

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное 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 новых материалов
(композиционных материалов).

Слушатель

Запорожец Марина Владимировна

Москва, 2023

Содержание

Содержание	2
Введение	4
1 Аналитическая часть.....	6
1.1 Постановка задачи	6
1.2 Описание используемых методов	8
1.2.1 LinearRegression (линейная регрессия).....	8
1.2.2 KNeighborsRegressor (метод k-ближайших соседей)	9
1.2.3 Random Forest (случайный лес)	10
1.2.4 Support Vector Regression (SVR, Метод опорных векторов)	11
1.2.5 GradientBoostingRegressor (Градиентный бустинг).....	12
1.2.6 Полносвязная нейронная сеть.....	13
1.3 Разведочный анализ данных	14
1.3.1 Проверка распределения данных, нормальное распределение.....	14
1.3.2 Исследование зависимостей в данных, корреляция.....	17
1.3.3 Исследование выбросов в данных, удаление выбросов	19
1.3.4 Ход решения задачи.....	23
1.3.5 Препроцессинг	24
1.3.6 Перекрестная проверка.....	25
1.3.7 Поиск гиперпараметров по сетке, случайный поиск	25
1.3.8 Метрики качества моделей	26
2 Практическая часть	27
2.1 Разбиение и предобработка данных.....	27
2.2 Разработка и обучение модели	29

2.3 Таблицы работы моделей	40
2.3.1 Модуль упругости при растяжении, Гпа	40
2.3.2 Прочность при растяжении, МПа	41
2.4 Написать нейронную сеть, рекомендующую соотношение матрица-наполнитель	42
2.5 Разработка приложения.....	45
2.6 Удаленный репозиторий и загрузка работы на него	48
Заключение.....	51
Библиографический список	52

Введение

Композиционные материалы — это искусственно созданные материалы, состоящие из нескольких других с четкой границей между ними. Композиты обладают теми свойствами, которые не наблюдаются у компонентов по отдельности. При этом композиты являются монолитным материалом, т. е. компоненты материала неотделимы друг от друга без разрушения конструкции в целом. Яркий пример композита - железобетон. Бетон прекрасно сопротивляется сжатию, но плохо растяжению. Стальная арматура внутри бетона компенсирует его неспособность сопротивляться сжатию, формируя тем самым новые, уникальные свойства. Современные композиты изготавливаются из других материалов: полимеры, керамика, стеклянные и углеродные волокна, но данный принцип сохраняется. У такого подхода есть и недостаток: даже если мы знаем характеристики исходных компонентов, определить характеристики композита, состоящего из этих компонентов, достаточно проблематично. Для решения этой проблемы есть два пути: физические испытания образцов материалов, или прогнозирование характеристик. Суть прогнозирования заключается в симуляции представительного элемента объема композита, на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

На входе имеются данные о начальных свойствах компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.). На выходе необходимо спрогнозировать ряд конечных свойств получаемых композиционных материалов. Кейс основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

Актуальность: Созданные прогнозные модели помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов

возможными новыми характеристиками материалов, и цифровыми двойниками новых композитов.

Датасет со свойствами композитов. Объединение делать по индексу тип объединения INNER

https://drive.google.com/file/d/1B1s5gBlvgU81H9GGolLQVw_SOi-vyNf2/view?usp=sharing

1 Аналитическая часть

1.1 Постановка задачи

От специалистов в предметной области был получен датасет, содержащий данные о свойствах матрицы и наполнителя, производственных параметрах и свойствах готового композита. От нас, как специалистов в машинном обучении, требуется разработать модели, прогнозирующие значения некоторых свойств в зависимости от остальных. Так же требуется разработать приложение, делающее удобным использование данных моделей специалистом предметной области.

Датасет состоит из двух файлов: X_bp и X_nip.

Файл X_bp содержит:

- признаков: 10 и индекс;
- строк: 1023.

Файл X_nip содержит:

- признаков: 3 и индекс;
- строк: 1040.

Известно, что файлы требуют объединения с типом INNER по индексу. После объединения часть строк из файла X_nip была отброшена. И дальнейшие исследования проводим с объединенным датасетом, содержащим 13 признаков и 1023 строк или объектов.

Описание признаков объединенного датасета приведено на рисунке 1. Кроме значения «Угол нашивки, град», все признаки имеют тип float64, то есть значения представляют собой вещественные числа. Пропусков в данных нет. Все признаки, кроме «Угол нашивки», являются непрерывными, количественными. «Угол нашивки» принимает только два значения, однако, мы не будем его рассматривать как категориальный признак, значения угла нашивки могут быть различными. Оставим его числовым значением, тип int64.

```
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, С_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 1 - Описание признаков объединенного датасета

Пропусков (Рисунок 2) и дубликатов (Рисунок 3) объединенный датасет не содержит.

```
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, С_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 2 – Проверка на наличие пропусков в датасете

```
df.duplicated().sum()
```

0

Рисунок 3 – Проверка на наличие дубликатов в датасете

Так же нас интересует описательная статистика датасета. Таблица описательной статистики представлена на рисунке 4.

	Соотношение матрица-наполнитель	Плотность, кг/м ³	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп, %_2	Температура вспышки, С_2	Поверхностная плотность, г/м ²	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м ²	Угол нашивки, град	Шаг нашивки	Плотность нашивки
count	1023.00	1023.00	1023.00	1023.00	1023.00	1023.00	1023.00	1023.00	1023.00	1023.00	1023.00	1023.00	1023.00
mean	2.93	1975.73	739.92	110.57	22.24	285.88	482.73	73.33	2466.92	218.42	44.25	6.90	57.15
std	0.91	73.73	330.23	28.30	2.41	40.94	281.31	3.12	485.63	59.74	45.02	2.56	12.35
min	0.39	1731.76	2.44	17.74	14.25	100.00	0.60	64.05	1036.86	33.80	0.00	0.00	0.00
25%	2.32	1924.16	500.05	92.44	20.61	259.07	266.82	71.25	2135.85	179.63	0.00	5.08	49.80
50%	2.91	1977.62	739.66	110.56	22.23	285.90	451.86	73.27	2459.52	219.20	0.00	6.92	57.34
75%	3.55	2021.37	961.81	129.73	23.96	313.00	693.23	75.36	2767.19	257.48	90.00	8.59	64.94
max	5.59	2207.77	1911.54	198.95	33.00	413.27	1399.54	82.68	3848.44	414.59	90.00	14.44	103.99

Рисунок 4 – описательная статистика датасета

Статистика содержит показатели:

- count — количество наблюдений;
- mean — среднее арифметическое;
- std или standard deviation — среднее квадратическое отклонение;
- min и max — минимальное и максимальное значения;
- 25%, 50% и 75% — первый, второй (он же медиана) и третий квартили.

1.2 Описание используемых методов

1.2.1 LinearRegression (линейная регрессия)

Простая линейная регрессия имеет место, если рассматривается зависимость между одной входной и одной выходной переменными. Для этого определяется уравнение регрессии (1) и строится соответствующая прямая, известная как линия регрессии.

$$y = ax + b \quad (1)$$

Коэффициенты a и b , называемые также параметрами модели, определяются таким образом, чтобы сумма квадратов отклонений точек, соответствующих реальным наблюдениям данных, от линии регрессии была бы минимальной. Коэффициенты обычно оцениваются методом наименьших квадратов.

Если ищется зависимость между несколькими входными и одной выходной переменными, то имеет место множественная линейная регрессия. Соответствующее уравнение имеет вид (2).

$$Y = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_n * x_n, \quad (2)$$

где n - число входных переменных.

Очевидно, что в данном случае модель будет описываться не прямой, а гиперплоскостью. Коэффициенты уравнения множественной линейной регрессии подбираются так, чтобы минимизировать сумму квадратов отклонения реальных точек данных от этой гиперплоскости.

Линейная регрессия — первый тщательно изученный метод регрессионного анализа. Его главное достоинство — простота. Такую модель можно построить и рассчитать даже без мощных вычислительных средств. Простота является и главным недостатком этого метода. Тем не менее, именно с линейной регрессии целесообразно начать подбор подходящей модели.

На языке Python линейная регрессия реализована в `sklearn.linear_model.LinearRegression`.

1.2.2 KNeighborsRegressor (метод k-ближайших соседей)

Это метод классификации, который адаптирован для регрессии - метод k -ближайших соседей (k Nearest Neighbors). На интуитивном уровне суть метода проста: посмотри на соседей вокруг, какие из них преобладают, таковыми ты и являешься.

В случае использования метода для регрессии, объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны.

Для реализации метода необходима метрика расстояния между объектами. Используется, например, евклидово расстояние для количественных признаков или расстояние Хэмминга для категориальных.

Этот метод — пример непараметрической регрессии.

Он реализован в `sklearn.neighbors.KNeighborsRegressor`.

1.2.3 Random Forest (случайный лес)

Случайный лес (`RandomForest`) — представитель ансамблевых методов.

Если точность дерева решений оказалось недостаточной, мы можем множество моделей собрать в коллектив. Формула итогового решателя (3) — это усреднение предсказаний отдельных деревьев.

$$a(x) = \frac{1}{N} \sum_{i=1}^N b_i(x) \quad (3),$$

где

N — количество деревьев;

i — счетчик для деревьев;

b — решающее дерево;

x — сгенерированная нами на основе данных выборка.

Для определения входных данных каждому дереву используется метод случайных подпространств. Базовые алгоритмы обучаются на различных подмножествах признаков, которые выделяются случайнным образом.

Преимущества случайного леса:

- высокая точность предсказания;
- редко переобучается;
- практически не чувствителен к выбросам в данных;
- одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки, данные с большим числом признаков;
- высокая параллелизуемость и масштабируемость.

Из недостатков можно отметить, что его построение занимает больше времени. Так же теряется интерпретируемость.

Метод реализован в `sklearn.ensemble.RandomForestRegressor`.

1.2.4 Support Vector Regression (SVR, Метод опорных векторов)

Метод опорных векторов (support vector machine, SVM) — один из наиболее популярных методов машинного обучения. Он создает гиперплоскость или набор гиперплоскостей в многомерном пространстве, которые могут быть использованы для решения задач классификации и регрессии.

Чаще всего он применяется в постановке бинарной классификации.

Основная идея заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Интуитивно, хорошее разделение достигается за счет гиперплоскости, которая имеет самое большое расстояние до ближайшей точки обучающей выборке любого класса. Максимально близкие объекты разных классов определяют опорные вектора.

Если в исходном пространстве объекты линейно неразделимы, то выполняется переход в пространство большей размерности.

Решается задача оптимизации.

Для вычислений используется ядерная функция, получающая на вход два вектора и возвращающая меру сходства между ними:

- линейная;
- полиномиальная;
- гауссовская (rbf).

Эффективность метода опорных векторов зависит от выбора ядра, параметров ядра и параметра С для регуляризации.

Преимущество метода — его хорошая изученность.

Недостатки:

- чувствительность к выбросам;
- отсутствие интерпретируемости.

Вариация метода для регрессии называется SVR (Support Vector Regression).

В Python реализацию SVR можно найти в `sklearn.svm.SVR`.

1.2.5 GradientBoostingRegressor (Градиентный бустинг)

Градиентный бустинг (GradientBoosting) — еще один представитель ансамблевых методов.

В отличие от случайного леса, где каждый базовый алгоритм строится независимо от остальных, бустинг воплощает идею последовательного построения линейной комбинации алгоритмов. Каждый следующий алгоритм старается уменьшить ошибку предыдущего.

Чтобы построить алгоритм градиентного бустинга, нам необходимо выбрать базовый алгоритм и функцию потерь или ошибки (loss). Loss-функция — это мера, которая показывает, насколько хорошо предсказание модели соответствуют данным. Используя градиентный спуск и обновляя предсказания, основанные на скорости обучения (learning rate), ищем значения, на которых loss минимальна.

Бустинг, использующий деревья решений в качестве базовых алгоритмов, называется градиентным бустингом над решающими деревьями. Он отлично работает на выборках с «табличными», неоднородными данными и способен эффективно находить нелинейные зависимости в данных различной природы. На настоящий момент это один из самых эффективных алгоритмов машинного обучения. Благодаря этому он широко применяется во многих конкурсах и промышленных задачах. Он проигрывает только нейросетям на однородных данных (изображения, звук и т. д.).

Из недостатков алгоритма можно отметить только затраты времени на вычисления и необходимость грамотного подбора гиперпараметров.

В этой работе я использую реализацию градиентного бустинга из библиотеки `sklearn` — `sklearn.ensemble.GradientBoostingRegressor`. Хотя

существуют и другие реализации, некоторые из которых более мощные, например, XGBoost.

1.2.6 Полносвязная нейронная сеть

Нейронная сеть — это последовательность нейронов, соединенных между собой связями. Структура нейронной сети пришла в мир программирования из биологии. Вычислительная единица нейронной сети — нейрон или персептрон.

У каждого нейрона есть определённое количество входов, куда поступают сигналы, которые суммируются с учётом значимости (веса) каждого входа.

Смещение — это дополнительный вход для нейрона, который всегда равен 1 и, следовательно, имеет собственный вес соединения.

Так же у нейрона есть функция активации, которая определяет выходное значение нейрона. Она используется для того, чтобы ввести нелинейность в нейронную сеть. Примеры активационных функций: relu, sigmoid.

У полносвязной нейросети выход каждого нейрона подается на вход всем нейронам следующего слоя. У нейросети имеется:

- входной слой — его размер соответствует входным параметрам;
- скрытые слои — их количество и размерность определяем специалист;
- выходной слой — его размер соответствует выходным параметрам.

Прямое распространение — это процесс передачи входных значений в нейронную сеть и получения выходных данных, которые называются прогнозируемым значением.

Прогнозируемое значение сравниваем с фактическим с помощью функции потери. В методе обратного распространения ошибки градиенты (производные значений ошибок) вычисляются по значениям весов в направлении, обратном прямому распространению сигналов. Значение градиента вычитают из значения веса, чтобы уменьшить значение ошибки. Таким образом происходит процесс обучения. Обновляются веса каждого соединения, чтобы функция потерь минимизировалась.

Для обновления весов в модели используются различные оптимизаторы.

Количество эпох показывает, сколько раз выполнялся проход для всех примеров обучения.

Нейронные сети применяются для решения задач регрессии, классификации, распознавания образов и речи, компьютерного зрения и других. На настоящий момент это самый мощный, гибкий и широко применяемый инструмент в машинном обучении.

1.3 Разведочный анализ данных

1.3.1 Проверка распределения данных, нормальное распределение

Проверка на «нормальность» - гистограммы распределения значений (Рисунок 5).

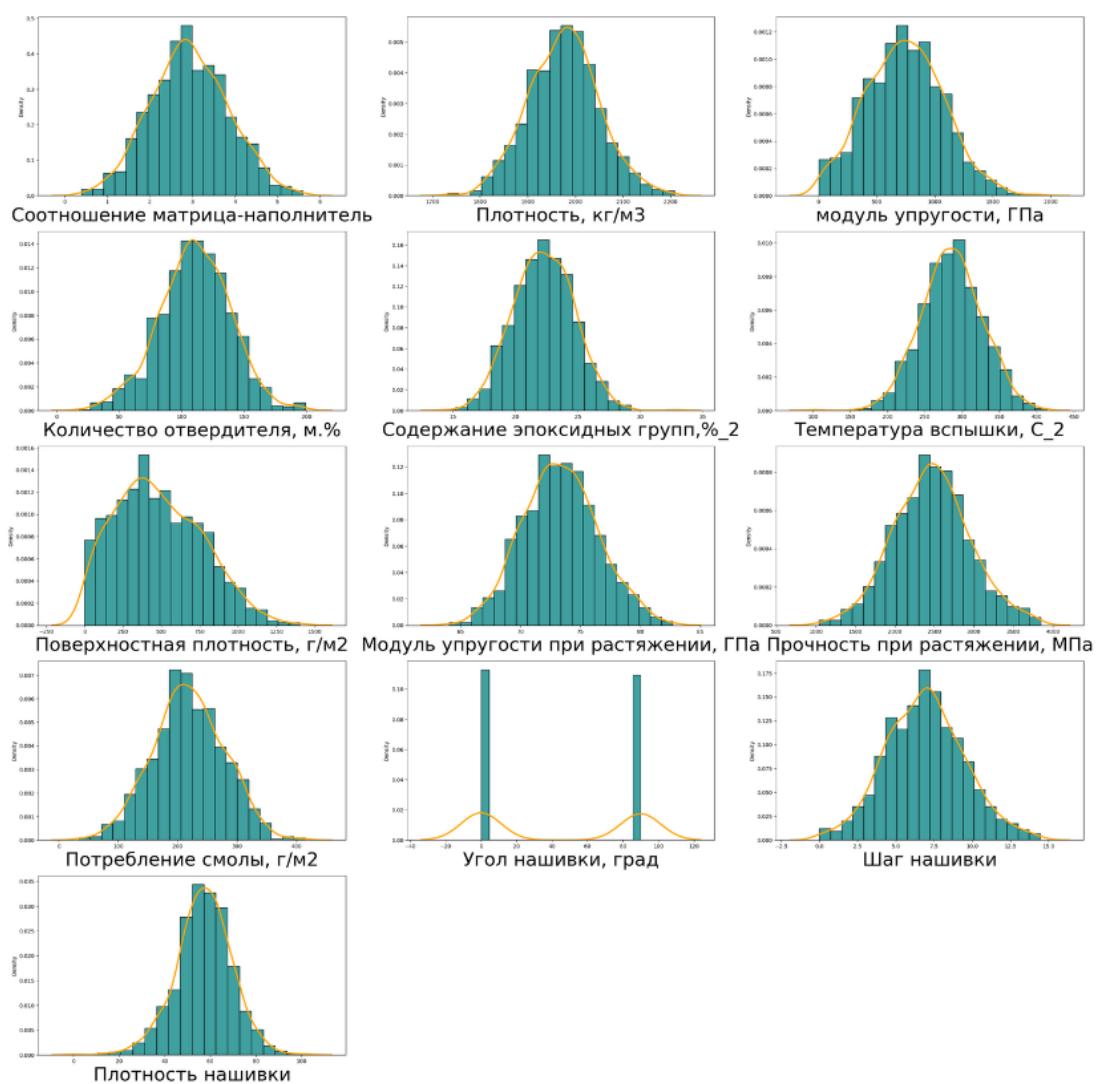


Рисунок 5 - Гистограммы распределения значений

Гистограмма распределения позволяет представить выборочные данные в графическом виде – в виде столбчатой диаграммы, где данные делятся на заранее определенное количество групп. Вид гистограммы дает наглядное представление функции плотности вероятности некоторой случайной величины, построенной по выборке.

На графиках оранжевым цветом представлена кривая стандартного нормального распределения (`kdeplot`), которая симметрична относительно среднего арифметического (Mean), медианы (Median) и моды (Mode). Более того, также являются нормальным распределением произведение двух нормальных распределений и их сумма.

Очевидно, что «Поверхностная плотность, г/м²» имеет явный скос значений от нормального распределения.

Проверяем данные на тестах:

- тест Шапиро-Уилка (Рисунок 6) — это статистический тест на нормальность. Это количественный метод проверки нормальности. Тест Шапиро-Уилка проверяет нулевую гипотезу о том, что данные были взяты из нормального распределения;
- тест Д'Агостино и Пирсона (Рисунок 7), этот метод использует асимметрию и эксцесс для проверки нормальности. Нулевая гипотеза для этого теста состоит в том, что распределение основано на нормальном распределении.

Тест Шапиро-Уилка.

Тест Шапиро-Уилка – это статистический тест на нормальность. Это количественный метод проверки нормальности. Тест Шапиро-Уилка проверяет нулевую гипотезу о том, что данные были взяты из нормального распределения.

```
: from scipy.stats import shapiro
shapiro_test = []
for i in df.columns:
    if shapiro(df[i])[1] < 0.05:
        shapiro_test.append(i)

print("Not Gaussian:", *shapiro_test, sep='\n')
```

Not Gaussian:
модуль упругости, ГПа
Поверхностная плотность, г/м²
угол нашивки, град
Плотность нашивки

Рисунок 6 - тест Шапиро-Уилка

Тест Д'Агостино и Пирсона.

Этот метод использует асимметрию и эксцесс для проверки нормальности. Нулевая гипотеза для этого теста состоит в том, что распределение основано на нормальном распределении.

```
from scipy.stats import normaltest
no_normtest = []
for i in df.columns:
    if normaltest(df[i].values)[1] < 0.05:
        no_normtest.append(i)

print("Not Gaussian:", *no_normtest, sep='\n')
```

Not Gaussian:
Поверхностная плотность, г/м²
Угол нашивки, град
Плотность нашивки

Рисунок 7 - тест Д'Агостино и Пирсона

Тест Шапиро-Уилка выдает четыре значения с отклонением от нормального распределения:

- «модуль упругости, ГПа»;
- «Поверхностная плотность, г/м²»;
- «Угол нашивки, град»;
- «Плотность нашивки».

Тест Д'Агостино и Пирсона указывает на 3 значения с отклонением от нормального распределения:

- «Поверхностная плотность, г/м²»;
- «Угол нашивки, град»;
- «Плотность нашивки».

В целом, отклонения от нормального распределения небольшие, поэтому в дальнейшем это никак не использовалось.

Построение графиков Q-Q («квантиль-квантиль») для проверки нормального распределения (Рисунок 8), которые используются для оценки того, потенциально ли набор данных получен из некоторого теоретического распределения. Если данные распределены нормально, то точки на графике QQ будут лежать на прямой диагональной линии.

Очевидным является отклонение от прямой диагональной линии в значениях «модуль упругости, ГПа», «Поверхностная плотность, г/м²», «Угол нашивки, град» (что ожидаемо, принимает 2 значения) и «Плотность нашивки».

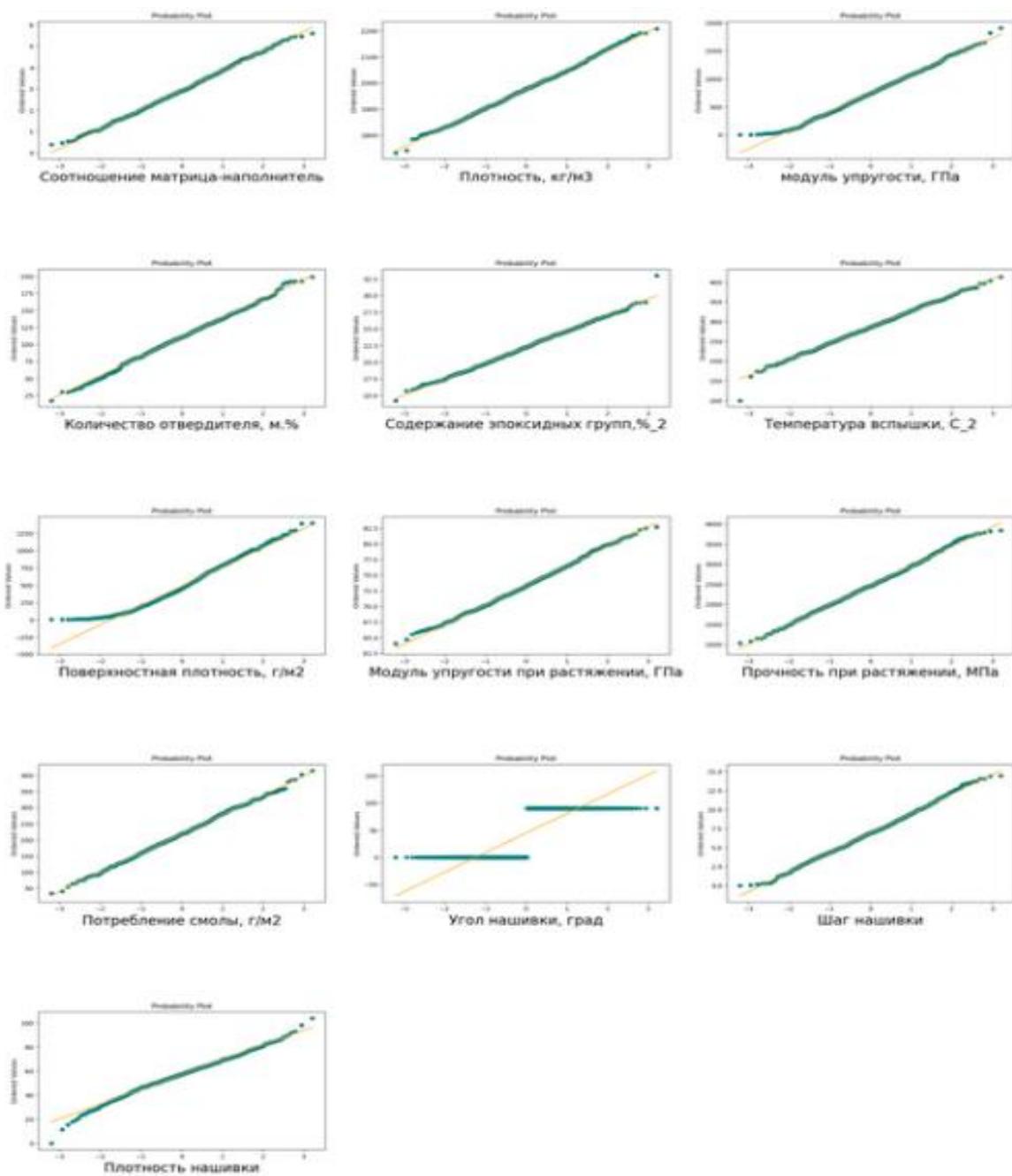


Рисунок 8 – Графики Q-Q

1.3.2 Исследование зависимостей в данных, корреляция

Корреляционная зависимость — статистическая взаимосвязь двух или более величин, при этом изменения значений одной или нескольких из этих величин сопутствуют систематическому изменению значений другой или других величин.

Тепловая карта представления корреляционной зависимости – это графическое представление квадратной таблицы, заголовками строк и столбцов которой являются обрабатываемые переменные, а на пересечении строк и столбцов выводятся коэффициенты корреляции для соответствующей пары признаков. Тепловая карта корреляции представлена на рисунке 9.

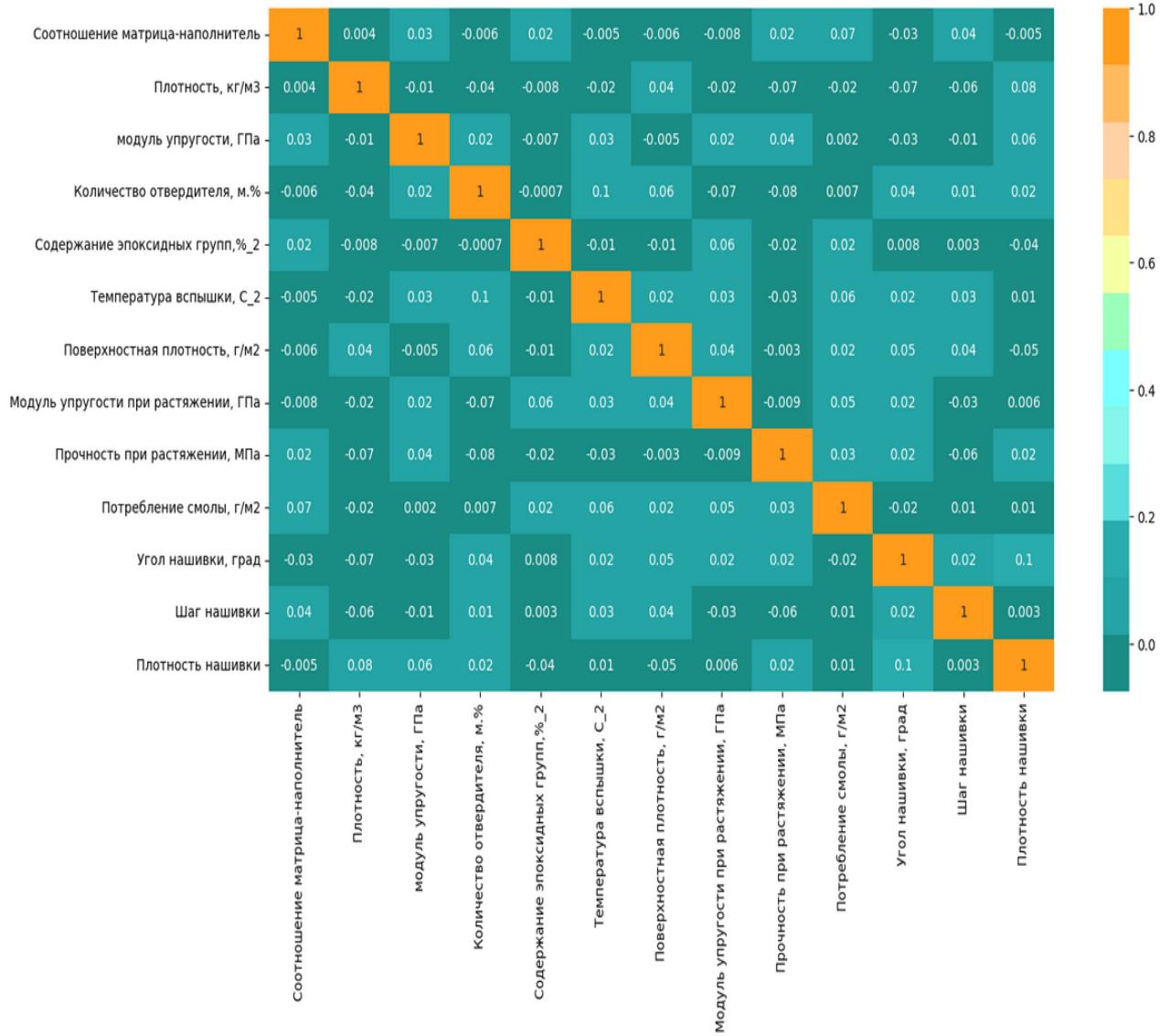


Рисунок 9 – Тепловая карта корреляции

Как видно из тепловой карты, корреляция между переменными очень слабая, близкая к 0. Максимальное значение зависимости наблюдается между переменными «Угол нашивки, град» и «Плотность нашивки» и составляет 0,1.

Построение парных диаграмм. Попарная диаграмма используется для сравнения распределения пар числовых переменных, создаёт сетку точечных

диаграмм (Рисунок 10). Она также содержит гистограмму для каждой функции в диагональных прямоугольниках.

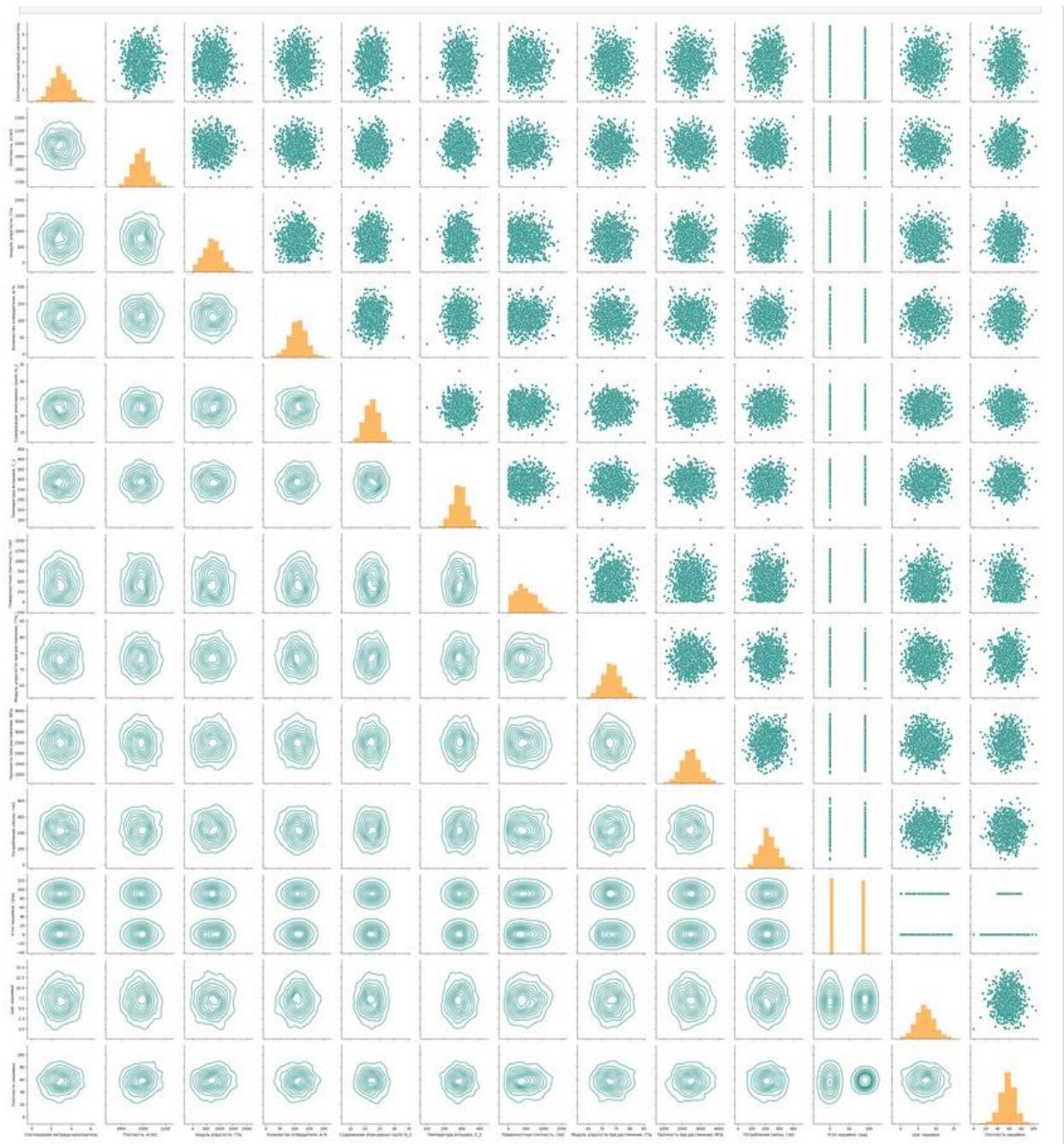


Рисунок 10 – парные диаграммы

1.3.3 Исследование выбросов в данных, удаление выбросов

Выброс — это наблюдение, которое лежит аномально далеко от других значений в наборе данных. Выбросы могут быть проблематичными, поскольку они могут повлиять на результаты анализа.

Boxplot — это диаграмма, которая показывает, как распределяются значения переменной. Он также известен как график "Ящик с усами" и дает информацию об изменчивости и дисперсии данных, используя сводку из пяти чисел. К ним относятся минимум, первый квартиль (Q1), медиана, третий квартиль (Q3) и максимум.

Проверка на выбросы. Визуализация с помощью диаграмм «Boxplot» (Рисунок 11). В данных присутствуют выбросы.

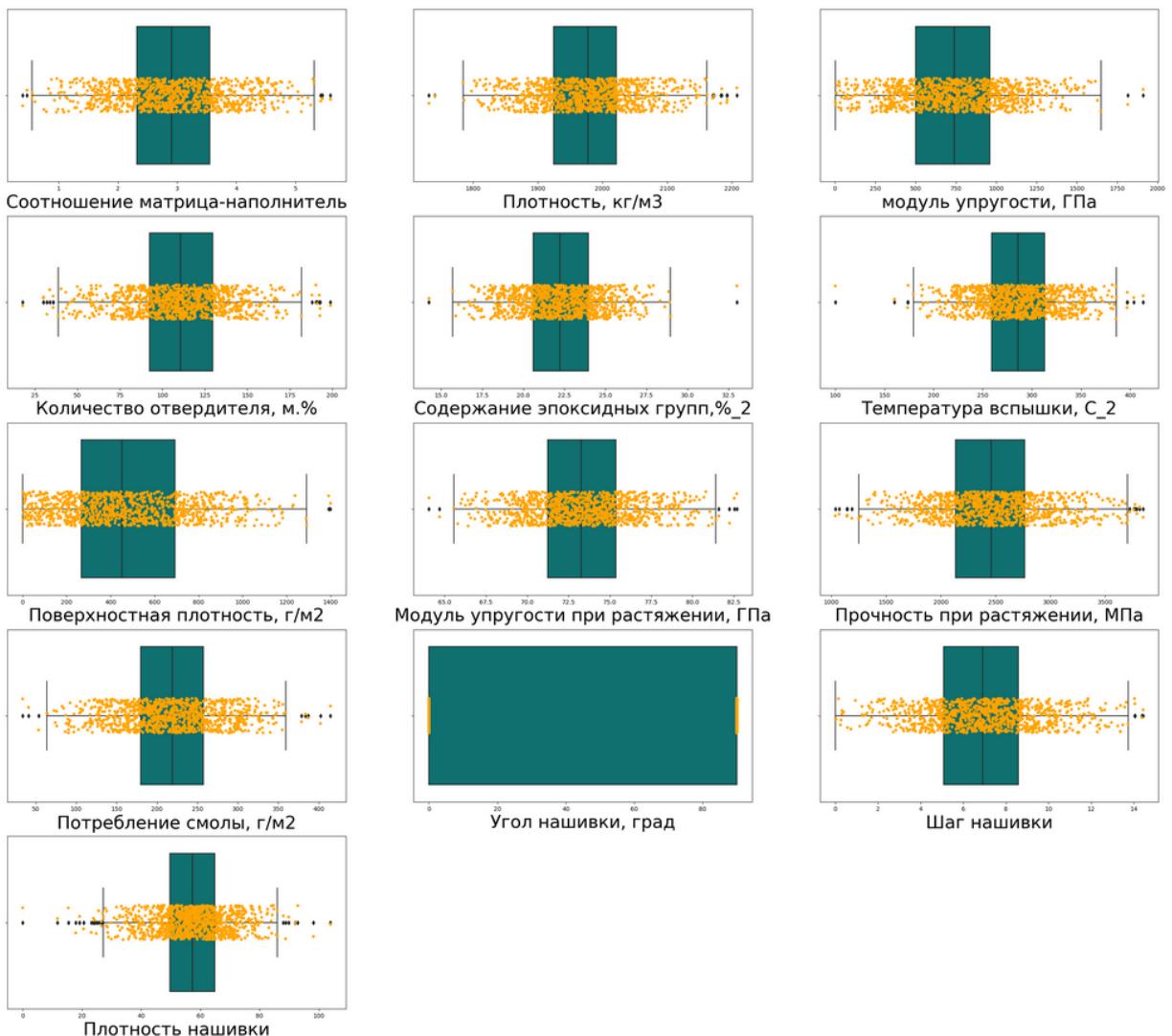


Рисунок 11— Диаграммы Boxplot

Довольно распространенным способом выявления и удаления выбросов является правило межквартильного $1.5 * \text{IQR}$. В стандартном нормальном распределении Q1 и Q3 соответствуют -0.6745 и 0.6745.

После удаления выбросов из датасета у нас осталось 922 значения и 13 признаков. Boxplot без выбросов представлен на рисунке 12.

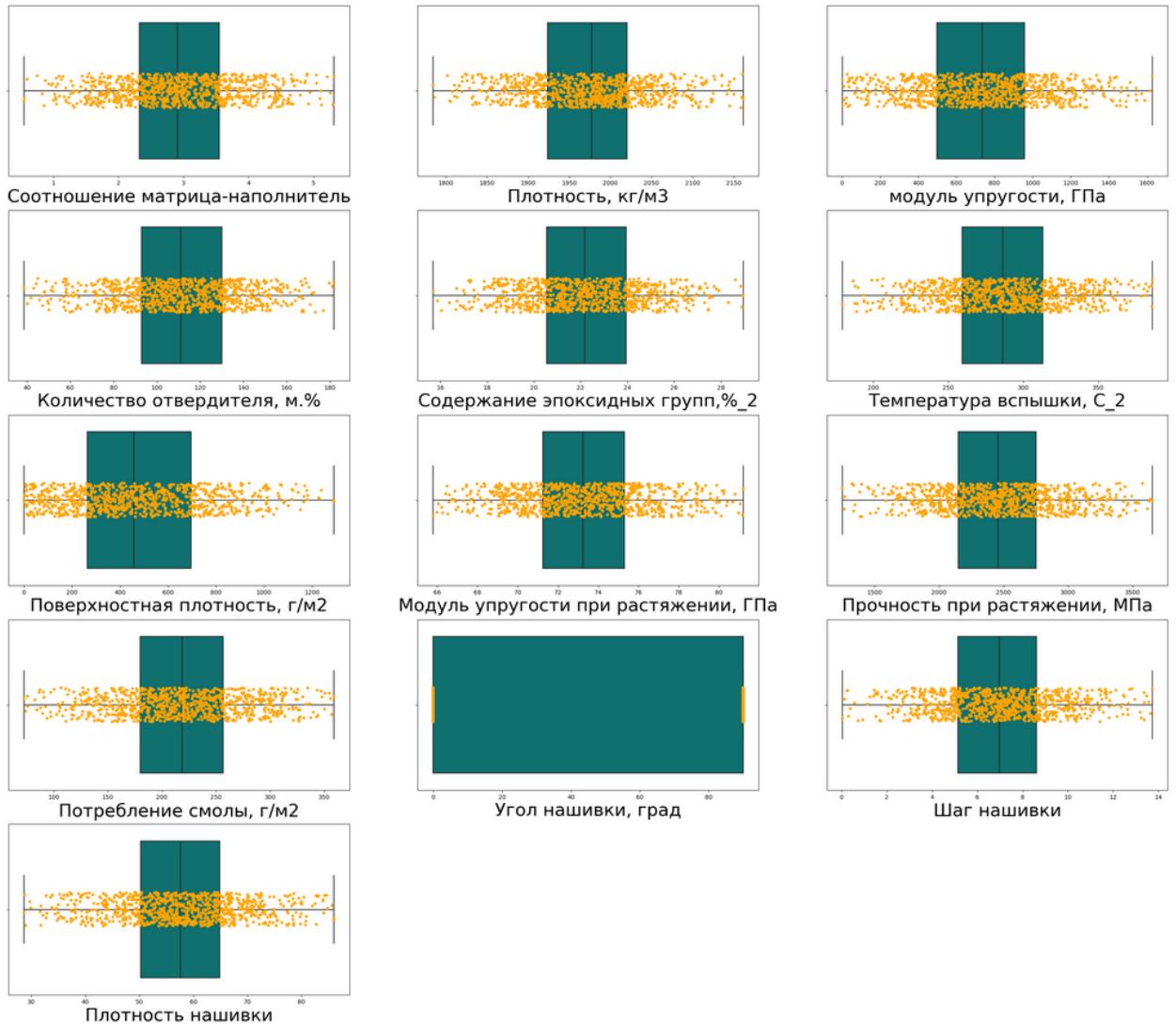


Рисунок 12 – графики Boxplot данных, очищенных от выбросов

Описательная статистика для очищенных от выбросов данных представлена в таблице на рисунке 13.

	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	922.00	922.00	922.00	922.00	922.00	922.00	922.00	922.00	922.00	922.00	922.00	922.00	922.00
mean	2.93	1974.12	736.12	111.14	22.20	286.18	482.43	73.30	2461.49	218.05	45.98	6.93	57.56
std	0.90	71.04	327.61	26.75	2.39	39.42	280.44	3.03	453.56	57.14	45.01	2.51	11.12
min	0.55	1784.48	2.44	38.67	15.70	179.37	0.60	65.79	1250.39	72.53	0.00	0.04	28.66
25%	2.32	1923.32	498.54	92.86	20.56	259.21	264.35	71.24	2148.18	179.88	0.00	5.14	50.28
50%	2.91	1977.32	736.18	111.16	22.18	286.22	457.73	73.25	2455.97	218.70	90.00	6.97	57.58
75%	3.55	2020.05	956.96	130.11	23.96	313.01	695.53	75.31	2751.23	256.62	90.00	8.61	64.84
max	5.31	2161.57	1628.00	181.83	28.96	386.07	1291.34	81.20	3654.43	359.05	90.00	13.73	86.01

Рисунок 13 – описательная статистика для очищенных данных

Повторно построим парные диаграммы (Рисунок 14) для выявления зависимостей.

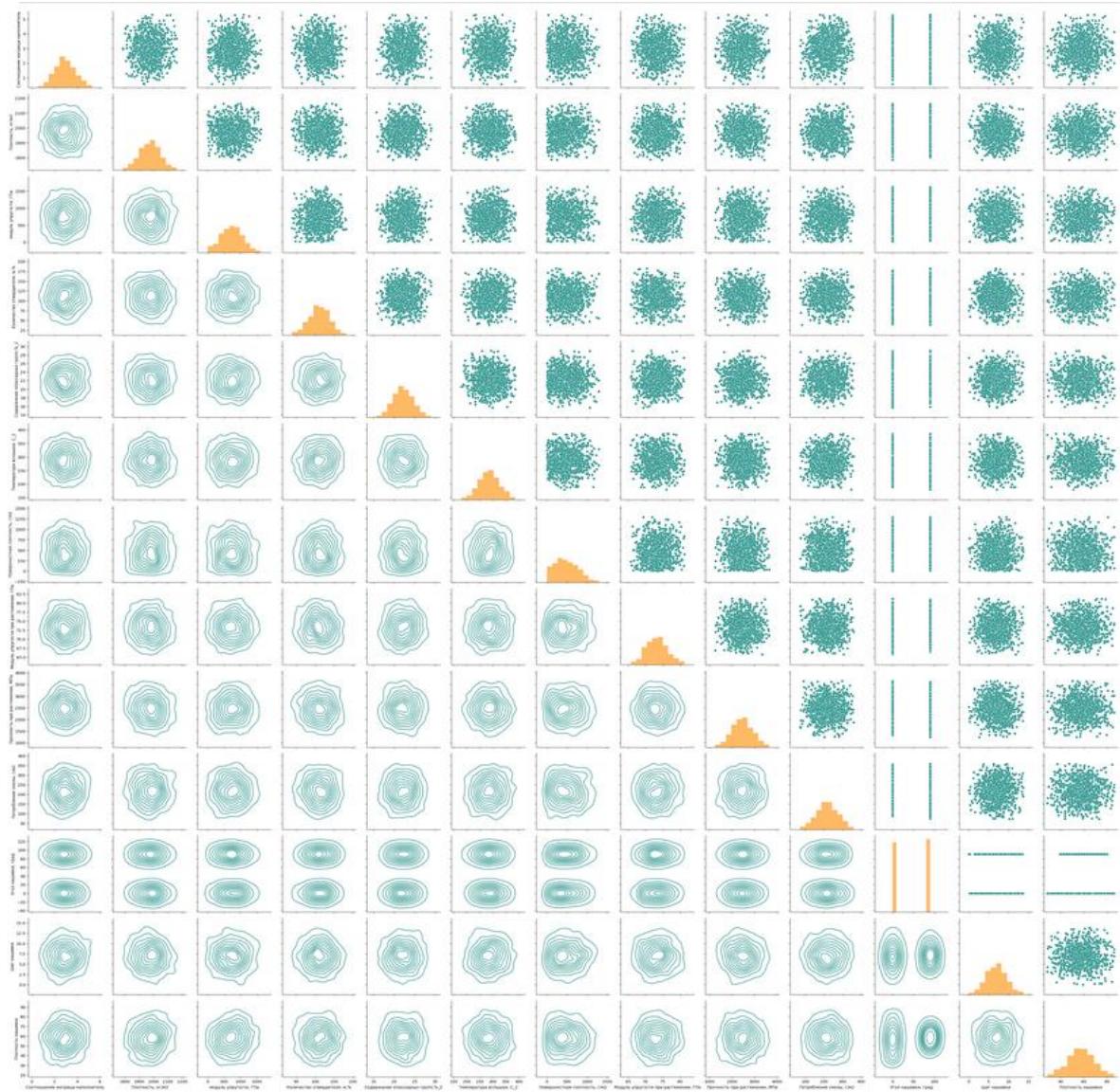


Рисунок 14 – парные диаграммы очищенных данных

Зависимостей не выявлено.

Построим тепловую карту (Рисунок 15) и посмотрим на коэффициенты корреляции на очищенных данных.

Корреляция стала еще хуже. Теперь лучшее значение коэффициента корреляции наблюдается между «Плотность нашивки» и «Плотность, кг/м3» и составляет 0,09.

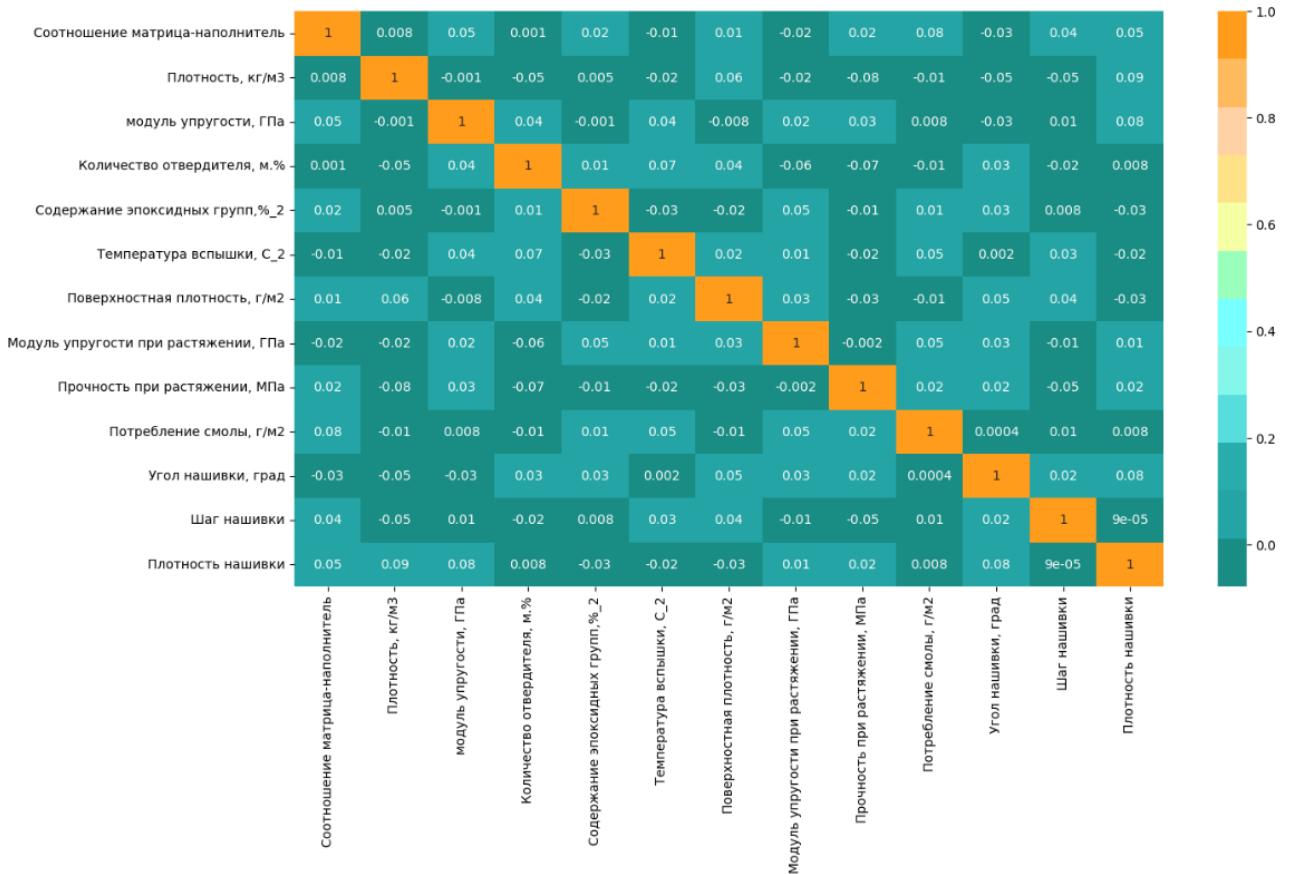


Рисунок 15 – тепловая карта взаимосвязи очищенных данных

1.3.4 Ход решения задачи

Ход решения каждой из задач и построения оптимальной модели будет следующим:

- разделить данные на тренировочную и тестовую выборки. В задании указано, что на тестирование оставить 30% данных;
- выполнить препроцессинг, то есть подготовку исходных данных;
- подобрать для моделей гиперпараметры с помощью с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10;
- сравнить метрики моделей после подбора гиперпараметров и выбрать лучшую;
- получить предсказания лучшей и базовой моделей на тестовой выборке, сделать выводы;
- сравнить качество работы лучшей модели.

1.3.5 Препроцессинг

Цель препроцессинга, или предварительной обработки данных — обеспечить корректную работу моделей.

Проблема вещественных признаков в том, что их значения лежат в разных диапазонах и в разных масштабах. Подадим на выбор модели четыре scaler-а, а модель сама выберет для себя лучший способ нормализации и стандартизации:

- MinMaxScaler – библиотека Scikit-learn, позволяющая произвести нормализацию (Normalization) данных перед использованием в модели (Model) машинного обучения (ML), то есть приведение числовых переменных (Numeric Variable) к диапазону от 0 до 1. Реализуется с помощью функции `sklearn.preprocessing.MinMaxScaler()`;
- StandardScaler - Стандартизация (Standartization) масштабирует каждую входную переменную отдельно путем вычитания среднего значения (так называемое центрирование) и деления на Стандартное отклонение (Standard Deviation), чтобы Среднее значение (Average) равнялось нулю, а стандартное отклонение – единице. Стандартизация выполняется с помощью функции `sklearn.preprocessing.StandardScaler()`;
- MaxAbsScaler - при такой нормировке каждый признак масштабируется по своему максимальному абсолютному значению, тем самым диапазон изменения каждого признака трансформируется в диапазон [-1; 1]. Реализуется с помощью `sklearn.preprocessing.MaxAbsScaler()`;
- RobustScaler - функция масштабирования с использованием статистики, которая устойчива к отклонениям. Этот Scaler удаляет медиану и масштабирует данные в соответствии с квантильным диапазоном (по умолчанию IQR). Центрирование и масштабирование происходят независимо для каждой функции путем вычисления соответствующей статистики по выборкам в обучающем наборе. Затем медиана и межквартильный размах сохраняются для использования в последующих данных с помощью метода `transform`. Выполняется с помощью функции `sklearn.preprocessing.RobustScaler()`.

Препроцессинг необходимо повторить в приложении для введенных данных.

Выходные переменные никак не изменяю.

1.3.6 Перекрестная проверка

Для обеспечения статистической устойчивости метрик модели используем перекрестную проверку или кросс-валидацию. Чтобы ее реализовать, выборка разбивается необходимое количество раз на тестовую и валидационную. Модель обучается на тестовой выборке, затем выполняется расчет метрик качества на валидационной. В качестве результата мы получаем средние метрики качества для всех валидационных выборок. Перекрестную проверку реализует функция `cross_validate` из `sklearn`.

1.3.7 Поиск гиперпараметров по сетке, случайный поиск

`GridSearchCV` - поиск гиперпараметров по сетке реализует класс из `sklearn`. При выполнении оптимизации гиперпараметров сначала необходимо определить область или сетку параметров, в которую мы включаем набор возможных значений гиперпараметров, которые могут быть использованы для построения модели. Затем используется метод поиска по сетке для размещения этих гиперпараметров в матрице, и модель обучается на каждой комбинации значений гиперпараметров. После этого выбирается модель с наилучшими показателями. Перекрестная проверка уже встроена в этот класс.

`RandomizedSearchCV` - случайный поиск. В то время как поиск по сетке рассматривает все возможные комбинации гиперпараметров, чтобы найти лучшую модель, случайный поиск выбирает и тестирует только случайную комбинацию гиперпараметров. Вместо исчерпывающего поиска делает случайную выборку из сетки гиперпараметров. Мы указываем общее количество

попыток случайного поиска, прежде чем будет получена наилучшая модель. Перекрестная проверка уже встроена в этот класс.

1.3.8 Метрики качества моделей

Существует множество различных метрик качества, применимых для регрессии. В этой работе я использую:

- R² или коэффициент детерминации измеряет долю дисперсии, объясненную моделью, в общей дисперсии целевой переменной. Если он близок к единице, то модель хорошо объясняет данные, если же он близок к нулю, то прогнозы сопоставимы по качеству с константным предсказанием;
- MSE (Mean Squared Error) средняя квадратичная ошибка, принимает значениях в тех же единицах, что и целевая переменная. Метрика использует возведение в квадрат, поэтому хорошо обнаруживает грубые ошибки, но сильно чувствительна к выбросам;
- RMSE (Root Mean Squared Error) или корень из средней квадратичной ошибки;
- MAE (Mean Absolute Error) - средняя абсолютная ошибка так же принимает значениях в тех же единицах, что и целевая переменная;
- MAPE (Mean Absolute Percentage Error) или средняя абсолютная процентная ошибка — безразмерный показатель, представляющий собой взвешенную версию MAE.

2 Практическая часть

2.1 Разбиение и предобработка данных

Выделим из датасета целевые переменные (Рисунок 16) и признаки, на которых будем проводить обучение.

```
X_no_norm = data[['Соотношение матрица-наполнитель', 'Плотность, кг/м3',
                   'модуль упругости, ГПа', 'Количество отвердителя, м.%',
                   'Содержание эпоксидных групп,%_2', 'Температура вспышки, С_2',
                   'Поверхностная плотность, г/м2',
                   'Потребление смолы, г/м2', 'Угол нашивки, град',
                   'Шаг нашивки', 'Плотность нашивки']]  
  
y_1 = data['Модуль упругости при растяжении, ГПа']
y_2 = data['Прочность при растяжении, МПа']
```

Посмотрим размерность.

```
X_no_norm.shape, y_1.shape, y_2.shape
```

```
((922, 11), (922,), (922,))
```

Рисунок 16 – Выделение целевых признаков

Размерности полученных наборов данных для «Модуля упругости при растяжении, ГПа» показаны на рисунке 17, для «Прочности при растяжении, МПа» - на рисунке 18. Выделение целевого признака для рекомендации «Соотношения матрица-наполнитель» представлено на рисунке 19, размерность полученного набора – рисунок 20.

```
x_train, x_test, y_train, y_test = train_test_split(X_no_norm, y_1,
                                                    test_size=0.3,
                                                    shuffle=True,
                                                    random_state=18)
```

```
# размерность обучающей
print(x_train.shape, y_train.shape)

# и тестовой выборки
print(x_test.shape, y_test.shape)
```

```
(645, 11) (645,)
(277, 11) (277,)
```

Рисунок 17 – размерность обучающей и тестовой выборки для

«Модуля упругости при растяжении, ГПа»

```
x_train, x_test, y_train, y_test = train_test_split(x_no_norm, y_2,  
                                                 test_size=0.3,  
                                                 shuffle=True,  
                                                 random_state=18)
```

```
# размерность обучающей  
print(x_train.shape, y_train.shape)  
  
# и тестовой выборки  
print(x_test.shape, y_test.shape)
```

(645, 11) (645,)
(277, 11) (277,)

Рисунок 18 – размерность обучающей и тестовой выборки для
«Прочности при растяжении, МПа»

```
X_no_norm = data[['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']]  
y = data['Соотношение матрица-наполнитель']
```

```
X_no_norm.shape, y.shape
```

((922, 12), (922,))

Рисунок 19 – выделение целевой переменной для рекомендации
«Соотношения матрица-наполнитель»

```
x_train, x_test, y_train, y_test = train_test_split(x, y,  
                                                 test_size=0.3,  
                                                 shuffle=True,  
                                                 random_state=18)
```

```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

((645, 12), (645,), (277, 12), (277,))

Рисунок 20 – размерность обучающей и тестовой выборки для
«Соотношения матрица-наполнитель»

2.2 Разработка и обучение модели

2.2.1 Разработка и обучение модели для «Модуля упругости при растяжении, ГПа»

Для подбора лучшей модели для этой задачи я взяла следующие модели:

a) LinearRegression — линейная регрессия (раздел 1.2.1)

Лучшие параметры поиска по сетке представлены на рисунке 21, результат работы модели показан на рисунке 22.

```
y_pred_lr = grid_search(pipe_lr, linreg_params)
{'lr__fit_intercept': 'True', 'scaler': MinMaxScaler()}
-0.026739052457467105
```

Рисунок 21 – лучшие параметры работы модели

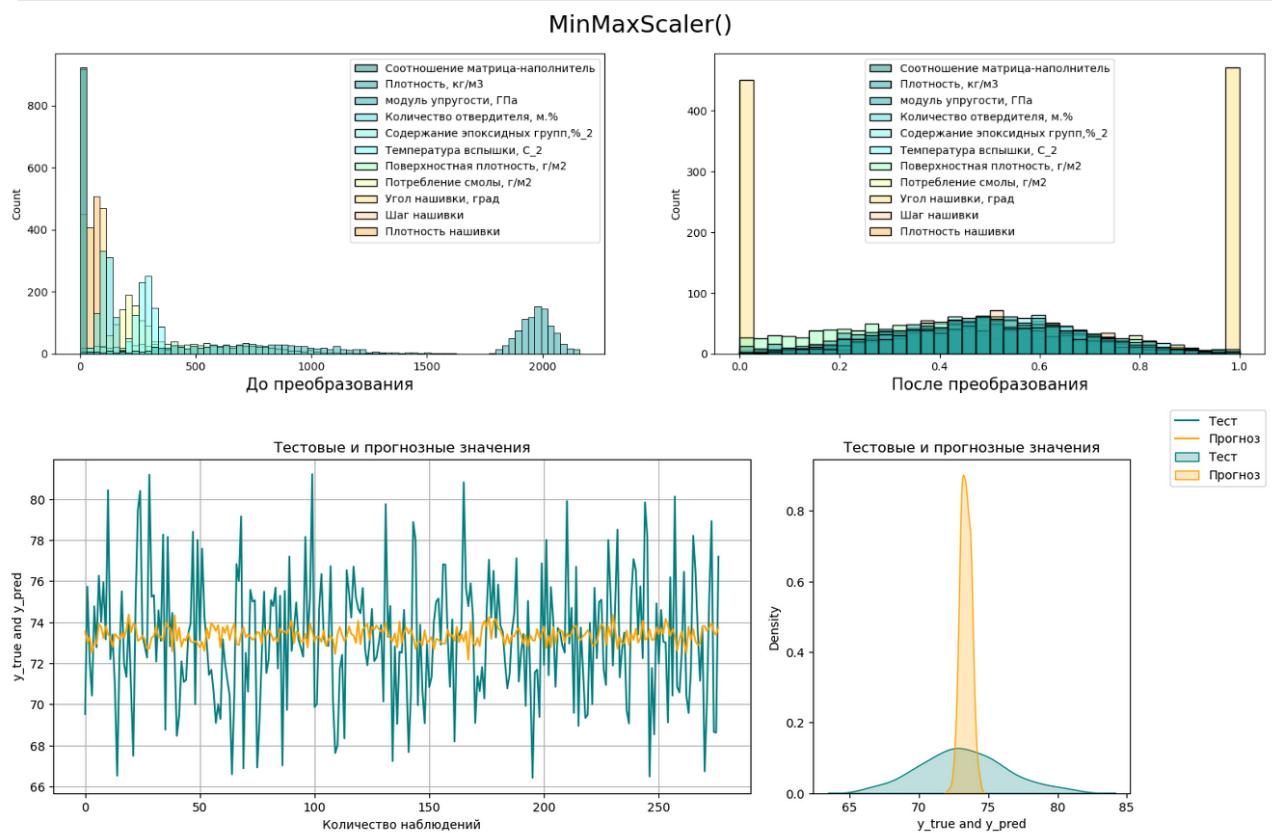


Рисунок 22 – результат работы модели LinearRegression

б) KneighborsRegressor — метод ближайших соседей (раздел 1.2.2)

Лучшие параметры поиска по сетке представлены на рисунке 23.

```

y_pred_rnn = grid_search(pipe_knn, knn_params)

{'knn_algorithm': 'auto', 'knn_n_neighbors': 43, 'knn_weights': 'uniform', 'scaler': MinMaxScaler()}
-0.011869076356469055

```

Рисунок 23 – лучшие параметры работы модели KneighborsRegressor

Как видно на рисунке 23, наилучших результатов удалось достичнуть при применении нормализатора MinMaxScaler. Посмотрим графическое представление работы нормализатора, а так же результат предсказания модели KneighborsRegressor (Рисунок 24).

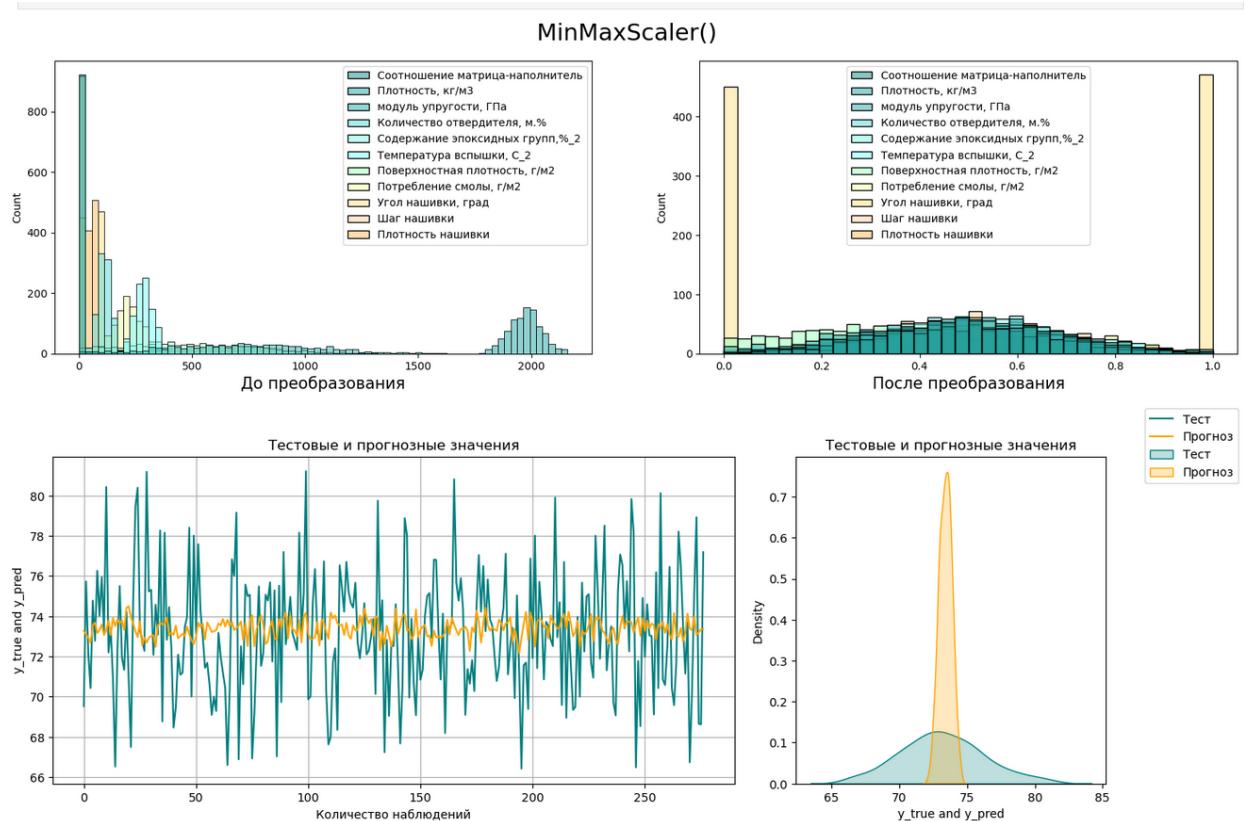


Рисунок 24 - результат работы модели KneighborsRegressor

в) RandomForestRegressor — случайный лес (раздел 1.2.3)

Лучшие параметры случайного поиска по сетке представлены на рисунке 25.

```

: y_pred_rf = random_search(pipe_rf, rf_params)

Fitting 10 folds for each of 10 candidates, totalling 100 fits
{'scaler': MaxAbsScaler(), 'rf_n_estimators': 460, 'rf_max_features': 'sqrt', 'rf_max_depth': 61, 'rf_criterion': 'absolute_error'}
-0.05095938924256127

```

Рисунок 25 - лучшие параметры работы модели RandomForestRegressor

Как видно на рисунке 25, наилучших результатов удалось достичнуть при применении MaxAbsScaler. Посмотрим графическое представление работы MaxAbsScaler, а так же результат предсказания модели RandomForestRegressor (Рисунок 26).

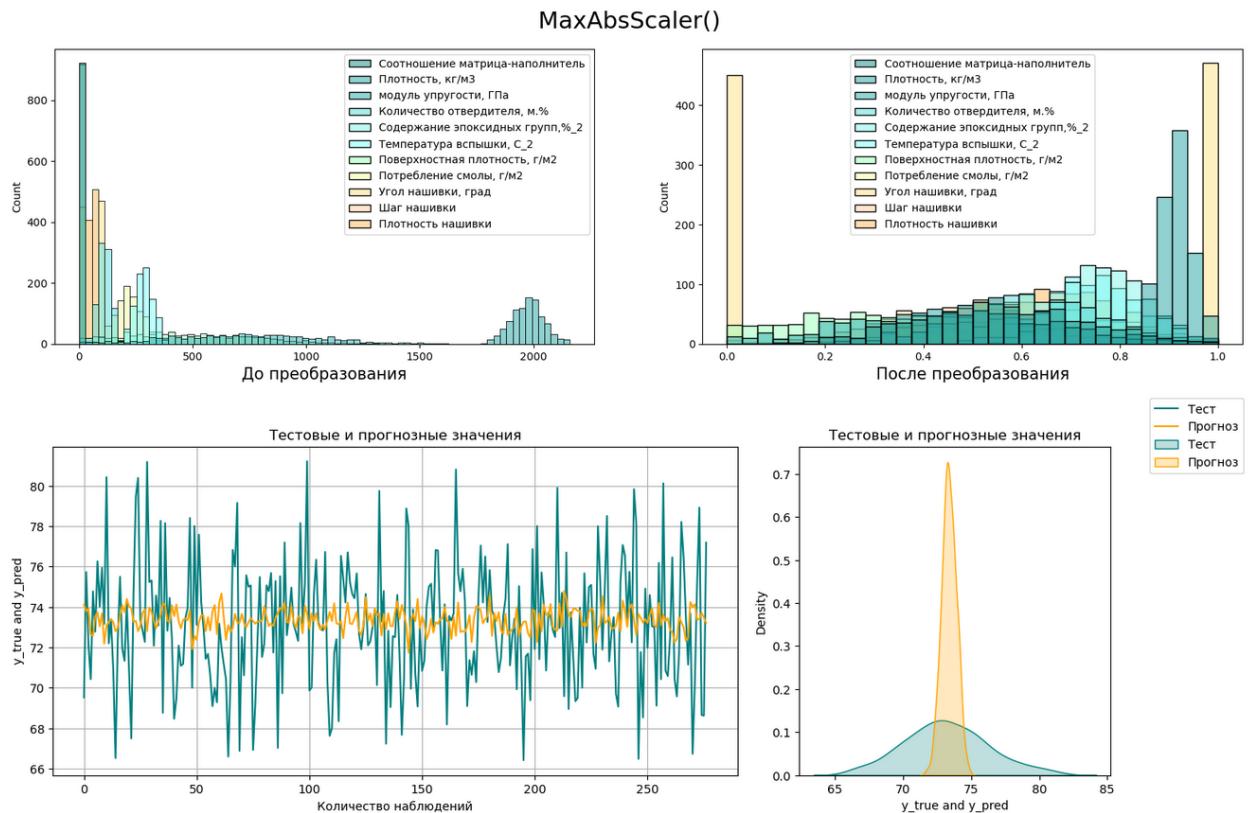


Рисунок 26 - результат работы модели RandomForestRegressor

г) SVR — метод опорных векторов (раздел 1.2.4)

Лучшие параметры случайного поиска по сетке представлены на рисунке 27.

```

: y_pred_svr = random_search(pipe_svr, svr_params)

Fitting 10 folds for each of 10 candidates, totalling 100 fits
{'svr_kernel': 'sigmoid', 'svr_gamma': 1, 'svr_C': 1, 'scaler': MaxAbsScaler()}
-0.00571213083781652

```

Рисунок 27 - лучшие параметры работы модели SVR

Как видно на рисунке 27, наилучших результатов удалось достичнуть при применении MaxAbsScaler. Посмотрим графическое представление работы MaxAbsScaler, а так же результат предсказания модели SVR (Рисунок 28).

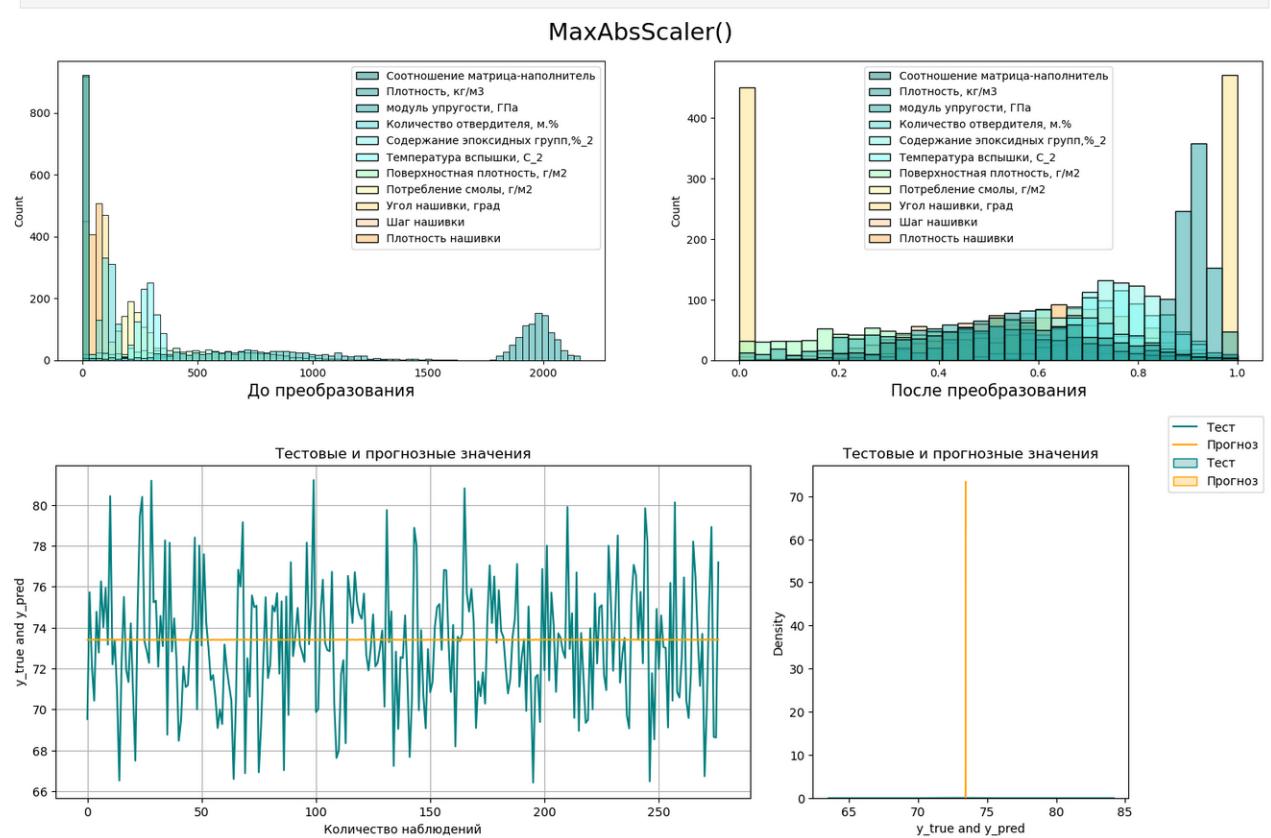


Рисунок 28 - результат работы модели SVR

д) GradientBoostingRegressor – градиентный бустинг (раздел 1.2.5)

Лучшие параметры случайного поиска по сетке представлены на рисунке 29.

```
|: y_pred_gbr = random_search(pipe_gbr, gbr_params)

Fitting 10 folds for each of 10 candidates, totalling 100 fits
{'scaler': StandardScaler(), 'gbr__subsample': 0.2, 'gbr__n_estimators': 100, 'gbr__max_depth': 4, 'gbr__learning_rate': 0.02}
-0.03972297173025119
```

Рисунок 29 - лучшие параметры работы модели GradientBoostingRegressor

Как видно на рисунке 29, наилучших результатов удалось достичнуть при применении стандартизатора StandardScaler. Посмотрим графическое

представление работы StandardScaler, а так же результат предсказания модели GradientBoostingRegressor (Рисунок 30).

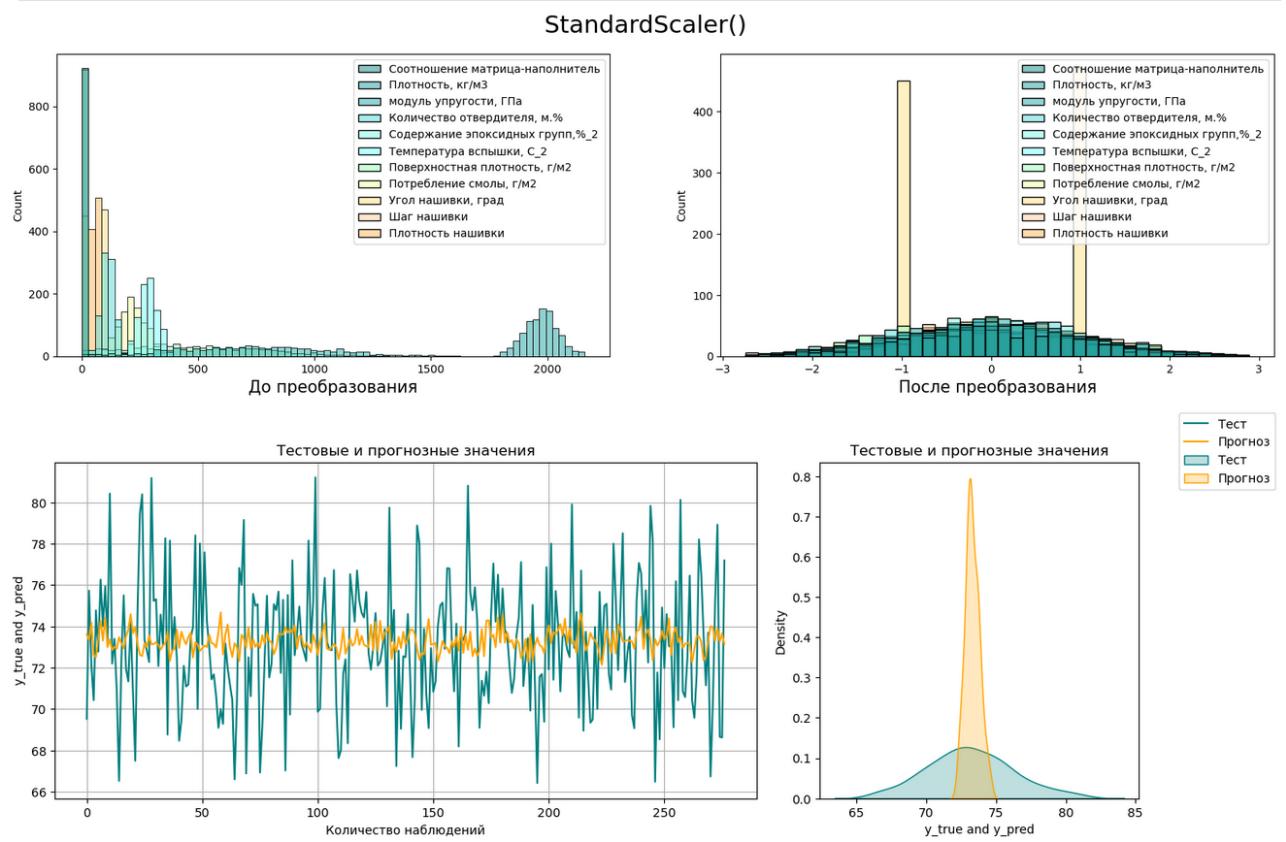


Рисунок 30 - результат работы модели GradientBoostingRegressor

е) Полносвязная нейронная сеть

MinMaxScaler показал лучший R² = 0.009. График обучения модели показан на рисунке 31, а результат работы нейронной сети – на рисунке 32.

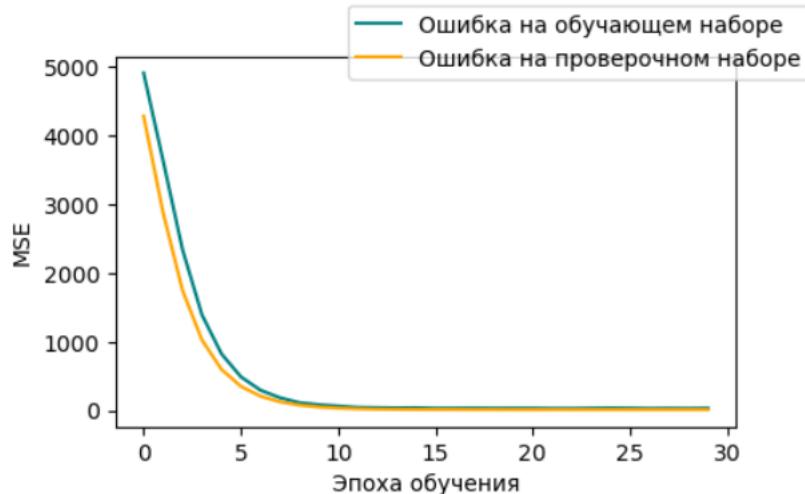


Рисунок 31 – график обучения нейронной сети

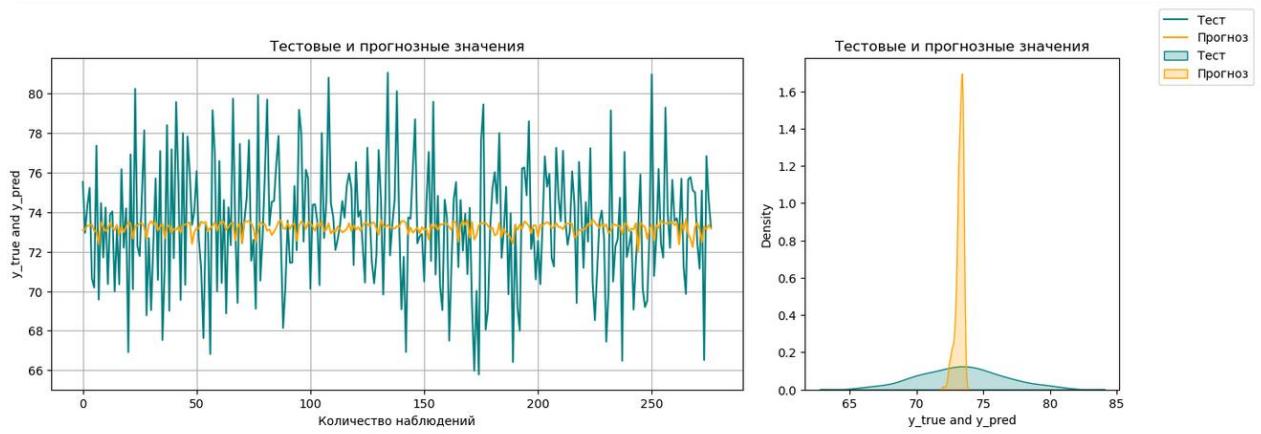


Рисунок 32 - результат работы нейронной сети с нормализатором MinMaxScaler

2.2.2 Разработка и обучение модели для «Прочности при растяжении, МПа»

Для подбора лучшей модели для этой задачи я взяла следующие модели:

- a) LinearRegression — линейная регрессия (раздел 1.2.1)

Лучшие параметры поиска по сетке представлены на рисунке 33, результат работы модели показан на рисунке 34.

```
y_pred_lr = grid_search(pipe_lr, linreg_params)
{'lr_fit_intercept': 'True', 'scaler': MinMaxScaler()}
-0.014821905358583853
```

Рисунок 33 – лучшие параметры работы модели

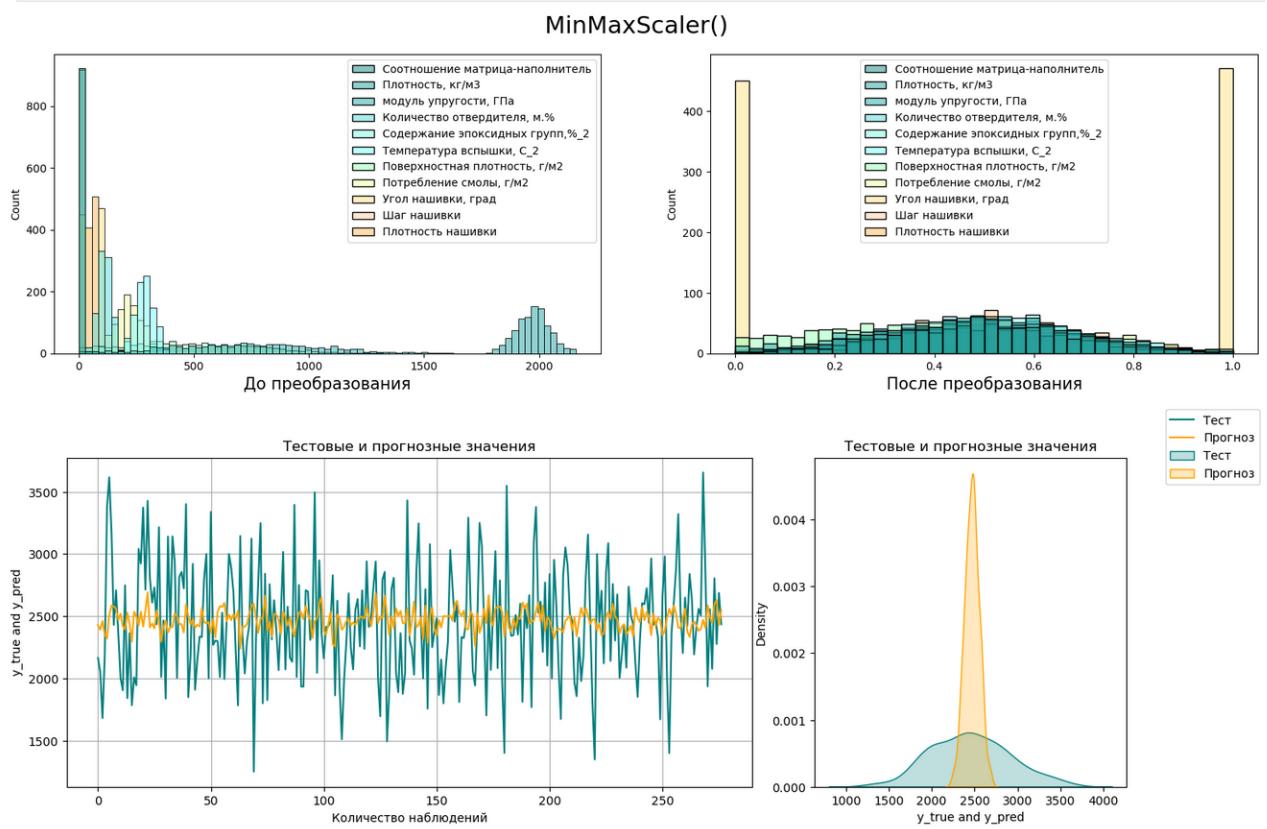


Рисунок 34 – результат работы модели LinearRegression

б) KneighborsRegressor — метод ближайших соседей (раздел 1.2.2)

Лучшие параметры поиска по сетке представлены на рисунке 35.

```
y_pred_rnn = grid_search(pipe_knn, knn_params)

{'knn_algorithm': 'auto', 'knn_n_neighbors': 47, 'knn_weights': 'uniform', 'scaler': MaxAbsScaler()}

0.006789149317663834
```

Рисунок 35 – лучшие параметры работы модели KneighborsRegressor

Как видно на рисунке 35, наилучших результатов удалось достигнуть при применении нормализатора MaxAbsScaler. Посмотрим графическое представление работы scaler, а так же результат предсказания модели KneighborsRegressor (Рисунок 36).

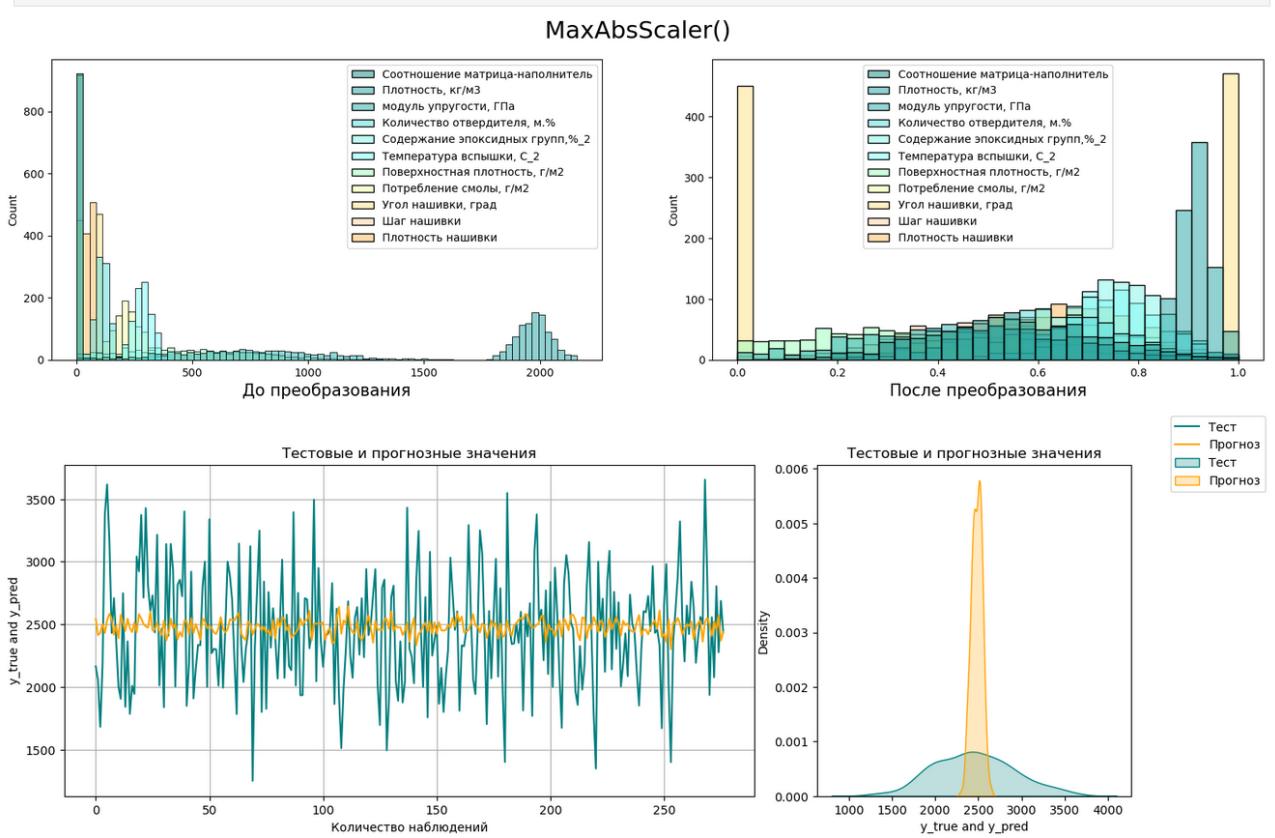


Рисунок 36 - результат работы модели KneighborsRegressor

в) RandomForestRegressor — случайный лес (раздел 1.2.3)

Лучшие параметры случайного поиска по сетке представлены на рисунке 37.

```
y_pred_rf = random_search(pipe_rf, rf_params)

Fitting 10 folds for each of 10 candidates, totalling 100 fits
{'scaler': MinMaxScaler(), 'rf_n_estimators': 490, 'rf_max_features': 'sqrt', 'rf_max_depth': 61, 'rf_criterion': 'squared_error'}
0.009060389832234117
```

Рисунок 37 - лучшие параметры работы модели RandomForestRegressor

Как видно на рисунке 37, наилучших результатов удалось достичнуть при применении нормализатора MinMaxScaler. Посмотрим графическое представление работы MinMaxScaler, а также результат предсказания модели RandomForestRegressor (Рисунок 38).

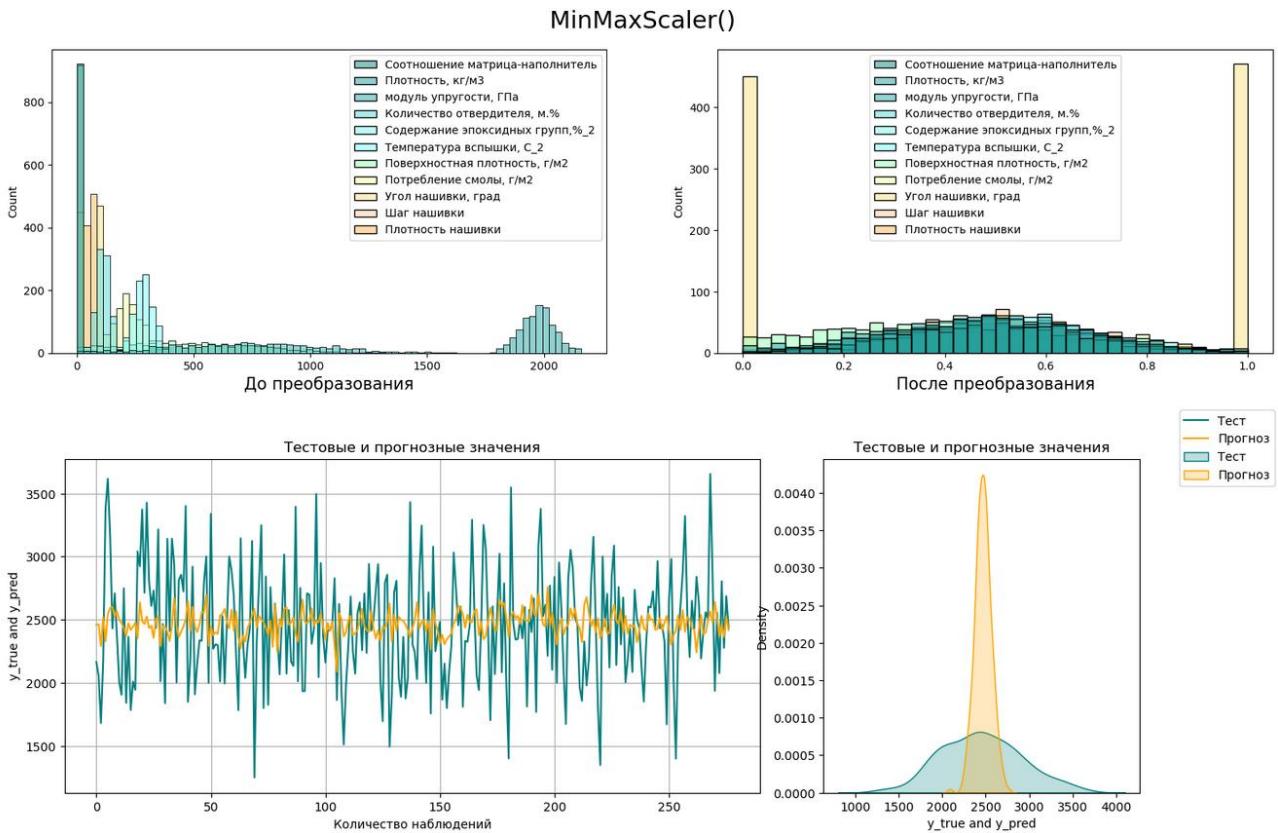


Рисунок 38 - результат работы модели RandomForestRegressor

г) SVR — метод опорных векторов (раздел 1.2.4)

Лучшие параметры случайного поиска по сетке представлены на рисунке 39.

```
y_pred_svr = random_search(pipe_svr, svr_params)

Fitting 10 folds for each of 10 candidates, totalling 100 fits
{'svr_kernel': 'sigmoid', 'svr_gamma': 0.01, 'svr_C': 41, 'scaler': RobustScaler()}
0.0013613777431288332
```

Рисунок 39 - лучшие параметры работы модели SVR

Как видно на рисунке 39, наилучших результатов удалось достичнуть при применении RobustScaler. Посмотрим графическое представление работы RobustScaler, а также результат предсказания модели SVR (Рисунок 40).

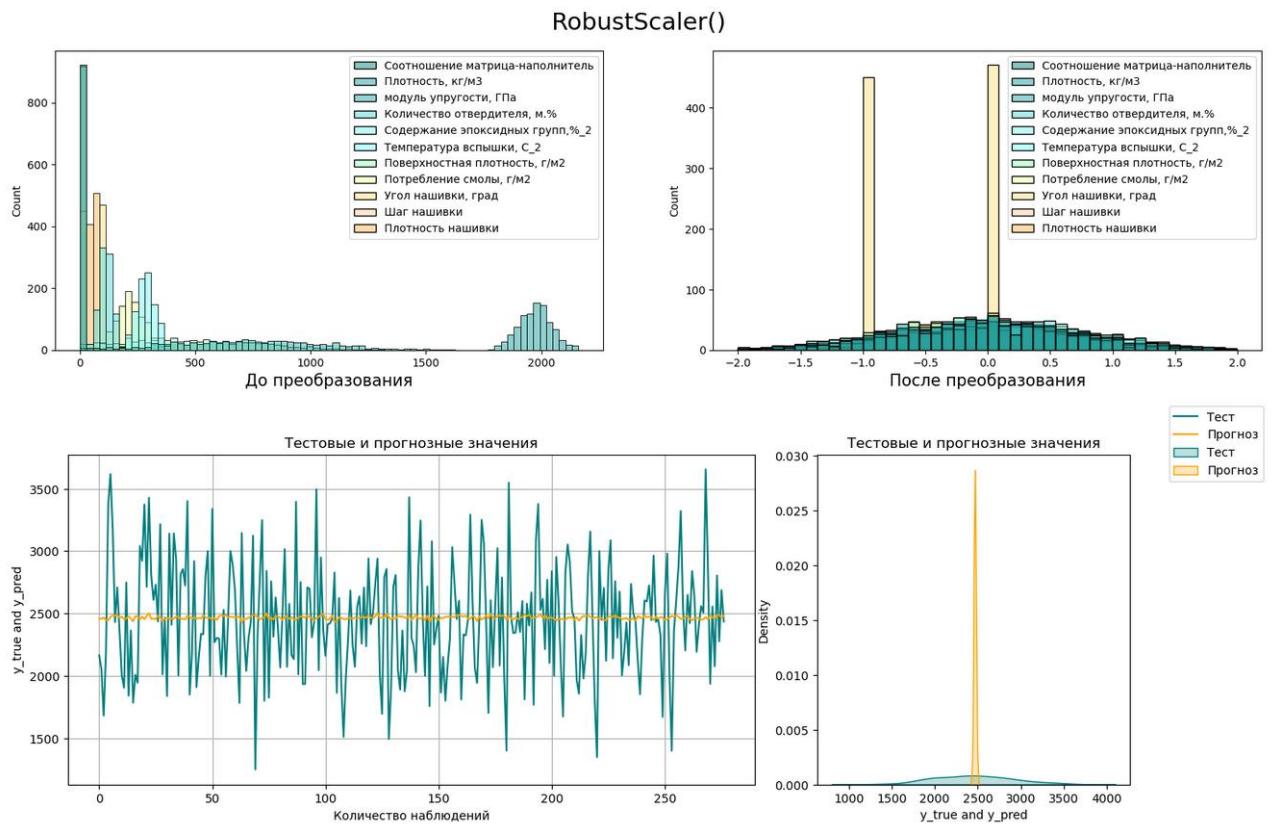


Рисунок 40 - результат работы модели SVR

д) GradientBoostingRegressor – градиентный бустинг (раздел 1.2.5)

Лучшие параметры случайного поиска по сетке представлены на рисунке 41.

```
... y_pred_gbr = random_search(pipe_gbr, gbr_params)

Fitting 10 folds for each of 10 candidates, totalling 100 fits
{'scaler': StandardScaler(), 'gbr__subsample': 0.2, 'gbr__n_estimators': 100, 'gbr__max_depth': 4, 'gbr__learning_rate': 0.02}
-0.018147496274878216
```

Рисунок 41 - лучшие параметры работы модели GradientBoostingRegressor

Как видно на рисунке 41, наилучших результатов удалось достичнуть при применении стандартизатора StandardScaler. Посмотрим графическое представление работы StandardScaler, а также результат предсказания модели GradientBoostingRegressor (Рисунок 42).

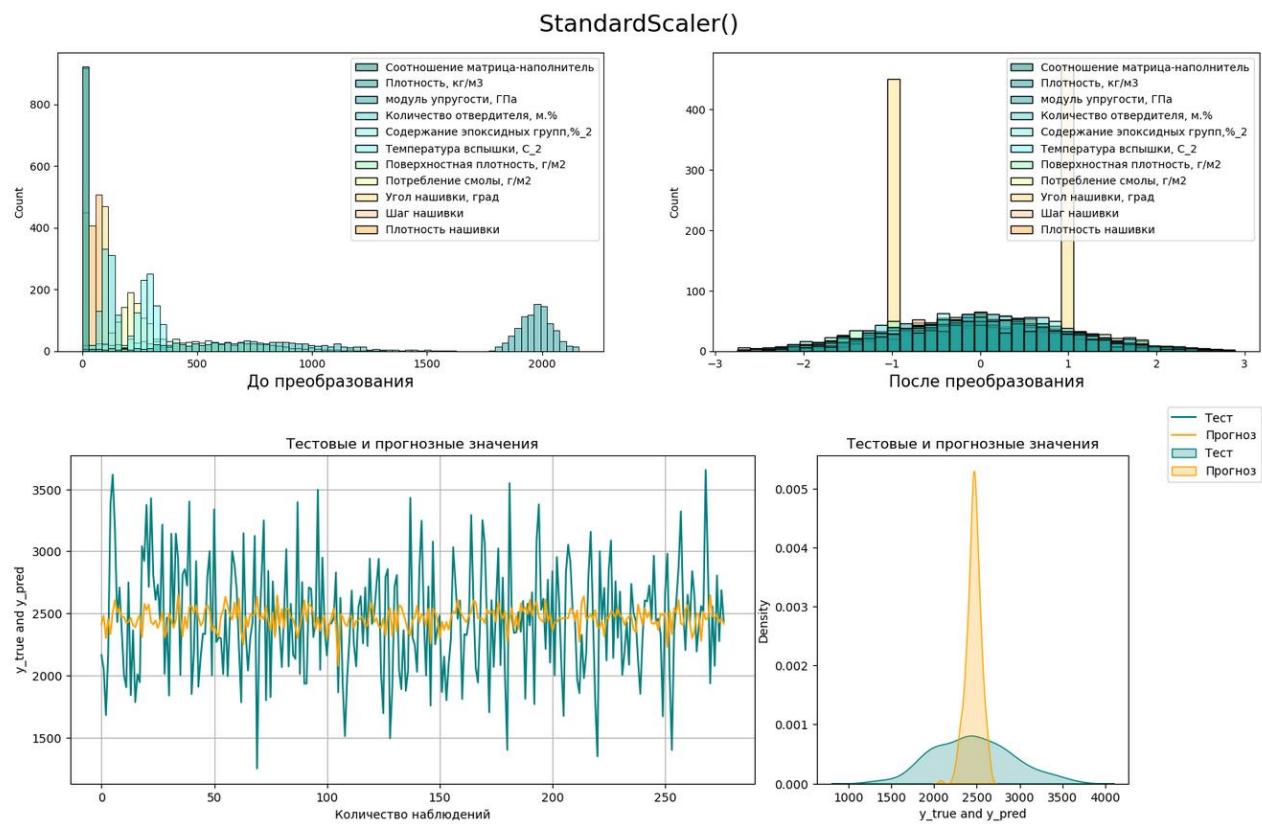


Рисунок 42 - результат работы модели GradientBoostingRegressor

е) Полносвязная нейронная сеть

Обучение на стандартизированных данных с помощью StandardScaler показали лучший RMSE = 445.62. График обучения нейронной сети показан на рисунке 43, а результат работы нейронной сети – на рисунке 44.

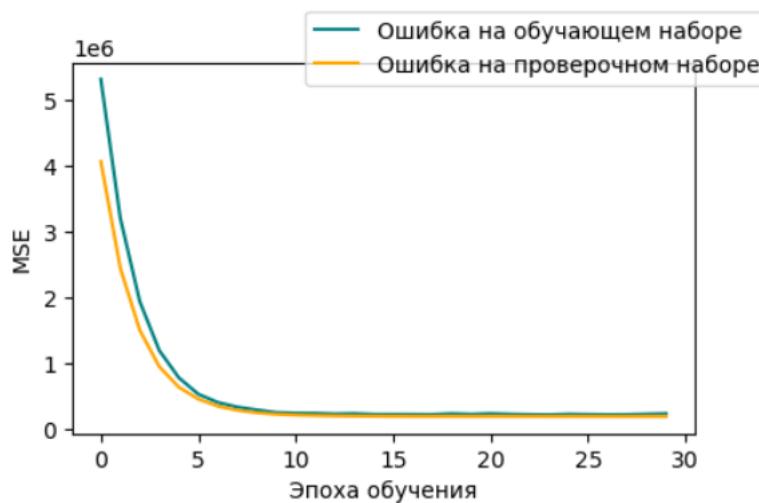


Рисунок 43 – график обучения нейронной сети

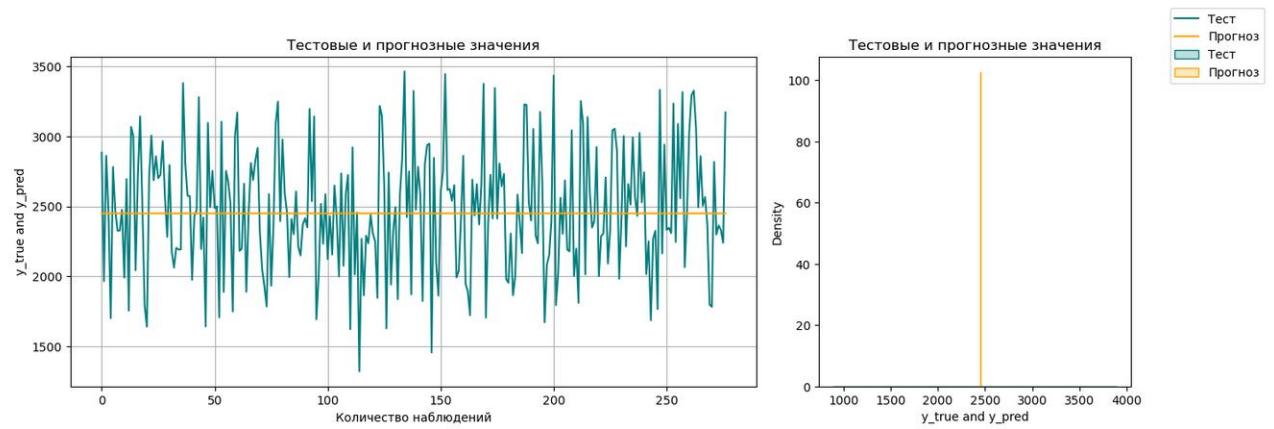


Рисунок 44 - результат работы нейронной сети со стандартизацией StandardScaler

2.3 Таблицы работы моделей

2.3.1 Модуль упругости при растяжении, ГПа

Сводная таблица метрик представлена на рисунке 45, а лучшие метрики работы моделей представлены на рисунке 46.

	Model_regr	R2	MSE	RMSE	MAE	MAPE
0	LinearRegression	-0.03	9.71	3.12	2.48	0.03
1	KNeighborsRegressor	-0.01	9.57	3.09	2.48	0.03
2	RandomForestRegressor	-0.05	9.94	3.15	2.48	0.03
3	SVR	-0.01	9.51	3.08	2.46	0.03
4	GradientBoostingRegressor	-0.04	9.83	3.14	2.50	0.03
5	Neural_net_minmax	0.01	9.75	3.12	2.50	0.03
6	Neural_net_stand_scal	-0.01	9.98	3.16	2.51	0.03
7	Neural_net_max_abs_scal	-0.01	9.89	3.14	2.52	0.03
8	Neural_net_robust_sc	-0.08	10.58	3.25	2.58	0.04

Рисунок 45 – сводная таблица метрик для «Модуля упругости при растяжении, ГПа»

Лучший показатель R2 = 0.009 в модели Neural_net_minmax
 Лучший показатель MSE = 9.5116 в модели SVR
 Лучший показатель MAE = 2.4591 в модели SVR

Рисунок 46 – показатели лучших метрик для «Модуля упругости при растяжении, ГПа»

Все значения R2 близки к 0, отрицательные значения говорят о том, что модель отработала хуже, чем если бы мы взяли просто средние значения. Положительный R2, но близкий к 0, говорит о том, что модель отработала чуть-чуть лучше, чем если бы мы взяли усредненные значения. Лучший показатель R2 = 0.009 у нейронной сети с нормализатором MinMaxScaler. Сохраним модель обучения нейронной сети с помощью `model.save()` из `tf.keras.models.save_model()`, а функцию нормализации с помощью `pickle` (Рисунок 47)

Сохраним модель

```
... pickle.dump(minmax, open('./models/minmax.pkl', 'wb'))  
... model_minmax_modul_elastic.save('./models/modul_elastic')  
INFO:tensorflow:Assets written to: ./models/modul_elastic\assets  
INFO:tensorflow:Assets written to: ./models/modul_elastic\assets
```

Рисунок 47 – сохранение модели предсказания Модуля упругости при растяжении, ГПа

2.3.2 Прочность при растяжении, МПа

Сводная таблица метрик представлена на рисунке 48, а лучшие метрики работы моделей представлены на рисунке 49.

	Model_regr	R2	MSE	RMSE	MAE	MAPE
0	LinearRegression	-0.01	214500.17	463.14	372.07	0.16
1	KNeighborsRegressor	0.01	209932.30	458.18	369.54	0.16
2	RandomForestRegressor	0.01	209452.24	457.66	366.52	0.16
3	SVR	0.00	211079.56	459.43	370.15	0.16
4	GradientBoostingRegressor	-0.02	215203.10	463.90	371.32	0.16
5	Neural_net_minmax	-0.00	198679.67	445.74	360.30	0.15
6	Neural_net_stand_scal	-0.00	198576.58	445.62	360.28	0.15
7	Neural_net_max_abs_scal	-0.01	199034.83	446.13	360.37	0.15
8	Neural_net_robust_sc	-0.00	198663.93	445.72	360.30	0.15

Рисунок 48 – сводная таблица метрик для «Прочности при растяжении, МПа»

```
Лучший показатель R2 = 0.0091 в модели RandomForestRegressor  
Лучший показатель MSE = 198576.5801 в модели Neural_net_stand_scal  
Лучший показатель MAE = 360.2825 в модели Neural_net_stand_scal
```

Рисунок 49 – показатели лучших метрик для
«Прочности при растяжении, МПа»

Все значения R2 близки к 0, отрицательные значения говорят о том, что модель отработала хуже, чем если бы мы взяли просто средние значения. Положительный R2, но близкий к 0, говорит о том, что модель отработала чуть-чуть лучше, чем если бы мы взяли усредненные значения. Лучший показатель R2 = 0.0091 у модели RandomForestRegressor, однако, лучшие значения MSE и MAE у нейронной сети со стандартизованными данными StandardScaler. Я приняла решение сохранить модель нейронной сети. Аналогично сохранению модели Модуля упругости при растяжении, сохраним модель обучения нейронной сети с помощью `model.save()` из `tf.keras.models.save_model()`, а функцию стандартизации с помощью `pickle` (Рисунок 50)

Сохраним модель

```
model_st_scal_tensile_strength.save('./models/tensile_strength')  
  
INFO:tensorflow:Assets written to: ./models/tensile_strength/assets  
INFO:tensorflow:Assets written to: ./models/tensile_strength/assets  
  
pickle.dump(robust_sc, open('./models/stand_scal.pkl', 'wb'))
```

Рисунок 50 – сохранение модели предсказания
Прочности при растяжении, МПа

2.4 Написать нейронную сеть, рекомендующую соотношение матрица-наполнитель

При построении нейронной сети для рекомендации соотношения матрица-наполнитель я решила предварительно нормализовать их с помощью `MinMaxScaler`.

Строю нейронную сеть с помощью класса keras.Sequential со следующими параметрами:

- входной слой для 12 признаков;
- выходной слой для 1 признака;
- скрытых слоев: 2;
- нейронов на скрытых слоях: 128, 64;
- слой Dropout(0.2) после каждого скрытого слоя
- активационная функция скрытых слоев: tanh;
- активационная функция выходного слоя: linear;
- оптимизатор: SGD;
- loss-функция: mse.

Запускаю обучение нейросети со следующими параметрами:

- количество эпох: 30;
- раннюю остановку не использую.

График уменьшения ошибки нейронной сети показан на рисунке 51, архитектура обучения – на рисунке 52.

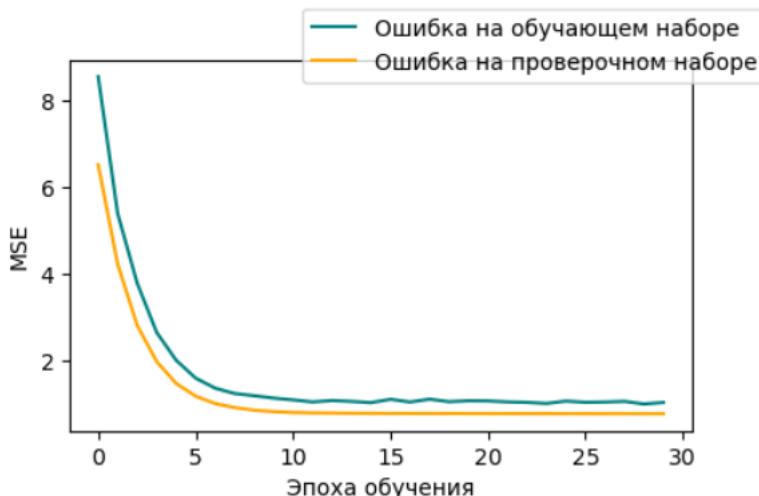


Рисунок 51 – график ошибки нейронной сети

```

Model: "sequential_8"
-----  

Layer (type)          Output Shape       Param #
-----  

dense_24 (Dense)      (None, 128)        1664  

dropout_16 (Dropout)  (None, 128)        0  

dense_25 (Dense)      (None, 64)         8256  

dropout_17 (Dropout)  (None, 64)         0  

dense_26 (Dense)      (None, 1)          65  

-----  

Total params: 9,985  

Trainable params: 9,985  

Non-trainable params: 0
-----  

None

```

Рисунок 52 – архитектура обучения нейронной сети

Результат работы нейронной сети для рекомендации соотношения матрица-наполнитель показан на рисунке 53, а метрики работы – на рисунке 54.

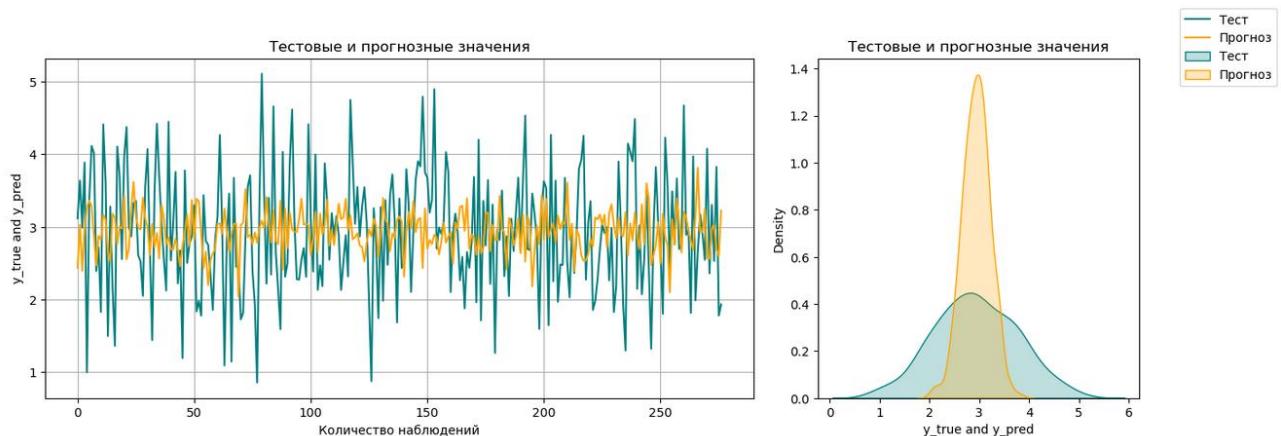


Рисунок 53 - результат работы нейронной сети

R2: -0.126
MSE: 0.774
RMSE: 0.88
MAE: 0.717
MAPE: 0.294

Рисунок 54 - метрики нейронной сети

Как видно из таблицы 54, нейронная сеть предсказала значения хуже, чем если бы мы взяли усредненные. Нейронная сеть не справилась с поставленной задачей.

Сохраним модель для написания приложения. Сделаем это так же, как в двух предыдущих случаях - с помощью `model.save()` из `tf.keras.models.save_model()`, а функцию нормализации сохраним с помощь pickle (Рисунок 55).

Сохраним модель

```
... model_matrix.filler.save('./models/matrixFiller')  
INFO:tensorflow:Assets written to: ./models/matrixFiller/assets  
INFO:tensorflow:Assets written to: ./models/matrixFiller/assets  
  
pickle.dump(minmax_matrix.filler, open('./models/minmax_matrixFiller.pkl', 'wb'))
```

Рисунок 55 – сохранение модели нейронной сети
для соотношения матрица-наполнитель

2.5 Разработка приложения

Несмотря на то, что хороших показателей к внедрению моделей получить не удалось, можно разработать функционал приложения. Возможно, дальнейшие исследования позволяют построить качественную модель и внедрить ее в готовое приложение.

В приложении необходимо реализовать следующие функции:

- выбор целевой переменной для предсказания;
- ввод входных параметров;
- проверка введенных параметров;
- загрузка сохраненной модели, получение и отображение прогноза выходных параметров.

Решено разработать веб-приложение с помощью языка Python, фреймворка Flask, шаблонизатора Jinja, WEB – страницы приложения написать на языке HTML с применением языка декорирования и описания внешнего вида документа CSS.

Эту задачу получилось решить. Главная страница (main) представлена на рисунке 56, окно с вводом параметров для рекомендации соотношения матрица-

наполнитель – на рисунке 57, окно с вводом параметров для предсказания модуля упругости при растяжении – на рисунке 58, окно ввода параметров для предсказания прочности при растяжении – рисунок 59. Окно с выводом рекомендации соотношения матрица-наполнитель показано на рисунке 60.



Рисунок 56 – главная страница разработанного WEB-приложения

Расчет соотношения матрица-наполнитель

Введите параметры:

Плотность, кг/м ³	<input type="text"/>
Модуль упругости, ГПа	<input type="text"/>
Количество отвердителя, м.%	<input type="text"/>
Содержание эпоксидных групп,%_2	<input type="text"/>
Температура вспышки, С_2	<input type="text"/>
Поверхностная плотность, г/м ²	<input type="text"/>
Модуль упругости при растяжении, ГПа	<input type="text"/>
Прочность при растяжении, МПа	<input type="text"/>
Потребление смолы, г/м ²	<input type="text"/>
Угол нашивки, град	<input type="text"/>
Шаг нашивки	<input type="text"/>
Плотность нашивки	<input type="text"/>

[Вернуться на главную страницу](#)

Рисунок 57 - окно с вводом параметров для рекомендации соотношения
матрица-наполнитель

Расчет модуля упругости при растяжении

Введите параметры

Соотношение матрица-наполнитель, МПа

Плотность, кг/м³

Модуль упругости, ГПа

Количество отвердителя, м.%

Содержание эпоксидных групп,%_2

Температура вспышки, С_2

Поверхностная плотность, г/м²

Потребление смолы, г/м²

Угол нашивки, град ⚡

Шаг нашивки

Плотность нашивки

Рассчитать **Сброс**

Вернуться на главную страницу

Рисунок 58 - окно с вводом параметров для предсказания
модуля упругости при растяжении

Расчет прочности при растяжении

Введите параметры

Соотношение матрица-наполнитель, МПа

Плотность, кг/м³

Модуль упругости, ГПа

Количество отвердителя, м.%

Содержание эпоксидных групп,%_2

Температура вспышки, С_2

Поверхностная плотность, г/м²

Потребление смолы, г/м²

Угол нашивки, град ⚡

Шаг нашивки

Плотность нашивки

Рассчитать **Сброс**

Вернуться на главную страницу

Рисунок 59 - окно с вводом параметров для предсказания
прочности при растяжении

**Соотношение матрица-наполнитель для введенных параметров:
[[2.2363853]]**

Расчет соотношения матрица-наполнитель

Введите параметры

Плотность, кг/м³

Рисунок 60 - вывод рекомендации соотношения матрица-наполнитель

Инструкция по использованию WEB-приложения «Прогнозирование конечных свойств новых материалов (композиционных материалов)»:

- 1) на главной странице выбрать значение, которое необходимо предсказать (рекомендовать);
- 2) ввести параметры, указанные на странице предсказания;
- 3) нажать на кнопку «Рассчитать» в случае, если введенные параметры верны, или кнопку «Сброс», если была допущена ошибка при вводе входных параметров;
- 4) нажать на кнопку «Вернуться на главную страницу», если был ошибочно выбран не тот пункт меню или работа с предсказанием по данной позиции завершена.

2.6 Удаленный репозиторий и загрузка работы на него

Данное исследование я загрузила в свой аккаунт на GitHub (Рисунок 61), ссылка на репозиторий: <https://github.com/mazavia/Diplom.git>

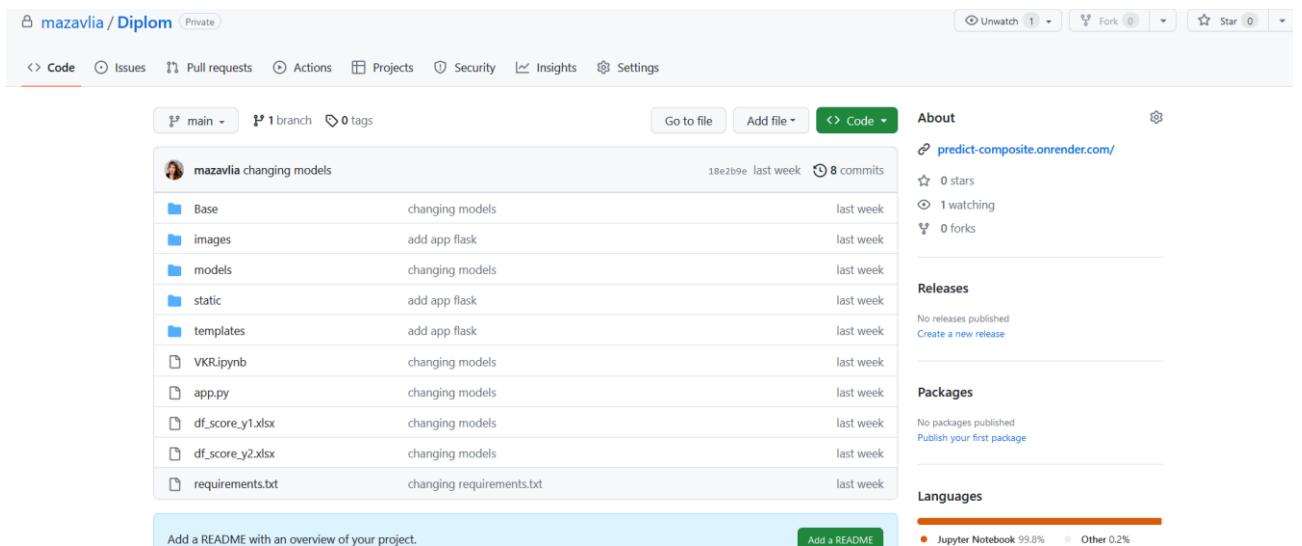


Рисунок 61 – файлы ВКР на GitHub

Deploy WEB – приложения сделала на сайте www.render.com (Рисунок 62). Адрес приложения в интернете: <https://composite-predict.onrender.com> (Рисунки 63, 64).

The screenshot shows the Render.com interface. At the top, there's a navigation bar with 'render', 'Dashboard', 'Blueprints', 'Env Groups', 'Docs', 'Community', 'Help', 'New +', a user profile for 'Marina Zaporozhets', and a gear icon. Below the navigation is a section for a 'WEB SERVICE' named 'composite_predict'. It shows details like 'Python 3', 'Free', 'mazavlia/Diplom', 'main', and a URL 'https://composite-predict.onrender.com'. There are 'Connect' and 'Manual Deploy' buttons. On the left, a sidebar lists various monitoring and management options: Events (selected), Logs, Disks, Environment, Shell, PRs, Jobs, Metrics, Scaling, and Settings. The main area displays a list of deployment events:

- Deploy live for 18e2b9e: changing models** (April 14, 2023 at 6:56 PM)
- Deploy started for 18e2b9e: changing models** (New commit, April 14, 2023 at 6:47 PM)
- Deploy live for 26d53e5: changing address** (April 14, 2023 at 11:20 AM) with a 'Rollback to this deploy' button
- Deploy started for 26d53e5: changing address** (New commit, April 14, 2023 at 11:12 AM)

Рисунок 62 – Deploy приложения на WEB- хостинге

The screenshot shows a web browser window with multiple tabs. The active tab is titled 'composite_predict - Web Server' and has the URL 'https://composite-predict.onrender.com/matrix_filter/'. The page content is as follows:

Предсказание конечных свойств получаемых композиционных материалов по имеющимся измерениям.

На вход подаются данные о начальных свойствах компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.).

Матрица-наполнитель
Модуль упругости при растяжении
Прочность при растяжении

Композиционные материалы – это искусственно созданные материалы, состоящие из нескольких других с четкой границей между ними. Композиты обладают теми свойствами, которые не наблюдаются у компонентов по отдельности. При этом композиты являются монолитным материалом, т.е. компоненты материала неотделимы друг от друга без разрушения конструкции в целом. Яркий пример композита – железобетон. Бетон прекрасно сопротивляется сжатию, но плохо растяжению. Стальная арматура внутри бетона компенсирует его неспособность сопротивляться сжатию, формируя тем самым новые, уникальные свойства. Современные композиты изготавливаются из других материалов: полимеры, керамика, стеклянные и углеродные волокна, но данный принцип сохраняется. У такого подхода есть и недостаток: даже если мы знаем характеристики исходных компонентов, определить характеристики композита, состоящего из этих компонентов, достаточно проблематично. Для решения этой проблемы есть два пути: физические испытания образцов материалов, или прогнозирование характеристик. Суть прогнозирования заключается в симуляции представительного элемента объема композита на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

Рисунок 63 – главная страница приложения на сайте render.com

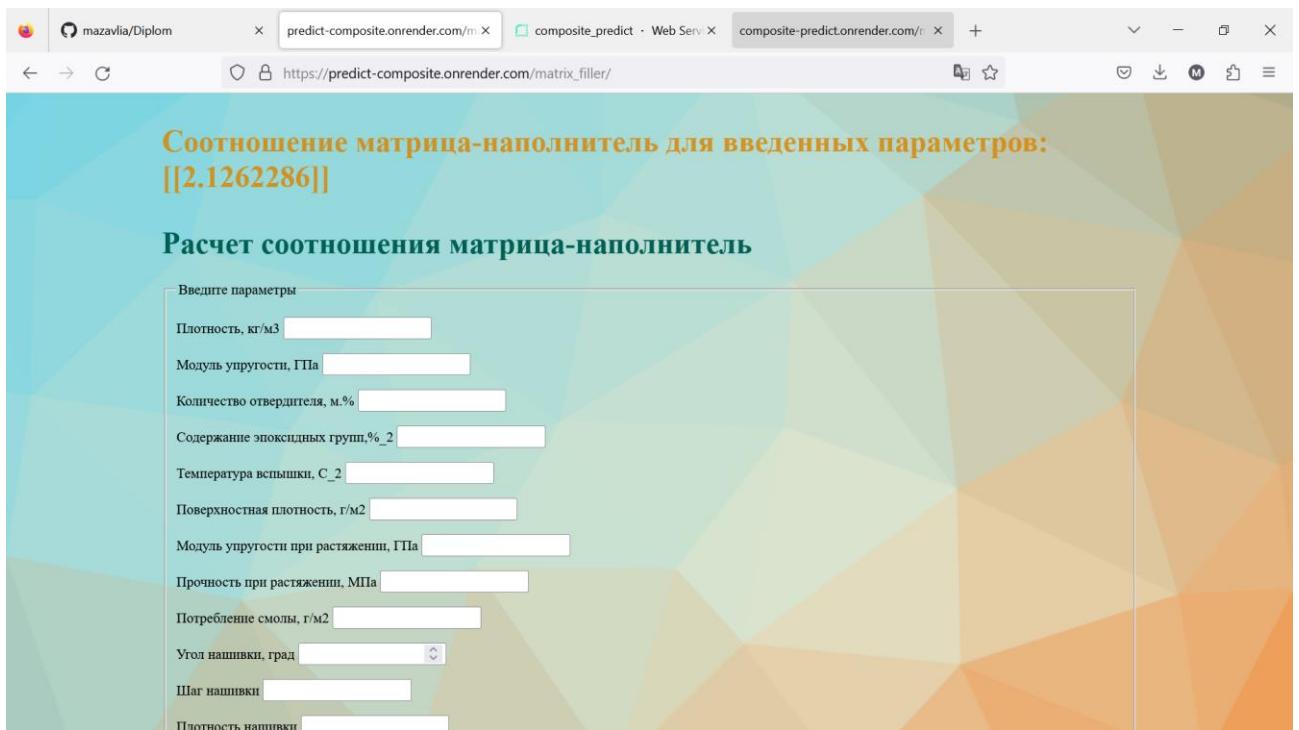


Рисунок 64 – рассчитанное рекомендуемое значение для соотношения матрица-наполнитель для введенных параметров

Заключение

В этой работе мы имели дело с реальной производственной задачей. И, к сожалению, не смогли поставленную задачу решить — не получили моделей, которые бы описывали закономерности предметной области. Я проделала максимум исследований, которые в моей компетенции, как начинающего специалиста в Data Science, применила большую часть знаний, полученных в ходе прохождения курса.

Возможные причины неудачи:

- нечеткая постановка задачи, отсутствие дополнительной информации о зависимости признаков с точки зрения физики процесса. Незначимые признаки являются для модели шумом, и мешают найти зависимость целевых от значимых входных признаков;
- исследование предварительно обработанных данных. Возможно, на "сырых" данных можно было бы получить более качественные модели, воспользовавшись другими методами очистки и подготовки;
- мой недостаток знаний и опыта. Нейросети являются самым современным подходом к решению такого рода задач. Они способны находить скрытые и нелинейные зависимости в данных. Но выбор оптимальной архитектуры нейросети является неочевидной задачей.

Дальнейшие возможные пути решения этой задачи могли бы быть:

- углубиться в изучение нейросетей, попробовать различные архитектуры, параметры обучения и т.д.;
 - провести отбор признаков разными методами. Испробовать методы уменьшения размерности, например метод главных компонент;
 - после уменьшения размерности градиентный бустинг может улучшить свои результаты. Так же есть большой простор для подбора гиперпараметров для этого метода;
- проконсультироваться у экспертов в предметной области. Возможно, они могли бы поделиться знаниями, необходимыми для решения задачи.

Библиографический список

1 Композиционные материалы : учебное пособие для вузов / Д. А. Иванов, А. И. Ситников, С. Д. Шляпин ; под редакцией А. А. Ильина. — Москва : Издательство Юрайт, 2019 — 253 с. — (Высшее образование). — Текст : непосредственный.

2 Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.

3 ГрасД. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.

4 Документация по библиотеке numpy: – Режим доступа:
<https://numpy.org/doc/1.22/user/index.html#user>.

5 Документация по библиотеке pandas: – Режим доступа:
https://pandas.pydata.org/docs/user_guide/index.html#user-guide.

6 Документация по библиотеке matplotlib: – Режим доступа:
<https://matplotlib.org/stable/users/index.html>.

7 Документация по библиотеке seaborn: – Режим доступа:
<https://seaborn.pydata.org/tutorial.html>.

8 Документация по библиотеке sklearn: – Режим доступа: https://scikit-learn.org/stable/user_guide.html.

9 Документация по библиотеке keras: – Режим доступа: <https://keras.io/api/>.

10 Руководство по быстрому старту в flask: – Режим доступа: <https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html>.

11 Loginom Вики. Алгоритмы: – Режим доступа:
<https://wiki.loginom.ru/algorithms.html>.

12 Andre Ye. 5 алгоритмов регрессии в машинном обучении, о которых вам следует знать: – Режим доступа:<https://habr.com/ru/company/vk/blog/513842/>.

13 Alex Maszański. Метод k-ближайших соседей (k-nearest neighbour): – Режим доступа: <https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19>.

14 Yury Kashnitsky. Открытый курс машинного обучения. Тема 3. Классификация, деревья решений и метод ближайших соседей: – Режим доступа: <https://habr.com/ru/company/ods/blog/322534/>.

15 Alex Maszański. Машинное обучение для начинающих: алгоритм случайного леса (Random Forest): – Режим доступа: <https://proglib.io/p/mashinnoe-obuchenie-dlya-nachinayushchih-algoritm-sluchaynogo-leza-random-forest-2021-08-12>.

16 Alex Maszański. Решаем задачи машинного обучения с помощью алгоритма градиентного бустинга: – Режим доступа: <https://proglib.io/p/reshaem-zadachi-mashinnogo-obucheniya-s-pomoshchyu-algoritma-gradientnogo-bustinga-2021-11-25>.

17 Dmitry Makarov. Машинное обучение. Вводный курс ML: – Режим доступа: <https://www.dmitrymakarov.ru/intro/>