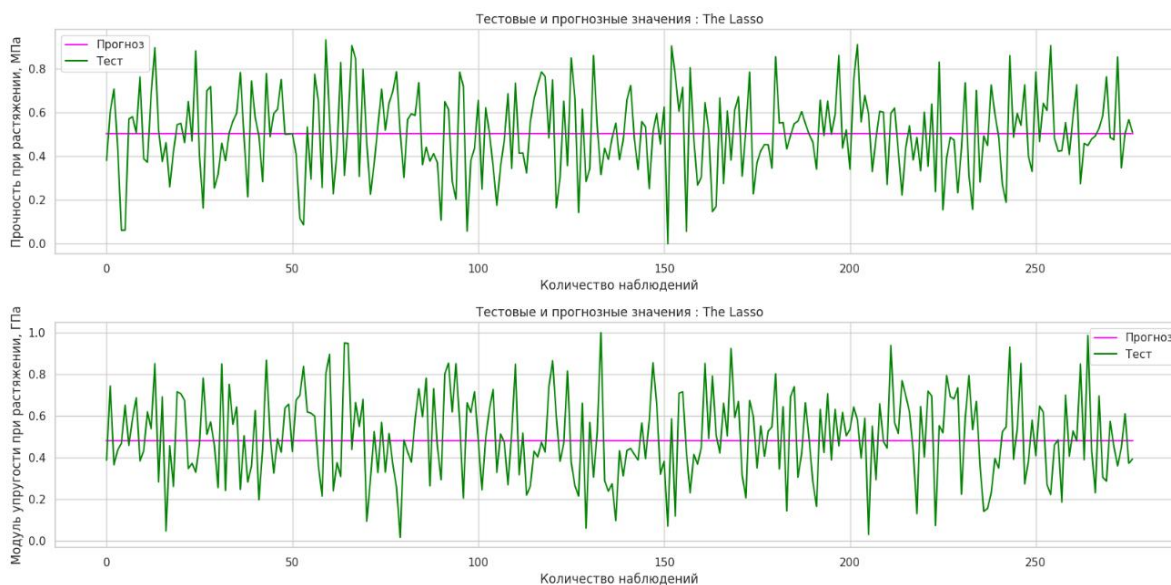


Рисунок 23. Lasso Regressor

Приведём полную таблицу с результатами всех моделей :

	target_var	model_name	MAE	MAPE	MSE	RMSE	R2
0	Модуль упругости при растяжении, ГПа	K-NN	0.17	6526587660746.40	0.04	0.21	-0.18
1	Прочность при растяжении, МПа	K-NN	0.18	0.68	0.05	0.23	-0.22
2	Модуль упругости при растяжении, ГПа	Linear_Regression	0.16	9232724774322.15	0.04	0.19	-0.01
3	Прочность при растяжении, МПа	Linear_Regression	0.17	0.65	0.04	0.21	-0.03
4	Модуль упругости при растяжении, ГПа	Random_Forest	0.16	8769553567468.01	0.04	0.20	-0.09
5	Прочность при растяжении, МПа	Random_Forest	0.17	0.67	0.04	0.21	-0.06
6	Модуль упругости при растяжении, ГПа	Support_Vector_Regression	0.15	8176304369191.76	0.04	0.19	-0.00
7	Прочность при растяжении, МПа	Support_Vector_Regression	0.17	0.68	0.04	0.20	-0.00
8	Модуль упругости при растяжении, ГПа	Gradient_Boosting	0.16	8395674120401.24	0.04	0.20	-0.04
9	Прочность при растяжении, МПа	Gradient_Boosting	0.17	0.68	0.05	0.21	-0.09
10	Модуль упругости при растяжении, ГПа	Decision Tree Regression	0.19	8198168168923.67	0.06	0.25	-0.72
11	Прочность при растяжении, МПа	Decision Tree Regression	0.24	0.83	0.09	0.30	-1.11
12	Модуль упругости при растяжении, ГПа	Lasso_Regressor	0.15	8183534682026.43	0.04	0.19	-0.00
13	Прочность при растяжении, МПа	Lasso_Regressor	0.17	0.65	0.04	0.21	-0.01

Если коэффициент детерминации (R^2) отрицателен, то это свидетельствует о низкой способности модели к обобщению. Такая модель работает хуже, чем простое вычисление среднего. Наилучшие результаты R^2 и MAE на тестовой выборке были получены с помощью алгоритма регрессии «SVR».

Все модели, используемые для прогнозирования модуля упругости при растяжении, показали неудовлетворительный результат. Аналогично, при разработке моделей для прогнозирования прочности при растяжении были получены модели с низкой обобщающей способностью, которые также не решают поставленную задачу.

Попробуем прогнать датасет через модели ещё раз, но теперь с данными нормализованными с помощью Normalizer.

Normalizer в машинном обучении - это метод нормализации данных, который преобразует каждую строку данных (т.е. объект) в единичную

норму (норму длины равной 1). Это означает, что каждый вектор признаков будет масштабирован таким образом, что его длина (норма) станет равной 1.

В простейшем случае, Normalizer может быть использован для нормализации каждой строки данных путем деления каждого значения признака на евклидову длину всего вектора признаков. Однако, Normalizer также может быть настроен для использования других норм, таких как Манхэттенская норма или норма Чебышева.

Преимущества использования Normalizer включают устранение влияния различных масштабов и единиц измерения признаков в данных, что может повысить качество работы модели. Кроме того, Normalizer может уменьшить влияние выбросов в данных, так как он фокусируется на относительных значениях каждого признака, а не на абсолютных значениях.

2) Нормализация данных с помощью Normalizer

```
[132] normalizer = Normalizer()
res = normalizer.fit_transform(df_clean_3)
df_norm_norm = pd.DataFrame(res, columns = df_clean_3.columns)
df_norm_norm
```

	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	0.000499	0.545436	0.198490	0.013434	0.006381	0.076473	0.056424	0.018808	0.806064	0.059111	0.000000	0.001075	0.016121
1	0.000499	0.545011	0.198335	0.034634	0.005705	0.080543	0.056380	0.018793	0.805435	0.059065	0.000000	0.001342	0.012618
2	0.000744	0.544829	0.202097	0.030022	0.005976	0.076388	0.056362	0.018787	0.805167	0.059046	0.000000	0.001342	0.015298
3	0.000746	0.539271	0.201687	0.030161	0.006004	0.076742	0.056623	0.018874	0.808906	0.059320	0.000000	0.001348	0.016178
4	0.000699	0.519919	0.219673	0.030449	0.006062	0.077475	0.057164	0.019055	0.816627	0.059886	0.000000	0.001361	0.019055
...
917	0.000700	0.601520	0.281289	0.026806	0.006201	0.100077	0.064463	0.022522	0.735625	0.038520	0.027733	0.002797	0.014489
918	0.001078	0.641541	0.139172	0.045683	0.006133	0.079552	0.109733	0.022819	0.738645	0.036842	0.028164	0.003306	0.016820
919	0.000953	0.572927	0.121081	0.032107	0.006959	0.072161	0.214894	0.021709	0.773510	0.068729	0.026143	0.001209	0.019645
920	0.001191	0.664389	0.238353	0.045454	0.006187	0.088652	0.206205	0.023802	0.665970	0.063368	0.028931	0.002029	0.018728
921	0.001071	0.531558	0.117343	0.036325	0.007726	0.084624	0.213349	0.020895	0.803159	0.054762	0.025307	0.001709	0.021773

922 rows x 13 columns

Рисунок 24. Нормализация данных с помощью Normalizer

	target_var	model_name	MAE	MAPE	MSE	RMSE	R2
0	Модуль упругости при растяжении, ГПа	K-NN	0.01	0.02	0.00	0.02	0.94
1	Прочность при растяжении, МПа	K-NN	0.00	0.05	0.00	0.00	0.72
2	Модуль упругости при растяжении, ГПа	Linear_Regression	0.01	0.02	0.00	0.01	0.96
3	Прочность при растяжении, МПа	Linear_Regression	0.00	0.04	0.00	0.00	0.79
4	Модуль упругости при растяжении, ГПа	Random_Forest	0.01	0.02	0.00	0.02	0.94
5	Прочность при растяжении, МПа	Random_Forest	0.00	0.04	0.00	0.00	0.78
6	Модуль упругости при растяжении, ГПа	Support_Vector_Regression	0.07	0.10	0.01	0.09	-0.45
7	Прочность при растяжении, МПа	Support_Vector_Regression	0.00	0.11	0.00	0.00	-0.17
8	Модуль упругости при растяжении, ГПа	Gradient_Boosting	0.01	0.01	0.00	0.01	0.98
9	Прочность при растяжении, МПа	Gradient_Boosting	0.00	0.04	0.00	0.00	0.78
10	Модуль упругости при растяжении, ГПа	Decision Tree Regression	0.01	0.02	0.00	0.02	0.93
11	Прочность при растяжении, МПа	Decision Tree Regression	0.00	0.06	0.00	0.00	0.54
12	Модуль упругости при растяжении, ГПа	Lasso_Regressor	0.06	0.08	0.01	0.07	-0.00
13	Прочность при растяжении, МПа	Lasso_Regressor	0.00	0.09	0.00	0.00	-0.00
14	Соотношение матрица/наполнитель	Model 1	0.92	0.39	1.32	1.15	-0.86
15	Соотношение матрица/наполнитель	Model 2	1.88	0.62	4.22	2.05	-4.95

Рисунок 24. Результаты моделей на данных, преобразованных с помощью Normalizer

Модели дали на удивление очень хороший результат. Наилучший результат по R2 наблюдается у Gradient Boosting решреессора – 0.92 и 0.78.

Продолжим эксперимент и прогоним датасет, преобразованный с помощью StandardScaler через модели.

	target_var	model_name	MAE	MAPE	MSE	RMSE	R2
0	Модуль упругости при растяжении, ГПа	K-NN	0.89	3.08	1.26	1.12	-0.21
1	Прочность при растяжении, МПа	K-NN	0.95	1.94	1.35	1.16	-0.25
2	Модуль упругости при растяжении, ГПа	Linear_Regression	0.82	2.66	1.05	1.02	-0.01
3	Прочность при растяжении, МПа	Linear_Regression	0.86	1.23	1.11	1.05	-0.03
4	Модуль упругости при растяжении, ГПа	Random_Forest	0.85	3.24	1.13	1.07	-0.09
5	Прочность при растяжении, МПа	Random_Forest	0.88	1.52	1.15	1.07	-0.06
6	Модуль упругости при растяжении, ГПа	Support_Vector_Regression	0.91	6.09	1.29	1.13	-0.24
7	Прочность при растяжении, МПа	Support_Vector_Regression	0.94	1.92	1.36	1.16	-0.25
8	Модуль упругости при растяжении, ГПа	Gradient_Boosting	0.85	4.01	1.09	1.05	-0.05
9	Прочность при растяжении, МПа	Gradient_Boosting	0.89	1.62	1.19	1.09	-0.10
10	Модуль упругости при растяжении, ГПа	Decision Tree Regression	1.11	5.37	1.96	1.40	-0.88
11	Прочность при растяжении, МПа	Decision Tree Regression	1.25	3.79	2.29	1.51	-1.11
12	Модуль упругости при растяжении, ГПа	Lasso_Regressor	0.81	1.03	1.04	1.02	-0.00
13	Прочность при растяжении, МПа	Lasso_Regressor	0.85	1.02	1.09	1.05	-0.01

Рисунок 25. Результаты моделей на данных, преобразованных с помощью StandardScaler

Результат снова неудовлетворительный, как и в случае с данными, нормализованными с помощью MinMaxScaler.

Проведем тот же опыт с необработанными данными, то есть, с изначальными данными без изменений и преобразований.

	target_var	model_name	MAE	MAPE	MSE	RMSE	R2
0	Модуль упругости при растяжении, ГПа	K-NN	413.89	0.18	272226.77	521.75	-0.22
1	Прочность при растяжении, МПа	K-NN	2.63	0.04	11.26	3.36	-0.13
2	Модуль упругости при растяжении, ГПа	Linear_Regression	379.79	0.17	222990.58	472.22	0.00
3	Прочность при растяжении, МПа	Linear_Regression	2.56	0.03	10.12	3.18	-0.02
4	Модуль упругости при растяжении, ГПа	Random_Forest	383.61	0.17	227244.36	476.70	-0.01
5	Прочность при растяжении, МПа	Random_Forest	2.64	0.04	10.77	3.28	-0.08
6	Модуль упругости при растяжении, ГПа	Support_Vector_Regression	399.80	0.17	249540.70	499.54	-0.11
7	Прочность при растяжении, МПа	Support_Vector_Regression	3.52	0.05	19.03	4.36	-0.92
8	Модуль упругости при растяжении, ГПа	Gradient_Boosting	394.80	0.17	239939.17	489.84	-0.07
9	Прочность при растяжении, МПа	Gradient_Boosting	2.65	0.04	10.98	3.31	-0.11
10	Модуль упругости при растяжении, ГПа	Decision Tree Regression	518.05	0.22	449589.80	670.51	-1.01
11	Прочность при растяжении, МПа	Decision Tree Regression	3.67	0.05	21.00	4.58	-1.11
12	Модуль упругости при растяжении, ГПа	Lasso_Regressor	379.77	0.17	222989.11	472.22	0.00
13	Прочность при растяжении, МПа	Lasso_Regressor	2.56	0.03	10.11	3.18	-0.02

Рисунок 26. Результаты моделей на данных без преобразований

Результаты на данных без преобразований неудовлетворительны. Делаем вывод, что лучше всех модели работают на данных, нормализованных с помощью Normalizer.

2.3. Нейронная сеть для рекомендации соотношения матрица – наполнитель

Нейронные сети - это метод искусственного интеллекта, который обучает компьютеры обрабатывать данные так же, как и человеческий мозг, используя процесс машинного обучения, известный как глубокое обучение. Он состоит из взаимосвязанных узлов или нейронов, организованных в слоистую структуру, которая напоминает человеческий мозг. Это позволяет создавать адаптивные системы, которые учатся на своих ошибках и постоянно улучшают свою производительность. Таким образом, искусственные нейронные сети стремятся решать сложные задачи с большей точностью.

Обучение нейронной сети - это такой процесс, при котором происходит подбор оптимальных параметров модели, с точки зрения минимизации функционала ошибки.

Всего в работе было обучено более 20 моделей, на презентацию в работе было выбрано 5 следующих моделей.

Определим входы и выход для модели

```
✓ [353] target_var = df['Соотношение матрица-наполнитель']  
0.5 train_vars = df.loc[:, df.columns != 'Соотношение матрица-наполнитель']  
  
# Разбиваем выборки на обучающую и тестовую  
x_train, x_test, y_train, y_test = train_test_split(train_vars, target_var, test_size=0.3, random_state=14)
```

Нормализация

```
✓ [354] x_train_normalizer = tf.keras.layers.Normalization(axis=-1)  
0.5 x_train_normalizer.adapt(np.array(x_train))
```

Рисунок 27. Предобработка данных на подачу в нейросеть

1) Модель 1

Создание модели и слоёв

```
[357] X_train_norm = X_train_normalizer

model = tf.keras.Sequential([X_train_norm, layers.Dense(128, activation='relu'),
                             layers.Dense(64, activation='relu'),
                             layers.Dense(32, activation='relu'),
                             layers.Dense(16, activation='relu'),
                             layers.Dense(1)
                             ])

model.compile(optimizer=tf.keras.optimizers.Adam(0.001), loss='mean_squared_error')
```

Архитектура модели

```
[358] model.summary()

Model: "sequential_6"
_____
Layer (type)                 Output shape          Param #
-----
normalization_2 (Normalizat (None, 12)            25
ion)

dense_26 (Dense)             (None, 128)           1664
dense_27 (Dense)             (None, 64)            8256
dense_28 (Dense)             (None, 32)            2080
dense_29 (Dense)             (None, 16)            528
dense_30 (Dense)             (None, 1)             17
_____
Total params: 12,570
Trainable params: 12,545
Non-trainable params: 25
_____
```

Рисунок 28. Создание слоёв и архитектуры модели

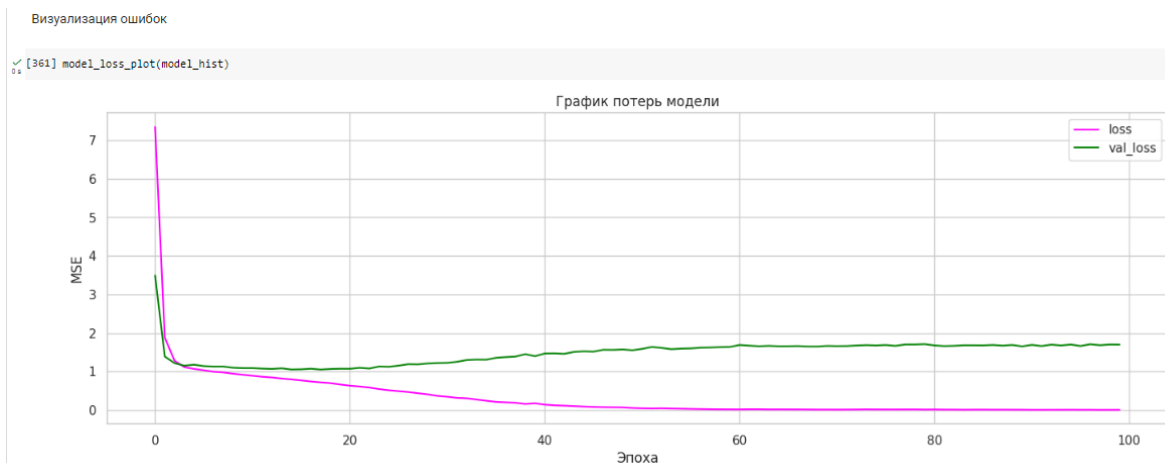


Рисунок 29. Визуализация ошибок модели

Результат работы модели

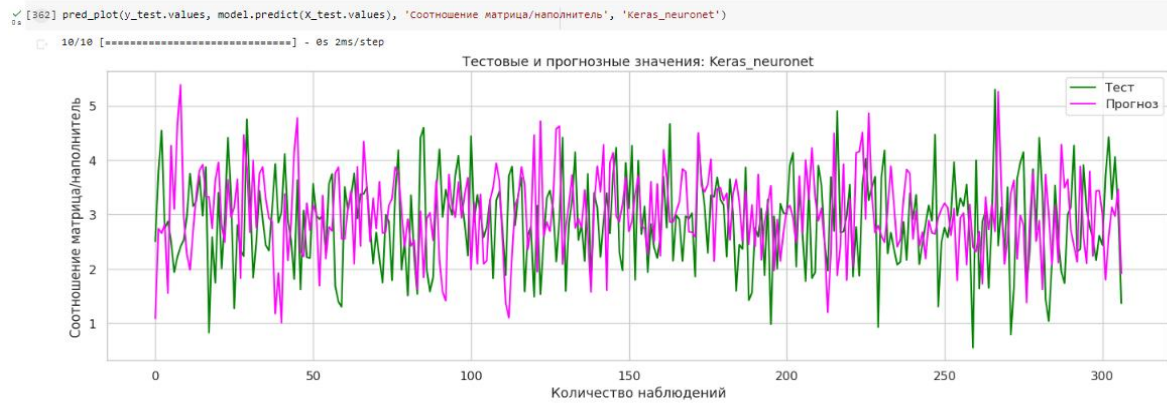


Рисунок 30. Результат работы модели

2) Модель 2

```
model_2 = tf.keras.Sequential([X_train_normalizer, layers.Dense(128, activation='LeakyReLU'),
                                layers.Dropout(0.18),
                                layers.Dense(128, activation='relu'),
                                BatchNormalization(),
                                layers.Dropout(0.18),
                                layers.Dense(64, activation='relu'),
                                BatchNormalization(),
                                layers.Dropout(0.18),
                                layers.Dense(32, activation='relu'),
                                BatchNormalization(),
                                layers.Dropout(0.18),
                                layers.Dense(16, activation='relu'),
                                BatchNormalization(),
                                layers.Dense(1, activation='linear')
                                ])

model_2.compile(optimizer=tf.keras.optimizers.Adam(0.001), loss='mean_squared_error')
```

Рисунок 31. Создание слоёв и архитектуры второй модели

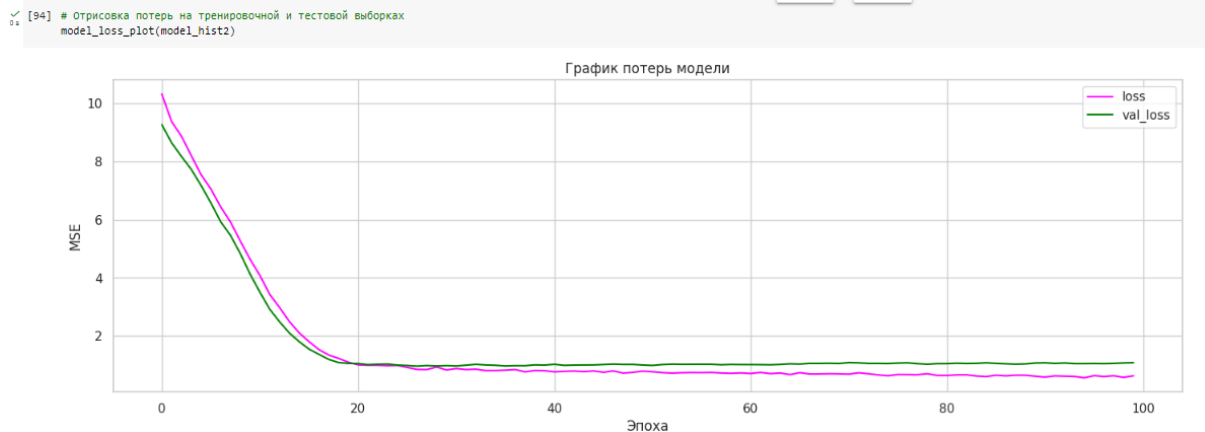


Рисунок 32. Визуализация ошибок второй модели



Рисунок 33. Результат работы второй модели

3) Модель 3

```
[102] model_3 = tf.keras.Sequential([X_train_normalizer, layers.Dense(256, activation='LeakyReLU'),
                                     BatchNormalization(),
                                     layers.Dropout(0.18),
                                     layers.Dense(256, activation='relu'),
                                     BatchNormalization(),
                                     layers.Dropout(0.18),
                                     layers.Dense(128, activation='LeakyReLU'),
                                     BatchNormalization(),
                                     layers.Dropout(0.18),
                                     layers.Dense(128, activation='relu'),
                                     BatchNormalization(),
                                     layers.Dropout(0.18),
                                     layers.Dense(64, activation='relu'),
                                     BatchNormalization(),
                                     layers.Dropout(0.18),
                                     layers.Dense(32, activation='relu'),
                                     BatchNormalization(),
                                     layers.Dropout(0.18),
                                     layers.Dense(16, activation='relu'),
                                     BatchNormalization(),
                                     layers.Dense(1, activation='linear')
                                     ])

model_3.compile(optimizer=tf.keras.optimizers.Adam(0.001), loss='mean_squared_error')
```

Рисунок 34. Создание слоёв и архитектуры третьей модели

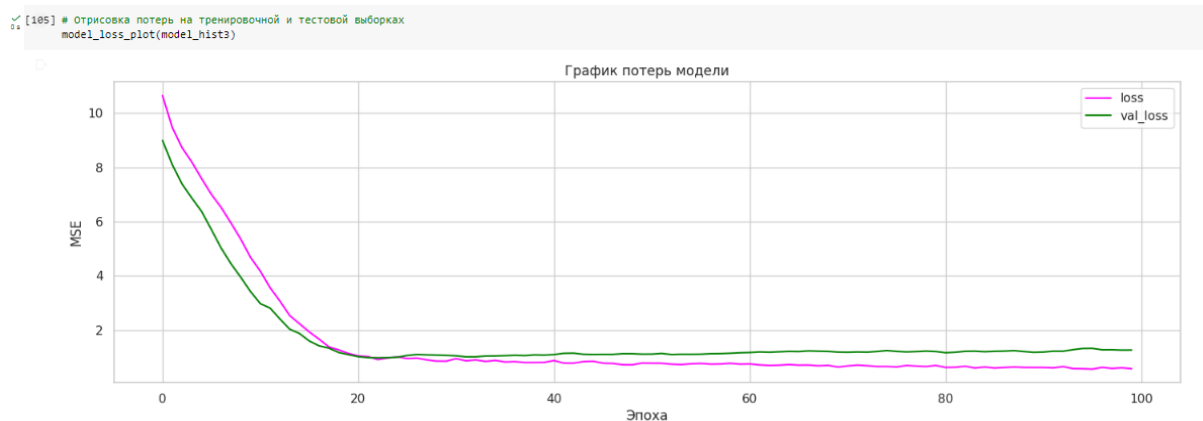


Рисунок 35. Визуализация ошибок третьей модели

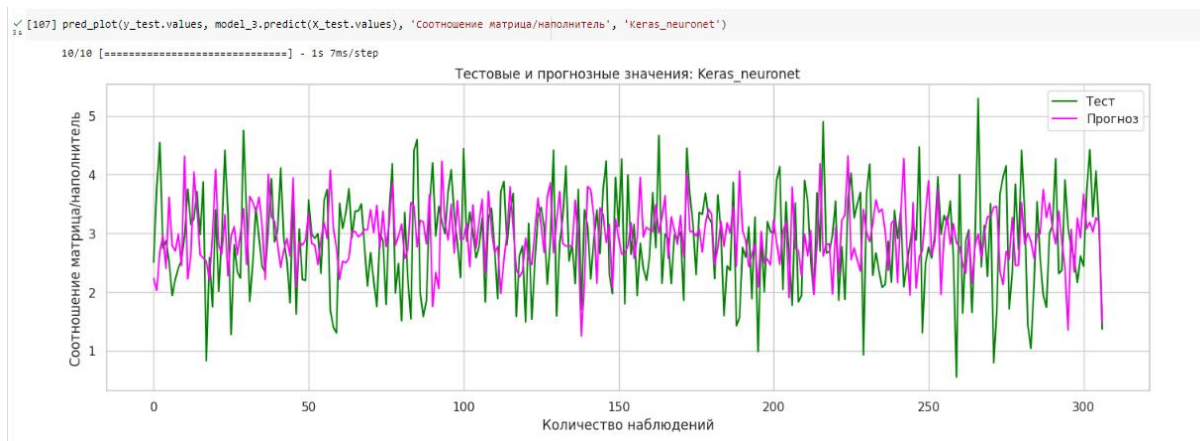


Рисунок 36. Результат работы третьей модели

4) Модель 4

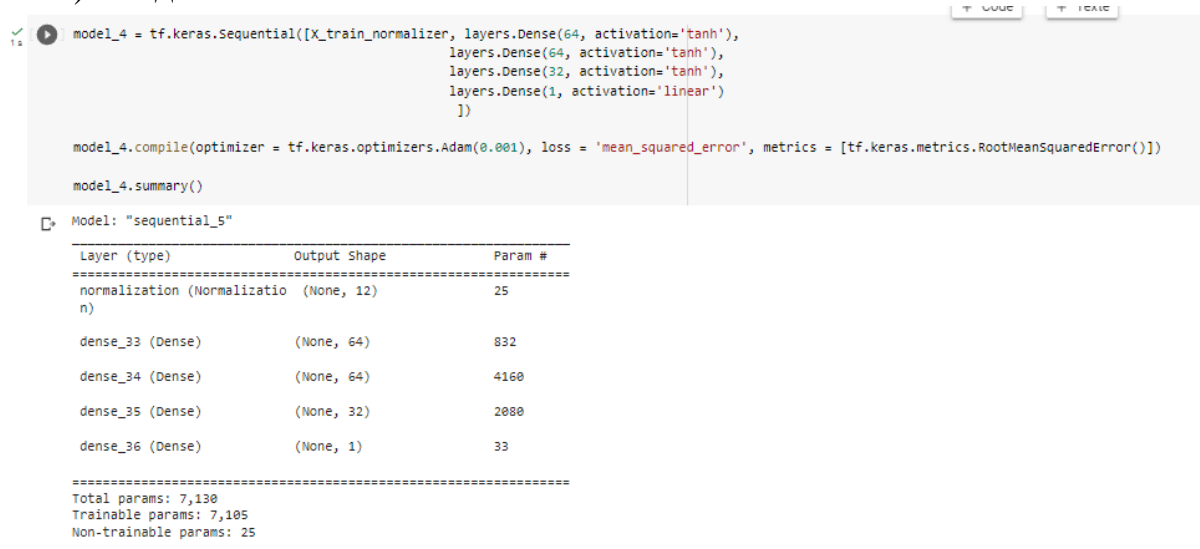


Рисунок 37. Создание слоёв и архитектуры четвертой модели

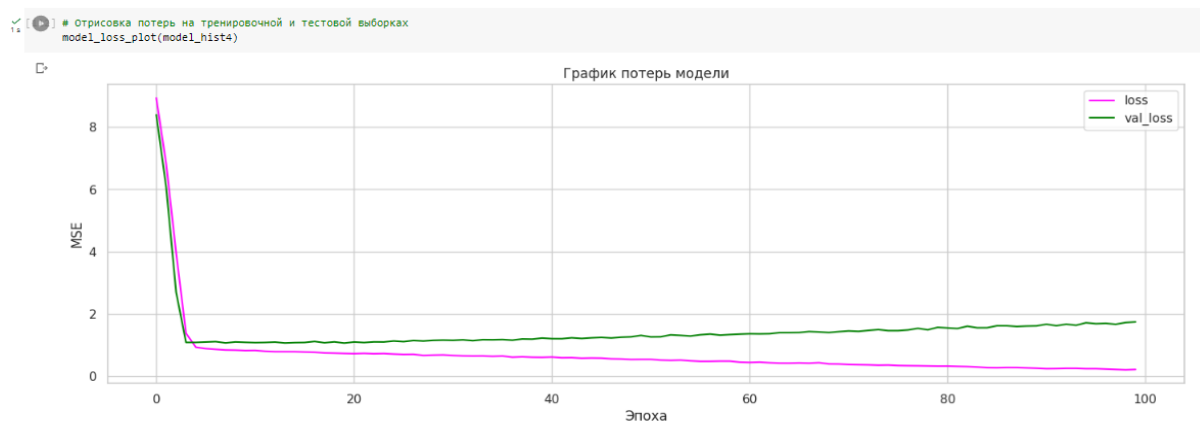


Рисунок 38. Визуализация ошибок четвертой модели

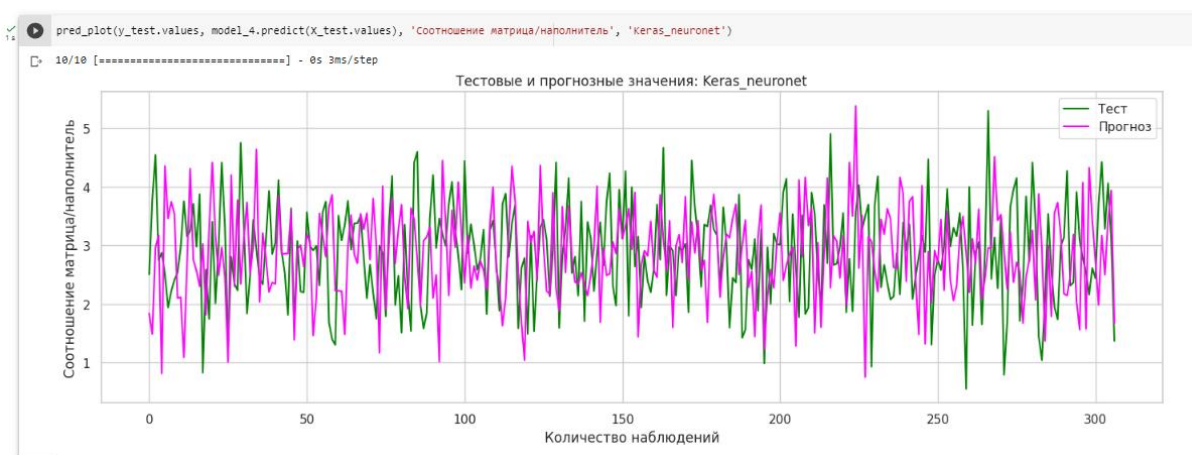


Рисунок 38. Результат работы четвертой модели

5) Модель 5

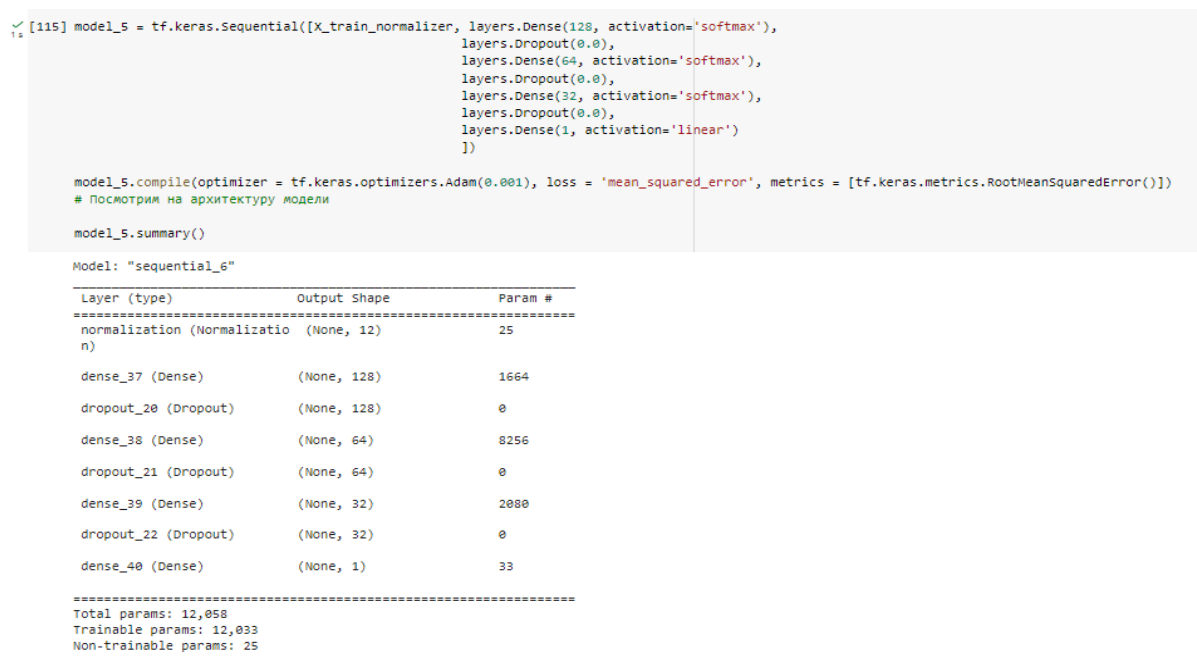


Рисунок 39. Создание слоёв и архитектуры пятой модели

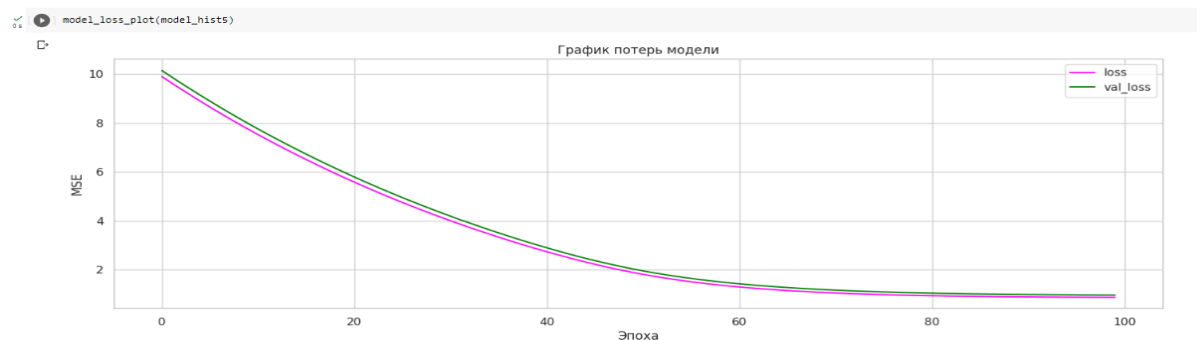


Рисунок 40. Визуализация ошибок пятой модели



Рисунок 41. Результат работы пятой модели

	target_var	model_name	MAE	MAPE	MSE	RMSE	R2
0	Модуль упругости при растяжении, ГПа	K-NN	0.17	6526587660746.40	0.04	0.21	-0.18
1	Прочность при растяжении, МПа	K-NN	0.18	0.68	0.05	0.23	-0.22
2	Модуль упругости при растяжении, ГПа	Linear_Regression	0.16	9232724774322.15	0.04	0.19	-0.01
3	Прочность при растяжении, МПа	Linear_Regression	0.17	0.65	0.04	0.21	-0.03
4	Модуль упругости при растяжении, ГПа	Random_Forest	0.16	8769553567468.01	0.04	0.20	-0.09
5	Прочность при растяжении, МПа	Random_Forest	0.17	0.67	0.04	0.21	-0.06
6	Модуль упругости при растяжении, ГПа	Support_Vector_Regression	0.15	8176304369191.76	0.04	0.19	-0.00
7	Прочность при растяжении, МПа	Support_Vector_Regression	0.17	0.68	0.04	0.20	-0.00
8	Модуль упругости при растяжении, ГПа	Gradient_Boosting	0.16	8395674120401.24	0.04	0.20	-0.05
9	Прочность при растяжении, МПа	Gradient_Boosting	0.17	0.68	0.05	0.21	-0.10
10	Модуль упругости при растяжении, ГПа	Decision Tree Regression	0.20	8286233635861.67	0.06	0.25	-0.68
11	Прочность при растяжении, МПа	Decision Tree Regression	0.25	0.85	0.09	0.30	-1.22
12	Модуль упругости при растяжении, ГПа	Lasso_Regressor	0.15	8183534682026.43	0.04	0.19	-0.00
13	Прочность при растяжении, МПа	Lasso_Regressor	0.17	0.65	0.04	0.21	-0.01
14	Соотношение матрица/наполнитель	Model 1	0.94	0.41	1.44	1.20	-1.02
15	Соотношение матрица/наполнитель	Model 2	2.78	0.96	8.47	2.91	-10.94
16	Соотношение матрица/наполнитель	Model 3	0.80	0.34	0.95	0.98	-0.34
17	Соотношение матрица/наполнитель	Model 4	0.95	0.39	1.34	1.16	-0.90
18	Соотношение матрица/наполнитель	Model 5	0.68	0.30	0.71	0.84	-0.00

Рисунок 42. Объединенная таблица результатов всех моделей

Проанализировав полученные результаты, можно сделать следующие выводы. По метрике минимизации средней абсолютной ошибки (MAE) Model 3 и Model 5 справились лучше остальных, так как они имеют

наименьшее значение MAE. По метрике минимизация среднеквадратической ошибки (MSE) Model 5 справилась лучше, так как имеет наименьшее значение MSE.

Также следует учитывать, что R^2 (коэффициент детерминации) отражает соответствие модели данным, поэтому отрицательные значения этой метрики могут указывать на то, что модель не соответствует данным вообще, и ее не следует использовать.

Обобщая вывод, можно сказать, что Model 5 будет наилучшим выбором, так как имеет наименьшее значение MAE и R^2 близок к нулю.

2.4. Разработка приложения

В работе было проделана попытка создания web приложения и создание консольного приложения в Google Colab.

Рассмотрим созданное консольное приложение.

Создание функции с переменными для приложения

```
✓ 0 s ▶ def input_variable():
    a = float(input('Плотность, кг/м3: '))
    b = float(input('Модуль упругости, ГПа: '))
    c = float(input('Количество отвердителя, м.-%: '))
    d = float(input('Содержание эпоксидных групп,%_2: '))
    e = float(input('Температура вспышки, С_2: '))
    f = float(input('Поверхностная плотность, г/м2: '))
    g = float(input('Модуль упругости при растяжении, ГПа: '))
    h = float(input('Прочность при растяжении, МПа: '))
    i = float(input('Потребление смолы, г/м2: '))
    j = float(input('Угол нашивки: '))
    k = float(input('Шаг нашивки: '))
    l = float(input('Плотность нашивки: '))
    return a, b, c, d, e, f, g, h, i, j, k, l
```

Рисунок 43. Создание функции с переменными для приложения

Создание функции приложения

```
▶ def app_model():
    #nn_model = load_model('C:/Users/serzh/OneDrive/Bureau//model_1/')
    application_model = model
    data_x = load('/content/df_x_minmax.pkl')
    data_y = load('/content/df_y_minmax.pkl')

    print('Приложение даёт прогноз соотношение матрица-наполнитель')
    for i in range(1000):
        try:
            print('Введите "да" для прогноза, "нет" для выхода')
            check = input()

            if check == 'да':
                print('Введите данные')
                X = input_variable()
                X = data_x.transform(np.array(X).reshape(1,-1))
                prediction = application_model.predict(X)
                output = data_y.inverse_transform(prediction)
                print('Прогноз значения соотношение матрица-наполнитель: ')
                print(output[0][0])

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

        except Exception as e:
            print(e)
            print('Данные неверны. Повторите ввод')

app_model()
```

Рисунок 44. Создание функции приложения

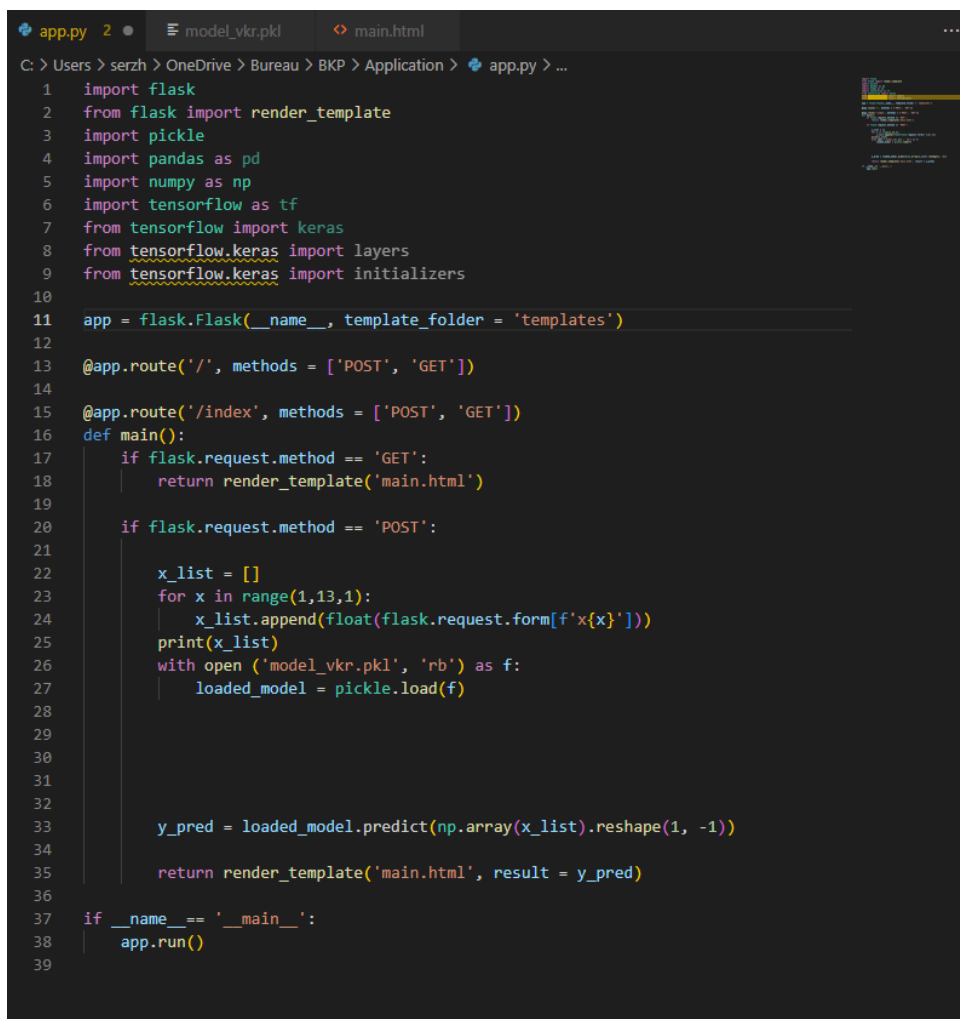
```

Приложение даёт прогноз соотношение матрица-наполнитель
Введите "да" для прогноза, "нет" для выхода
да
Введите данные
Плотность, кг/м3: 1
Модуль упругости, ГПа: 1
Количество отвердителя, м.-%: 1
Содержание эпоксидных групп, %_2: 1
Температура вспышки, C_2: 1
Поверхностная плотность, г/м2: 1
Модуль упругости при растяжении, ГПа: 1
Прочность при растяжении, МПа: 1
Потребление смолы, г/м2: 1
Угол нашивки: 1
Шаг нашивки: 1
Плотность нашивки: 1
1/1 [=====] - 0s 41ms/step
Прогноз значения соотношение матрица-наполнитель:
54.649784
Введите "да" для прогноза, "нет" для выхода
нет

```

Рисунок 45. Результат работы консольного приложения

Рассмотрим попытку создания web приложения на Flask.



```

app.py 2 • model_vkr.pkl main.html
C: > Users > serzh > OneDrive > Bureau > BKP > Application > app.py > ...
1  import flask
2  from flask import render_template
3  import pickle
4  import pandas as pd
5  import numpy as np
6  import tensorflow as tf
7  from tensorflow import keras
8  from tensorflow.keras import layers
9  from tensorflow.keras import initializers
10
11 app = flask.Flask(__name__, template_folder = 'templates')
12
13 @app.route('/', methods = ['POST', 'GET'])
14
15 @app.route('/index', methods = ['POST', 'GET'])
16 def main():
17     if flask.request.method == 'GET':
18         return render_template('main.html')
19
20     if flask.request.method == 'POST':
21
22         x_list = []
23         for x in range(1,13,1):
24             x_list.append(float(flask.request.form[f'x{x}']))
25         print(x_list)
26         with open ('model_vkr.pkl', 'rb') as f:
27             loaded_model = pickle.load(f)
28
29
30
31
32
33         y_pred = loaded_model.predict(np.array(x_list).reshape(1, -1))
34
35         return render_template('main.html', result = y_pred)
36
37 if __name__ == '__main__':
38     app.run()
39

```

Рисунок 46. Разработка приложения на Flask

```

1  <html><head><style>
2      form {
3          margin: auto;
4          width: 35%;
5      }
6
7      .result {
8          margin: auto;
9          width: 35%;
10         border: 1px solid #ccc;
11     }
12 </style>
13
14
15     <title>Рекомендация соотношения матрица-наполнитель</title>
16 </head>
17
18 <body><form action="/index" method="POST">
19     <fieldset>
20         <legend>Введите значение</legend>
21
22         x1
23         <input name="x1" type="number" required="">
24
25         <br>
26         <br>
27
28         x2
29         <input name="x2" type="number" required="">
30
31         <br>
32         <br>
33
34         x3
35         <input name="x3" type="number" required="">
36
37         <br>
38         <br>
39
40         x4
41         <input name="x4" type="number" required="">
42
43         <br>
44         <br>
45
46         x5
47         <input name="x5" type="number" required="">
48

```

Рисунок 47. Разработка HTML страницы для приложения (1)

```
app.py 2  model_vkr.pkl  main.html X
C: > Users > serzh > OneDrive > Bureau > BKP > Application > templates > > main.html > html

52      x6
53      <input name="x6" type="numder" required="">
54
55      <br>
56      <br>
57
58      x7
59      <input name="x7" type="numder" required="">
60
61      <br>
62      <br>
63
64      x8
65      <input name="x8" type="numder" required="">
66
67      <br>
68      <br>
69
70      x9
71      <input name="x9" type="numder" required="">
72
73      <br>
74      <br>
75
76      x10
77      <input name="x10" type="numder" required="">
78
79      <br>
80      <br>
81
82      x11
83      <input name="x11" type="numder" required="">
84
85      <br>
86      <br>
87
88      x12
89      <input name="x12" type="numder" required="">
90
91      <br>
92      <br>
93
94      <input type="submit">
95  </fieldset>
96  </form>
97
98  <div class="result" align="center">
99
```

Рисунок 48. Разработка HTML страницы для приложения (2)

Введите значение

x1

x2

x3

x4

x5

x6

x7

x8

x9

x10

x11

x12

Рисунок 49. Запуск web приложения

2.5. Создание репозитория на GitHub

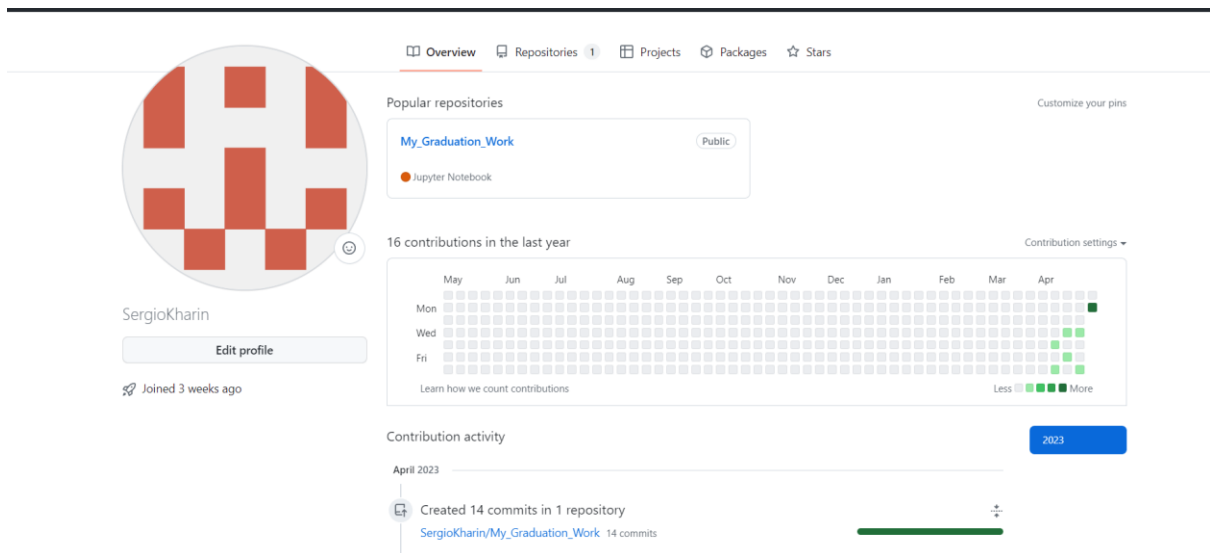


Рисунок 50. Профиль на GitHub

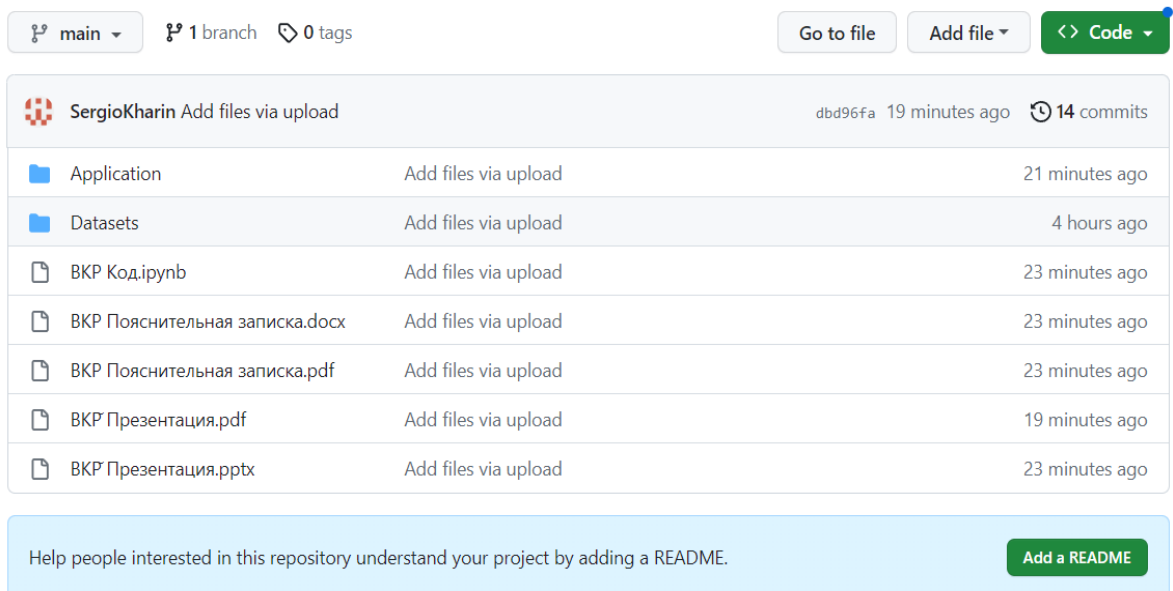


Рисунок 50. Выложенная ВКР на GitHub

Ссылка на репозиторий :

https://github.com/SergioKharin/My_Graduation_Work

Заключение

Работа с моделями прогнозирования, использующими машинное обучение, является сложным процессом, который требует как навыков программирования, так и профессиональных знаний в области композитных материалов. Чтобы обеспечить достоверность полученных прогнозов, необходимо иметь контакт с экспертами в данной области для консультаций. В ходе работы был проведен анализ датасета, построено множество графиков и выполнено разбиение данных на обучающую и тестовую выборки. Для реализации моделей машинного обучения были применены различные алгоритмы, включая метод К ближайших соседей, линейную регрессию, деревья решений, опорные вектора и случайный лес. Были составлены отчеты, оценивающие качество проводимого обучения, а также сравнение результатов работы моделей с помощью графиков и диаграмм. Была разработана нейронная сеть. Из результатов исследования стало понятно, что для повышения достоверности прогнозов необходимо дополнить датасет недостающими данными, а для этого необходимо иметь команду из специалистов по данной области для консультаций и сбора информации. Метрика, которую следует использовать для оценки прочности при растяжении, - метод градиентного бустинга, а для модуля упругости при растяжении - метод опорных векторов.

Список литературы

1. Aroorva, D. Регрессия (Regression) [Электронный ресурс] : – Режим доступа: <https://www.helenkapatsa.ru/rieghriessiia/> (дата обращения: 14.04.2023).
2. 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).
3. 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).
4. Keras - последовательная модель Sequential [Электронный ресурс] : – Режим доступа: <https://proproprogs.ru/tensorflow/keras-posledovatel'naya-model-sequential> (дата обращения: 23.04.2023).
5. 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: 24.04.2023).
6. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Обработка данных с использованием Python, SQL и машинного обучения [Текст] / Пер. с англ. – М.: ДМК Пресс, 2019. – 442 с.
7. Сравнительное изучение алгоритмов классического машинного обучения [Электронный ресурс] : – Режим доступа: <https://machinelearningmastery.ru/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222/> (дата обращения: 13.04.2023).

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

9. Что такое нейронная сеть? [Электронный ресурс] : – Режим доступа: <https://aws.amazon.com/ru/what-is/neural-network/> (дата обращения: 24.04.2023).

10. 2.3. Разведочный анализ данных (рад) [Электронный ресурс] : – Режим доступа: <https://studfile.net/preview/8858767/page:8/> (дата обращения: 05.04.2023).

11. Документация Seaborn [Электронный ресурс]: – Режим доступа: <https://seaborn.pydata.org/> (дата обращения: 21.04.2023).

12. Документация Flask [Электронный ресурс]: – Режим доступа: <https://pypi.org/project/Flask/> (дата обращения: 21.04.2023).

13. Документация Plotly [Электронный ресурс]: – Режим доступа: <https://pypi.org/project/Plotly/> (дата обращения: 21.04.2023).