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»

Слушатель Ларионов Владислав Викторович

Москва, 2023

Содержание

Введение 4
1. Аналитическая часть 6
1.1. Постановка задачи 6
1.2. Описание используемых методов 14
1.2.1 Линейная регрессия 14
1.2.2 Лассо (LASSO) и гребневая (Ridge) регрессия 14
1.2.3 Метод опорных векторов для регрессии 15
1.2.4 Метод k-ближайших соседей 16
1.2.5 Деревья решений 17
1.2.6 Случайный лес 18
1.2.7 Градиентный бустинг. 19
1.2.8 Нейронная сеть 20
1.3. Разведочный анализ данных 22
1.3.1 Выбор признаков 23
1.3.2 Ход решения задачи 24
1.3.2 Препроцессинг 25
1.3.3 Перекрестная проверка 26
1.3.4 Поиск гиперпараметров по сетке 26
1.3.5 Метрики качества моделей 26
2. Практическая часть 28
2.1. Разбиение и предобработка данных 28
2.1.1 Для прогнозирования модуля упругости при растяжении 28
2.1.2 Для прогнозирования прочности при растяжении 28
2.1.3 Для прогнозирования соотношения матрица-наполнитель 29
2.2 Разработка и обучение моделей для прогнозирования модуля упругости при растяжении 30
2.3 Для прогнозирования прочности при растяжении 33

2.4 Разработка нейронной сети для прогнозирования соотношения матрица-наполнитель 35
2.4.1 MLPRegressor из библиотеки sklearn 35
2.4.2 Нейросеть из библиотеки tensorflow 39
2.5 Тестирование модели 43
2.6. Разработка приложения 46
2.7. Создание удаленного репозитория 46
Заключение 47
Библиографический список 49
Приложение А. Скриншоты веб-приложения 51

Введение

Современная наука и технологии постоянно развиваются, и одним из ключевых направлений является разработка новых материалов с уникальными свойствами. В этой работе мы рассмотрим тему прогнозирования конечных свойств новых композиционных материалов.

Композиционными материалами называются материалы, в которых имеет место сочетание двух или более химических компонентов с четко определенной границей раздела между ними. Эти компоненты могут быть как однородными, так и неоднородными по своему составу и структуре. Структура композиционных материалов представляет собой матрицу, которая содержит в себе основной компонент и армирующие элементы, часто называемые наполнителем. Матрица и наполнитель разделены границей раздела. Наполнитель равномерно распределен в матрице и имеет определенную пространственную ориентацию.

Композиционные материалы характеризуются совокупностью свойств, которые не присущи каждому отдельно взятому компоненту. За счет использования армирующих элементов, варьирования их объемной доли в матричном материале, а также размеров, формы, ориентации и прочности связи по границе «матрица-наполнитель» свойства композиционных материалов можно регулировать в значительных пределах. Возможно получить композиты с уникальными эксплуатационными свойствами. Таким образом, композиты широко используются в различных областях техники.

Учитывая такое широкое распространение и высокую потребность в новых материалах, тема данной работы является очень актуальной.

Описание процесса создания композитного материала

Процесс создания композитного материала начинается с выбора основных компонентов, таких как углеродное волокно, стекловолокно, полимеры и другие. Затем происходит процесс смешивания этих компонентов в определенных пропорциях и условиях. После этого происходит формирование матрицы, которая

состоит из основного компонента и армирующих элементов. Далее происходит заполнение матрицы наполнителем, который равномерно распределяется в матрице и имеет определенную пространственную ориентацию. Наконец, композит подвергается испытаниям на прочность и устойчивость к различным нагрузкам.

Из указанного следует вывод о том, что прогнозирование конечных свойств новых композиционных материалов является важной задачей современной науки и технологий. Для достижения желаемых результатов необходимо проводить многочисленные испытания различных комбинаций компонентов. Система поддержки производственных решений может значительно ускорить этот процесс и снизить стоимость производства композитного материала.

1. Аналитическая часть

1.1. Постановка задачи

В данном исследовании мы рассматриваем композитный материал, состоящий из базальтопластика в качестве матрицы и углепластика в качестве нашивки. Мы получили датасет от специалистов, содержащий информацию о свойствах матрицы и наполнителя, производственных параметрах и свойствах готового композита. Наша задача, как специалистов в области машинного обучения, заключается в разработке моделей, которые могут прогнозировать значения определенных свойств в зависимости от остальных параметров. Кроме того, мы также планируем создать приложение, которое облегчит использование этих моделей для специалистов в данной области.

Датасет состоит из двух файлов: X_bp (матрица) и X_nup (наполнитель).

Файл X_bp содержит:

– признаков: 10 и индекс;

– строк: 1023.

Файл X_nup содержит:

– признаков: 3 и индекс;

– строк: 1040.

В соответствии с первым непоименованным заданием нам установлено, что файлы требуют объединения с типом INNER по индексу.

После объединения файла X_nup часть строк была исключена. Дальнейшие исследования проводятся с использованием объединенного датасета, который содержит 13 признаков и 1023 строки или объекта.

Таблица 1 представляет описание признаков в объединенном датасете. Все признаки имеют тип float64, то есть являются вещественными. В данных отсутствуют пропущенные значения. Все признаки, за исключением "Угола нашивки", являются непрерывными и количественными. "Угол нашивки" принимает только два значения и будет рассматриваться как категориальный признак.

Таблица 1 — Описание признаков датасета

Название	Тип данных	Непустых значений	Уникальных значений	Количество пропусков
Соотношение матрица-наполнитель	float64	1023	1014	0
Плотность, кг/м3	float64	1023	1013	0
модуль упругости, ГПа	float64	1023	1020	0
Количество отвердителя, м.%	float64	1023	1005	0
Содержание эпоксидных групп,%_2	float64	1023	1004	0
Температура вспышки, С_2	float64	1023	1003	0
Поверхностная плотность, г/м2	float64	1023	1004	0
Модуль упругости при растяжении, ГПа	float64	1023	1004	0
Прочность при растяжении, МПа	float64	1023	1004	0
Потребление смолы, г/м2	float64	1023	1003	0
Угол нашивки, град	float64	1023	2	0
Шаг нашивки	float64	1023	989	0
Плотность нашивки	float64	1023	988	0

Исходя из представленного объединённого датасета делаем вывод, что датасет был предварительно подготовлен и очищен от пропусков.

После построения гистограмм распределения и скрипичных диаграмм было определено, что все признаки, кроме признака «Угол нашивки», имеют

нормальное распределение Гаусса Лапласа и принимают неотрицательные значения. Violinplot отлично визуализирует медианные значения «Угол нашивки» принимает значения: 0, 90. Данные изображены на рисунках 1-3.

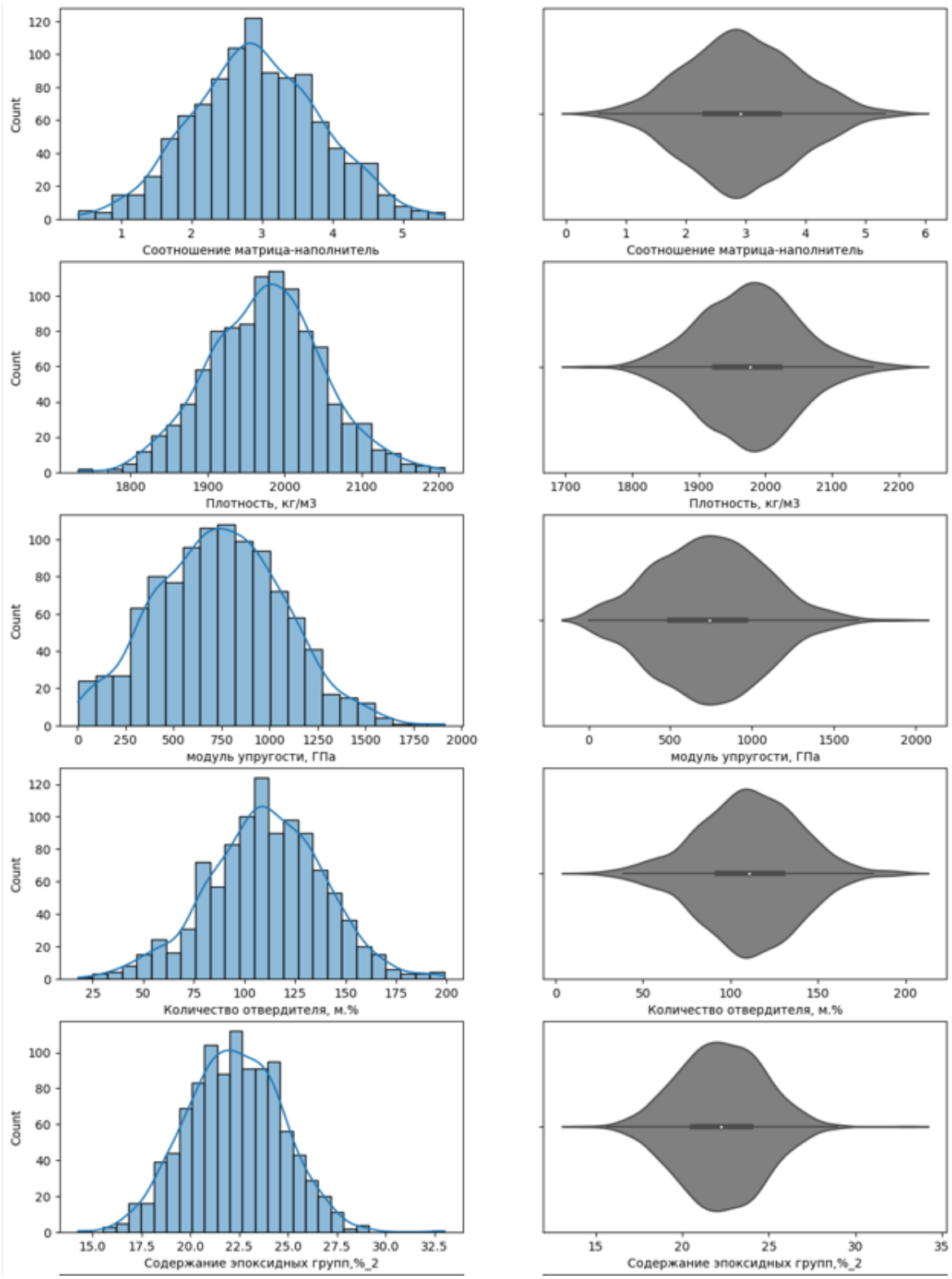

Рисунок 1- Гистограммы распределения и скрипичные диаграммы

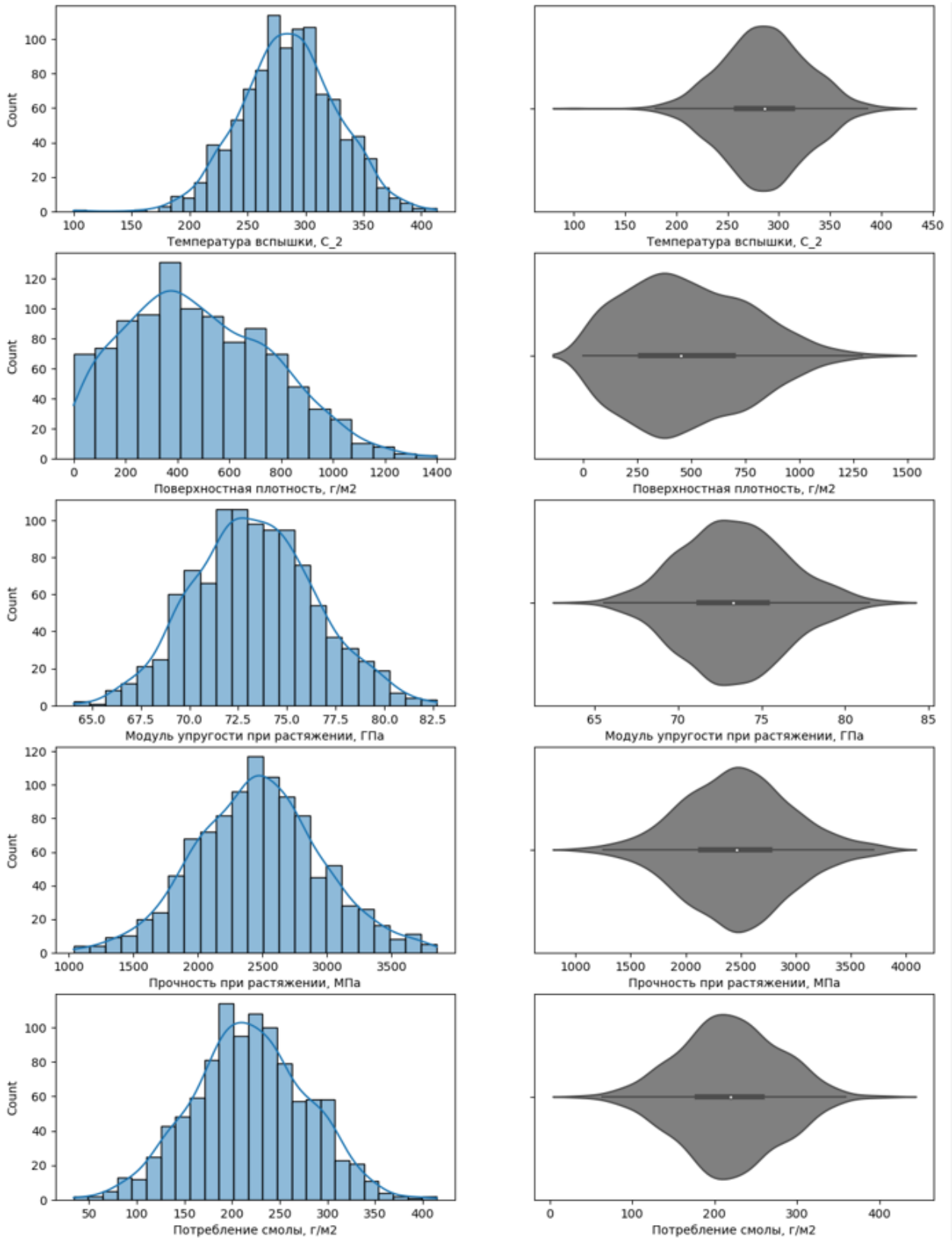

Рисунок 2 - Гистограммы распределения и скрипичные диаграммы

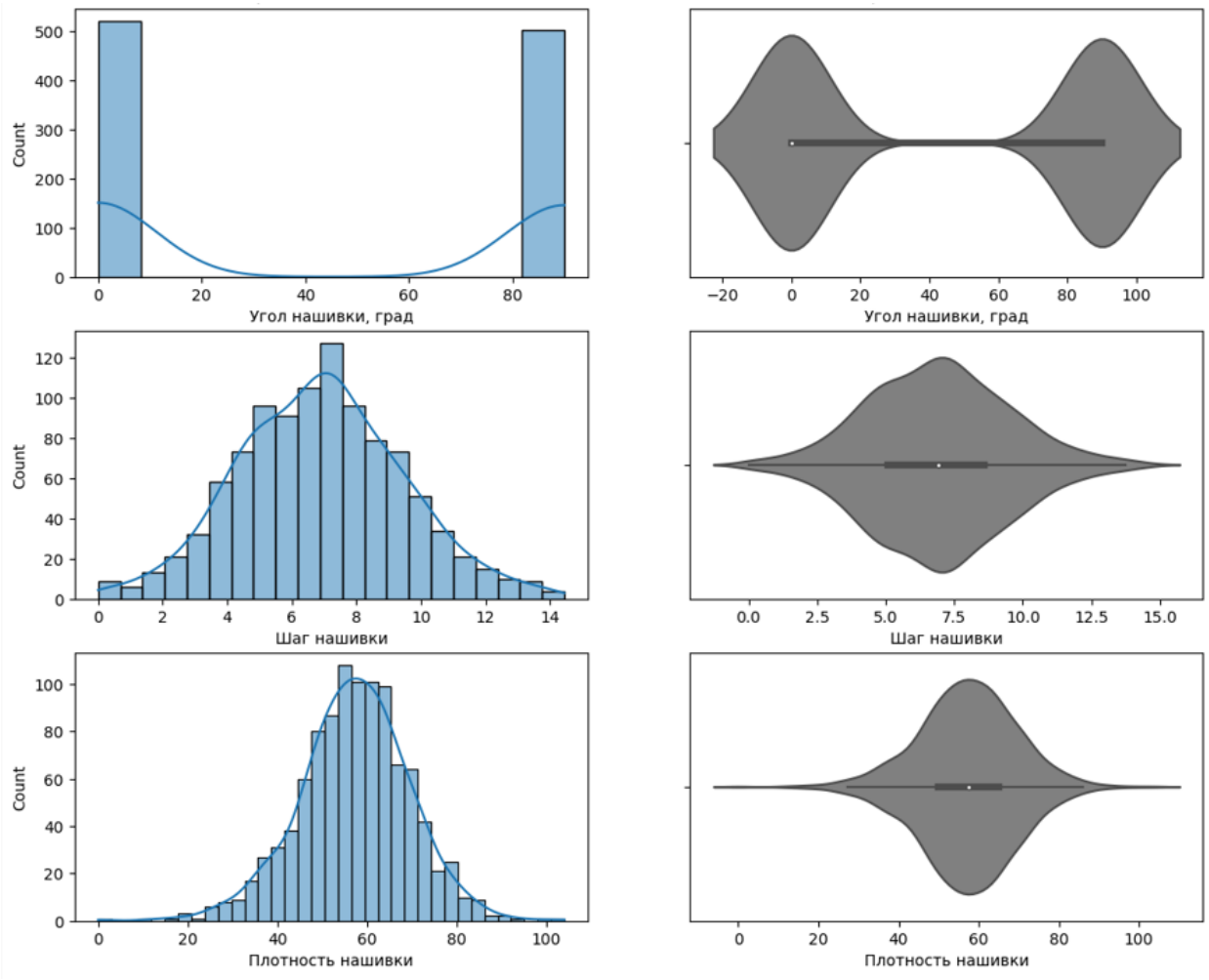

Рисунок 3 - Гистограммы распределения и скрипичные диаграммы

Описательная статистика датасета представлена в таблице 2. Она в численном виде отражает то, что мы видим на гистограммах.

На рисунке 4 представлены попарные графики рассеяния точек. Обратите внимание, что на них можно заметить некоторые точки, которые находятся значительно дальше от основного облака точек. Эти точки визуально представляют выбросы - аномальные значения данных, которые выходят за пределы допустимых значений признаков.

Таблица 2 — Описательная статистика признаков датасета

	Среднее	Стандартное отклонение	Минимум	Максимум	Медиана
Соотношение матрица-наполнитель	2.9304	0.9132	0.3894	5.5917	2.9069

Плотность, кг/м3	1975.7349	73.7292	1731.7646	2207.7735	1977.6217
модуль упругости, ГПа	739.9232	330.2316	2.4369	1911.5365	739.6643
Количество отвердителя, м.%	110.5708	28.2959	17.7403	198.9532	110.5648
Содержание эпоксидных групп,%_2	22.2444	2.4063	14.2550	33.0000	22.2307
Температура вспышки, С_2	285.8822	40.9433	100.0000	413.2734	285.8968
Поверхностная плотность, г/м2	482.7318	281.3147	0.6037	1399.5424	451.8644
Модуль упругости при растяжении, ГПа	73.3286	3.1190	64.0541	82.6821	73.2688
Прочность при растяжении, МПа	2466.9228	485.6280	1036.8566	3848.4367	2459.5245
Потребление смолы, г/м2	218.4231	59.7359	33.8030	414.5906	219.1989
Угол нашивки, град	44.2522	45.0158	0.0000	90.0000	0.0000
Шаг нашивки	6.8992	2.5635	0.0000	14.4405	6.9161
Плотность нашивки	57.1539	12.3510	0.0000	103.9889	57.3419

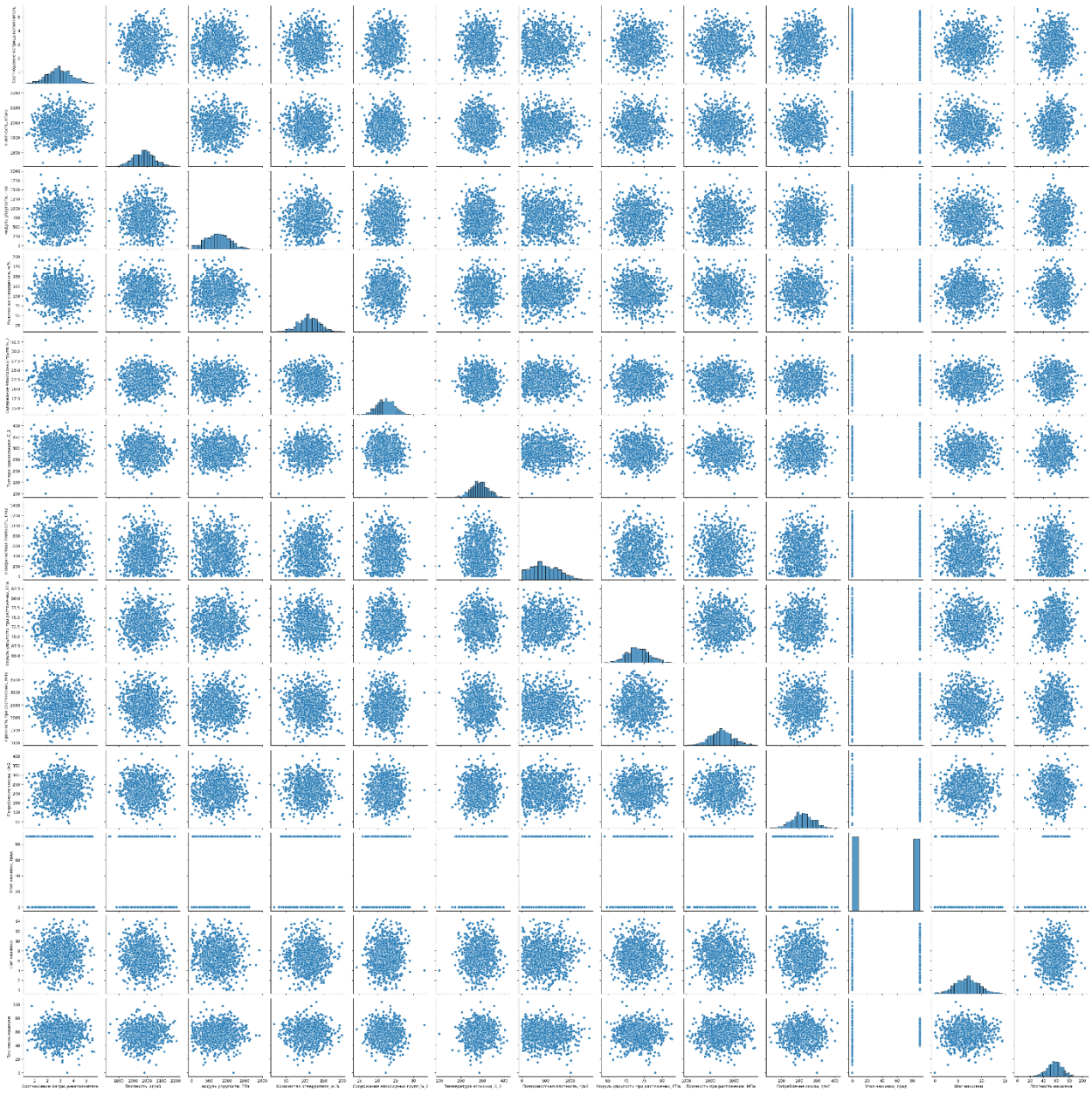

Рисунок 4 — Попарные графики рассеяния точек

Для признаков с нормальным распределением можно использовать следующие методы выявления выбросов:

1. Метод 3-х сигм: В этом методе выбросы считаются значениями, которые находятся за пределами трех стандартных отклонений от среднего значения признака.

2. Метод межквартильных расстояний: В этом методе выбросы определяются на основе разницы между третьим и первым квартилями (IQR). Значения, которые находятся за пределами величины 1.5 * IQR за пределами первого и третьего квартиля, считаются выбросами.

При применении этих методов к нашему датасету были найдены следующие выбросы:

- Методом 3-х сигм: обнаружено 24 выброса.

- Методом межквартильных расстояний: обнаружено 93 выброса. Пример выбросов на гистограмме распределения и диаграмме «ящик с усами» приведен на рисунке 5.

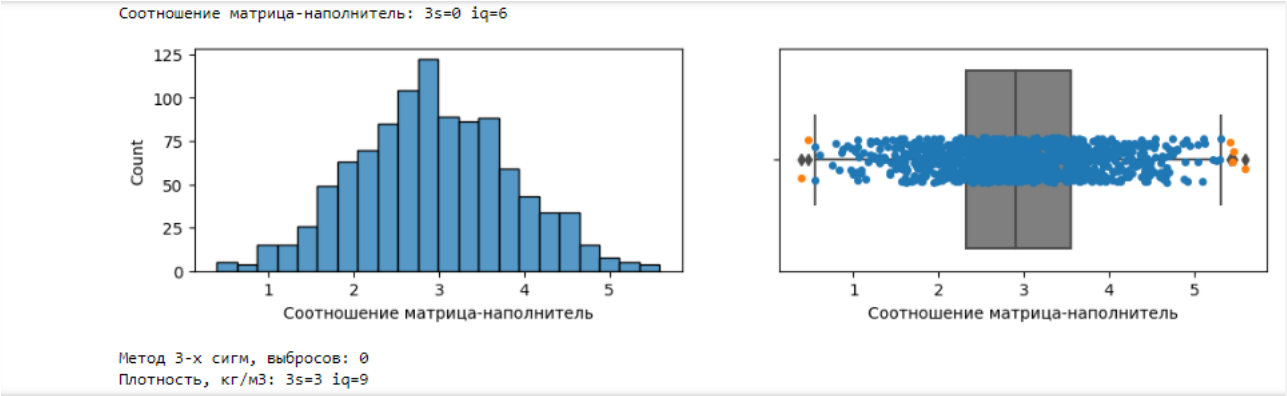

Рисунок 5 — Пример выбросов

Учитывая, что датасет уже был очищен от явного шума, рекомендуется применить метод 3-х сигм для выявления выбросов, так как он более деликатный и поможет избежать потери значимых данных. Кроме того в методе Метод межквартильных расстояний есть своеобразное «Философское затруднение» - искать выбросы с помощью среднего и отклонения, чьи значения как раз и обусловлены наличием выбросов - порочный круг.

Значения, определенные как выбросы, могут быть удалены из датасета. После этого удаления в датасете останется 1000 строк и 13 переменных-признаков.

В задании указано, что целевыми переменными являются:

1. Модуль упругости при растяжении, измеряемый в гигапаскалях (ГПа).

2. Прочность при растяжении, измеряемая в мегапаскалях (МПа).

3. Соотношение матрица-наполнитель.

Данные переменные будут использоваться в дальнейшем анализе и моделировании.

1.2. Описание используемых методов

Для предсказания значений вещественной, непрерывной переменной используется задача регрессии. В данной задаче зависимая переменная должна быть связана с одной или несколькими независимыми переменными, также известными как предикторы или регрессоры. Регрессионный анализ позволяет понять, как "типичное" значение зависимой переменной изменяется при изменении независимых переменных.

Существует множество методов регрессионного анализа, включая простую и множественную линейную регрессию. Эти модели являются параметрическими, что означает, что функция регрессии определяется конечным числом неизвестных параметров, которые оцениваются на основе доступных данных.

1.2.1 Линейная регрессия

Верно, простая линейная регрессия имеет место, когда рассматривается зависимость между одной входной и одной выходной переменными. Уравнение простой линейной регрессии имеет вид y = ax + b (1), где a и b - коэффициенты модели, которые определяют наклон и смещение линии регрессии. Коэффициенты обычно оцениваются методом наименьших квадратов.

Если требуется анализировать зависимость между несколькими входными и одной выходной переменными, то используется множественная линейная регрессия. Уравнение множественной линейной регрессии имеет вид Y = b0 + b1 * x1 + b2 * x2 + ... + bn * xn (2), где n - число входных переменных.

В Python линейная регрессия реализована в модуле sklearn.linear_model.LinearRegression. Этот модуль позволяет легко создавать и обучать модель линейной регрессии с использованием метода наименьших квадратов. где n - число входных переменных.

1.2.2 Лассо (LASSO) и гребневая (Ridge) регрессия

Метод регрессии LASSO (Least Absolute Shrinkage and Selection Operator) и гребневая регрессия (Ridge regression) являются вариациями линейной регрессии с регуляризацией. Они применяются для данных, которые имеют сильную корреляцию между признаками или мультиколлинеарность.

LASSO использует регуляризацию L1, добавляя штраф для весовна основе их абсолютных значений. Это позволяет автоматически отбирать наиболее значимые признаки и исключать избыточные или сильно коррелированные признаки. Регуляризация L1 также способствует упрощению модели и сокращению сложности данных.

Гребневая регрессия использует регуляризацию L2, которая штрафует коэффициенты модели на основе их квадратов. Это также помогает снизить влияние сильно коррелированных признаков и уменьшить мультиколлинеарность. Регуляризация L2 наказывает за более значительные ошибки, что способствует более устойчивой модели.

Оба метода, LASSO и гребневая регрессия, доступны в библиотеке scikit-learn (sklearn) в модулях sklearn.linear_model.Lasso и sklearn.linear_model.Ridge. Они позволяют создавать и обучать модели с использованием регуляризации и проводить отбор признаков на основе их значимости.

1.2.3 Метод опорных векторов для регрессии

Метод опорных векторов (support vector machine, SVM) является одним из наиболее популярных алгоритмов машинного обучения, который применяется для задач классификации и регрессии.

Основная идея SVM заключается в построении гиперплоскости или набора гиперплоскостей в многомерном пространстве, которые разделяют объекты разных классов оптимальным образом. Гиперплоскость выбирается таким образом, чтобы иметь наибольшее расстояние (зазор) до ближайших точек обучающей выборки разных классов. Такие ближайшие точки обучающей выборки называются опорными векторами.

Если исходные данные не являются линейно разделимыми, то SVM может использовать ядерную функцию для выполнения преобразования данных в пространство большей размерности, где они становятся линейно разделимыми. Некоторые известные ядерные функции включают линейную, полиномиальную и гауссовскую (rbf).

SVM решает задачу оптимизации, чтобы найти оптимальное разделение. Кроме того, параметр C используется для регуляризации, контролирующий баланс между достижением наилучшего разделения и минимизацией ошибок.

Основные преимущества SVM включают его хорошую изученность и способность обрабатывать данные с большим количеством признаков. Однако у SVM есть и некоторые недостатки, такие как чувствительность к выбросам и отсутствие интерпретируемости модели.

Вариация метода SVM для регрессии называется Support Vector Regression (SVR). Она используется для предсказания непрерывных значений. Реализация SVR в Python можно найти в модуле sklearn.svm.SVR библиотеки scikit-learn.

1.2.4 Метод k-ближайших соседей

Метод k-ближайших соседей (k Nearest Neighbors, kNN) является еще одним методом, который может быть использован для задачи регрессии.

В методе kNN для классификации объекту присваивается класс, который является наиболее распространенным среди k ближайших соседей этого объекта. В случае регрессии, объекту присваивается среднее значение целевой переменной по k ближайшим к нему объектам, значения которых уже известны.

Для работы метода kNN требуется определить метрику расстояния между объектами. Для количественных признаков обычно используется евклидово расстояние, а для категориальных признаков может применяться расстояние Хэмминга или другие подходящие метрики.

Метод kNN относится к непараметрическим методам регрессии, так как он не предполагает никаких предположений о распределении данных.

В библиотеке scikit-learn (sklearn) метод k-ближайших соседей для регрессии реализован в классе sklearn.neighbors.KNeighborsRegressor. Он позволяет создавать модели, основанные на методе kNN, и проводить регрессионные прогнозы на основе ближайших соседей.

1.2.5 Деревья решений

Деревья решений (Decision Trees) являются еще одним методом, который может быть использован и для классификации, и для регрессии. Деревья решений представляют собой иерархические структуры, где каждый узел представляет собой решающее правило, а листы содержат результаты, такие как классы или значения целевой переменной.

В деревьях решений, принятие решений происходит путем проверки условий на разных уровнях дерева, чтобы определить, какой путь следует выбрать. Процесс обучения дерева решений заключается в автоматическом формировании решающих правил на основе обучающих примеров. Это происходит путем разбиения множества примеров на более чистые подмножества, где каждое разбиение основано на выбранном атрибуте.

В деревьях решений для регрессии, критерием разбиения обычно является минимизация дисперсии вокруг среднего значения целевой переменной в каждом листе дерева. Это позволяет найти наилучшие признаки, которые разделяют выборку таким образом, чтобы значения целевой переменной в каждом листе были примерно равны.

Библиотека scikit-learn (sklearn) предоставляет класс sklearn.tree.DecisionTreeRegressor, который реализует деревья решений для задачи регрессии. Он позволяет создавать и обучать модели деревьев решений для прогнозирования значений целевой переменной на основе входных признаков.

Однако важно отметить, что деревья решений могут склоняться к переобучению, особенно если допущены слишком глубокие деревья. Для борьбы с

переобучением можно использовать различные стратегии, такие как ограничение глубины дерева, установка минимального количества примеров в листе или применение алгоритмов обрезки дерева.

Также стоит отметить, что деревья решений могут быть легко интерпретируемы, позволяя понять, какие признаки играют важную роль в принятии решений. Это делает их полезными для извлечения правил и понимания взаимосвязей в данных.

1.2.6 Случайный лес

Случайный лес (Random Forest) - это ансамблевый метод, который комбинирует множество решающих деревьев для выполнения классификации или регрессии. Он основывается на принципе усреднения предсказаний отдельных деревьев.

В случайном лесе каждое дерево обучается на различных подмножествах признаков, которые выбираются случайным образом. Этот метод называется методом случайных подпространств. Затем предсказания каждого дерева усредняются для получения итогового предсказания.

Преимущества случайного леса включают высокую точность предсказаний, редкое переобучение, устойчивость к выбросам в данных и хорошую обработку как непрерывных, так и дискретных признаков. Он также хорошо работает с данными, содержащими большое количество признаков. Кроме того, случайный лес обладает высокой параллелизуемостью и масштабируемостью, что позволяет его эффективно использовать на больших наборах данных.

Однако построение случайного леса может занимать больше времени по сравнению с отдельным деревом решений. Интерпретируемость случайного леса также может быть снижена, поскольку он представляет собой комбинацию множества деревьев.

Библиотека scikit-learn (sklearn) предоставляет класс sklearn.ensemble.RandomForestRegressor, который реализует случайный лес для

задачи регрессии. Он позволяет создавать и обучать модели случайного леса для прогнозирования значений целевой переменной на основе входных признаков.

Случайный лес также доступен для задачи классификации с помощью класса sklearn.ensemble.RandomForestClassifier.

1.2.7 Градиентный бустинг

Градиентный бустинг (GradientBoosting) является одним из ансамблевых методов машинного обучения. Он отличается от случайного леса тем, что каждый базовый алгоритм строится последовательно, а не независимо друг от друга. Бустинг направлен на уменьшение ошибки предыдущих алгоритмов.

Для построения алгоритма градиентного бустинга необходимо выбрать базовый алгоритм и функцию потерь (loss function). Функция потерь определяет, насколько хорошо предсказания модели соответствуют данным. С использованием градиентного спуска и скорости обучения (learning rate) мы находим значения, при которых функция потерь минимальна, обновляя предсказания на каждом шаге.

Градиентный бустинг с использованием деревьев решений в качестве базовых алгоритмов называется градиентным бустингом над решающими деревьями. Этот метод хорошо работает на "табличных" данных различной природы и способен находить нелинейные зависимости. Он является одним из самых эффективных алгоритмов машинного обучения и широко применяется в различных конкурсах и задачах.

Однако стоит отметить, что градиентный бустинг может быть времязатратным и требует грамотного подбора гиперпараметров. В данной работе мы используем реализацию градиентного бустинга из библиотеки scikit-learn - sklearn.ensemble.GradientBoostingRegressor.

1.2.8 Нейронная сеть

Нейронная сеть состоит из нейронов или персептронов, которые имеют входы, где сигналы суммируются с учетом весов каждого входа.

Смещение (bias) представляет собой дополнительный вход для нейрона, который всегда равен 1 и имеет свой собственный вес соединения. Функция активации определяет выходное значение нейрона и вводит нелинейность в нейронную сеть. Примеры активационных функций включают ReLU и сигмоиду.

У полносвязной нейронной сети выход каждого нейрона передается на вход всех нейронов следующего слоя. Структура нейронной сети включает входной слой (размерность соответствует входным параметрам), скрытые слои (количество и размерность определяются специалистом) и выходной слой (размерность соответствует выходным параметрам).

Процесс прямого распространения заключается в передаче входных значений в нейронную сеть и получении выходных данных, которые являются прогнозируемым значением. Затем прогнозируемое значение сравнивается с фактическим с помощью функции потери. В методе обратного распространения ошибки градиенты (производные значений ошибок) вычисляются относительно значений весов в направлении, обратном прямому распространению сигналов. Значение градиента вычитается из значения веса для уменьшения ошибки. Этот процесс обновления весов называется обучением модели.

Для обновления весов в модели используются различные оптимизаторы, которые помогают найти минимум функции потерь. Количество эпох указывает, сколько раз выполнено обучение на всех обучающих примерах.

Нейронные сети могут быть применены для решения различных задач, таких как регрессия, классификация, распознавание образов и речи, компьютерное зрение и другие. В настоящее время они являются самым мощным, гибким и широко применяемым инструментом в области машинного обучения.

1.3. Разведочный анализ данных

Задачей разведочного анализа данных является обнаружение закономерностей в предоставленной информации. Чтобы модели работали корректно, требуется, чтобы выходные переменные сильно зависели от входных переменных и не было зависимостей между входными переменными.

На рисунке 4 был представлен график, показывающий взаимосвязь между точками. Однако, по форме "облака точек" мы не заметили таких зависимостей, которые могли бы послужить основой для работы моделей. Для выявления связей между признаками может быть полезной матрица корреляции, которая представлена на рисунке 6.

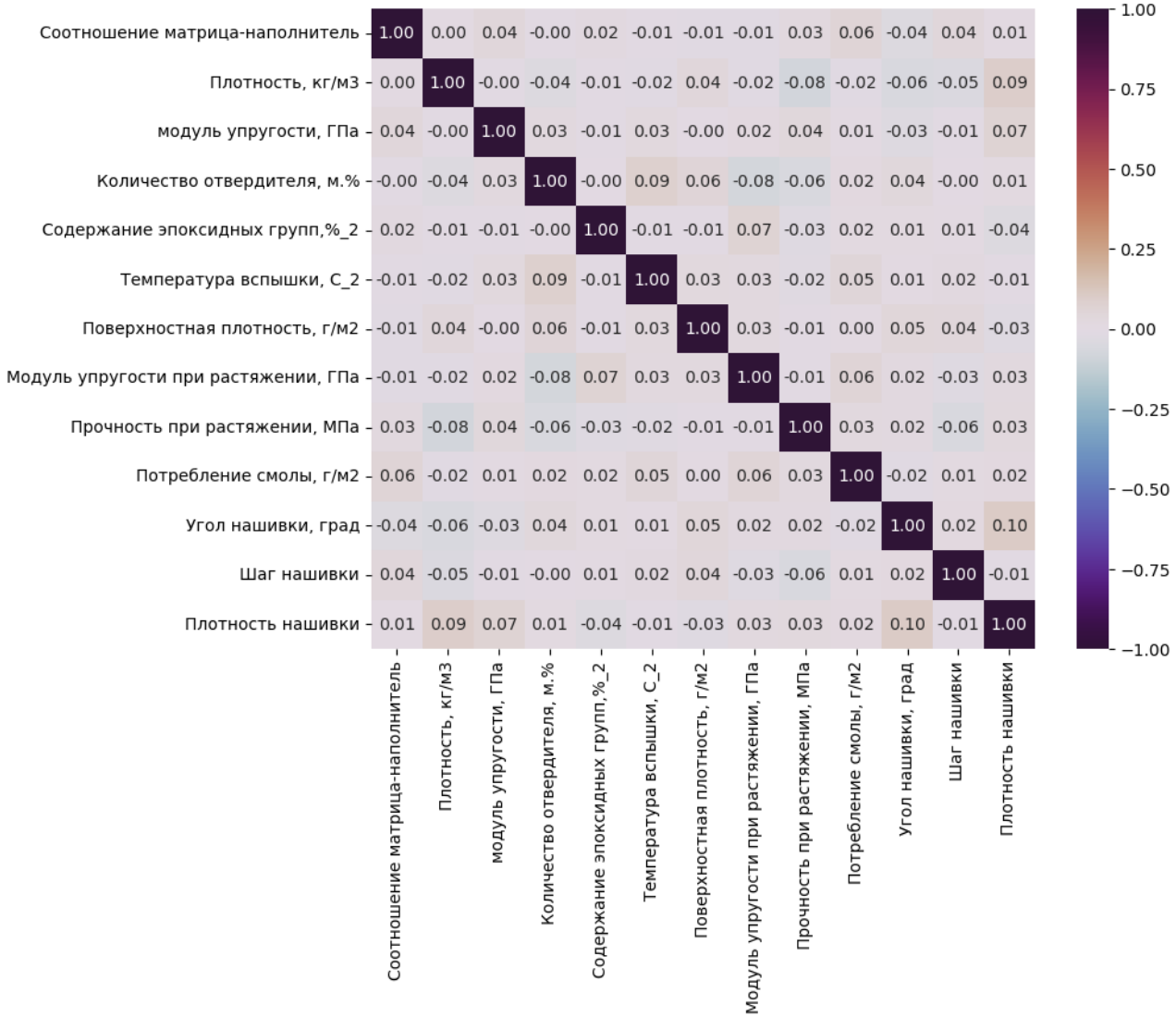

Рисунок 6 — Матрица корреляции

Из матрицы корреляции видно, что все коэффициенты корреляции приближены к нулю, что указывает на отсутствие линейной зависимости между признаками.

1.3.1 Выбор признаков

При использовании статистических методов мы не обнаружили зависимостей между признаками.

В исходных данных у нас было 2 датасета: X_bp с матрицей и X_nup с наполнителем. Их объединение в определённых коэффициентах/

пропорциях порождают КМ с определёнными свойствами. Таким образом категориальный признак соотношения Матрица-наполнитель, судя по всему, это процесс их объединения.

Одна из статей сообщает, что композитные материалы характеризуются следующим набором свойств:

- Не менее двух компонентов в составе, как правило, пластичной основы (матрицы) и наполнителей, обладающих специфическим химическим составом;

- Материалы, образующие эти компоненты, образуют производные свойства материала отличные от свойств исходных компонентов;

- В большом масштабе являются однородными, в маленьком масштабе - неоднородными;

- Свойства материала являются производными свойств исходных компонентов, содержащихся в материале в нужном количестве (больше определённого порогового значения).

Таким образом весь процесс создания композитных материалов делятся на химические свойства:

– пластичной основы (матрицы);
– наполнителей;
– смеси в процессе создания КМ;
– свойства КМ на выходе.

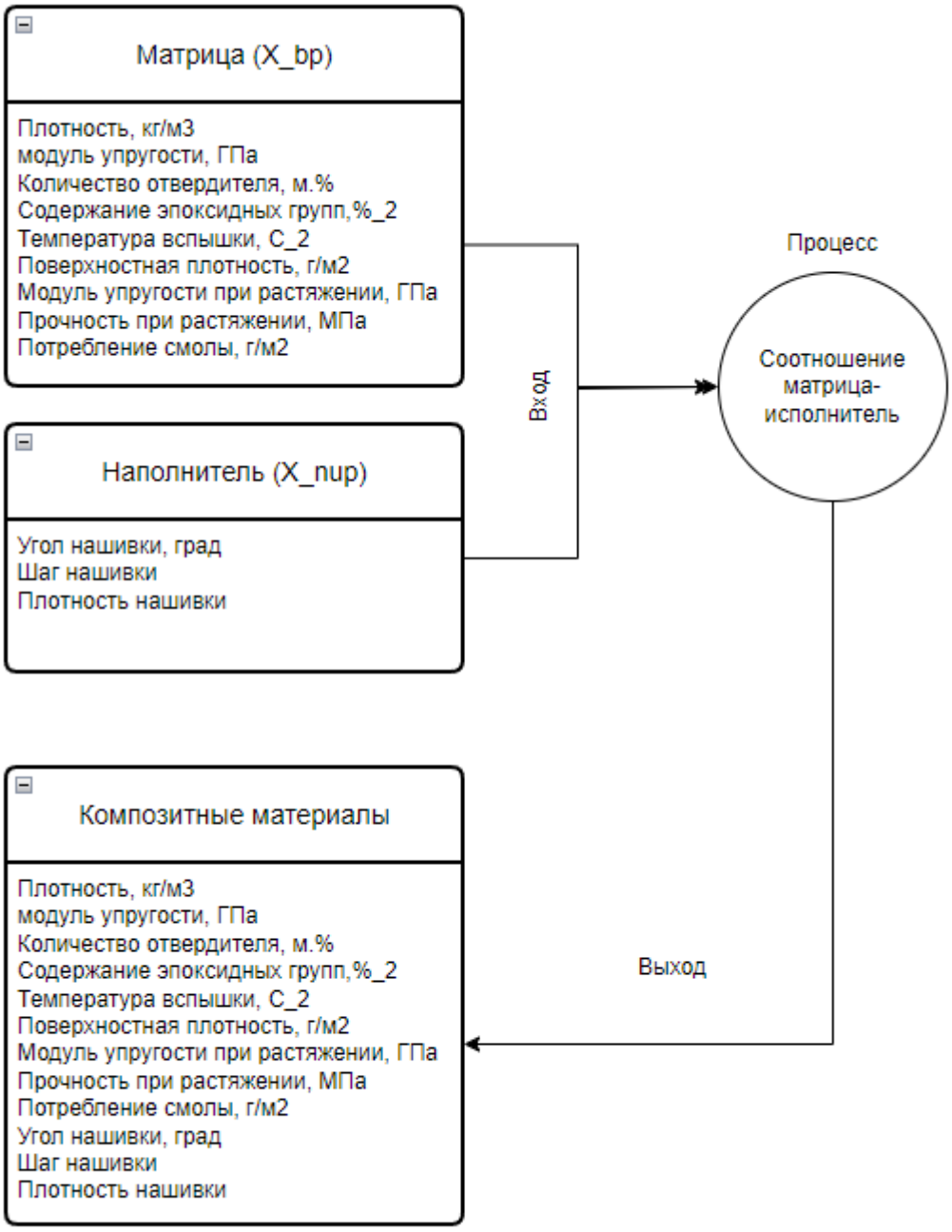

Рисунок 7 — Группы признаков
с точки зрения предметной области

По условиям задания нас интересуют зависимости вида:

модуль упругости при растяжении, Гпа (композит) = f(матрица, наполнитель, процесс);

прочность при растяжении, МПа (композит) = f(матрица, наполнитель, процесс);

соотношение матрица-наполнитель (процесс) = f(матрица, наполнитель, композит).

Для каждого из целевых признаков построим отдельную модель с идентичными признаками сформированного датасета.

1.3.2 Ход решения задачи

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

1. Разделение исходных данные на тренировочную и тестовую выборки. В данном случае, на тестирование отводится 30% данных (в соответствии с условиями, указанными в требованиях к аттестационной работе п.4).

2. Препроцессинг данных – подготовить исходные данные для анализа. В него обычно входят заполнение пропущенных значений, масштабирование признаков и другие необходимые преобразования.

3. Выбрать базовую модель для определения нижней границы качества предсказания. В данном случае, можно использовать модель, которая возвращает среднее значение целевого признака.

4. Выбрать несколько моделей с гиперпараметрами по умолчанию и оценить их метрики на тренировочной выборке, используя перекрестную проверку.

5. Подбор гиперпараметров для выбранных моделей с помощью поиска по сетке и перекрестной проверки. Количество блоков для перекрестной проверки можно выбрать равным 10.

6. Сравнение метрики моделей после подбора гиперпараметров и выбевыбрать лучшую модель на основе их характеристик.

7. Предсказания и выводы - получить предсказания лучшей модели и базовой модели на тестовой выборке и проанализировать результаты с подведением итогов.

8. Сравнить и оценить качество работы лучшей модели на тренировочной и тестовой выборках. Это поможет в оценке переобучения модели и ее обобщающей способности.

После выполнения всех этих шагов возможно выявить оптимальную модель, которая лучше базовой и способна предсказывать целевой признак на тестовых данных.

1.3.2 Препроцессинг

Цель препроцессинга данных состоит в обеспечении корректной работы моделей.

Для категориального признака «Угол нашивки, град», который принимает значения 0 и 90, рекомендуется преобразовать его в значения 0 и 1 с помощью LabelEncoder или OrdinalEncoder. Это может улучшить работу моделей.

У нас также есть множество вещественных количественных признаков. Проблема заключается в том, что их значения находятся в разных диапазонах и масштабах, как видно из таблицы 2. Для решения этой проблемы можно применить одно из двух преобразований:

- Нормализация: приведение значений в диапазон от 0 до 1 с использованием MinMaxScaler.

- Стандартизация: приведение значений к среднему значению 0 и стандартному отклонению 1 с помощью StandardScaler.

В данном случае, предлагается использовать стандартизацию с помощью StandardScaler.

Препроцессинг данных необходимо повторить для введенных данных в приложении. Для удобства, можно реализовать предварительную обработку данных с использованием ColumnTransformer и сохранить/загрузить этот объект аналогично объекту модели.

Выходные переменные остаются без изменений.

Эти шаги препроцессинга позволят нам обеспечить корректную работу моделей на новых данных и получить более точные прогнозы.

1.3.3 Перекрестная проверка

Для надежной оценки метрик качества модели и обеспечения ее статистической устойчивости рекомендуется использовать перекрестную проверку или кросс-валидацию. Для этого выборка разбивается на тестовую и валидационную части нужное количество раз. Модель обучается на тестовой выборке, а затем вычисляются метрики качества на валидационной выборке. Путем усреднения метрик качества всех валидационных выборок получаем окончательный результат. В библиотеке scikit-learn (sklearn) для реализации перекрестной проверки доступна функция cross_validate.

1.3.4 Поиск гиперпараметров по сетке

Для поиска оптимальных гиперпараметров модели можно использовать класс GridSearchCV из библиотеки scikit-learn (sklearn). Этот класс позволяет перебрать заданный набор гиперпараметров, передавая их поочередно в модель, выполнять обучение и определять лучшие комбинации гиперпараметров. Встроенная перекрестная проверка позволяет оценить качество модели для каждой комбинации гиперпараметров и выбрать наилучшую. Таким образом, GridSearchCV позволяет автоматически исследовать пространство гиперпараметров и найти оптимальные значения для модели.

1.3.5 Метрики качества моделей

Существует множество различных метрик качества, которые могут быть применены для оценки регрессионных моделей. В данной работе используются следующие метрики:

- Коэффициент детерминации (R2) измеряет долю объясненной моделью дисперсии в общей дисперсии целевой переменной. Значение R2 близкое к 1

указывает на хорошую способность модели объяснять данные, а значение близкое к 0 означает, что прогнозы модели сопоставимы с константным предсказанием.

- Корень из средней квадратичной ошибки (RMSE) измеряет ошибку модели в тех же единицах, что и целевая переменная. RMSE хорошо обнаруживает грубые ошибки, но является чувствительным к выбросам из-за возведения в квадрат.

- Средняя абсолютная ошибка (MAE) также измеряет ошибку модели в тех же единицах, что и целевая переменная. MAE является более устойчивой к выбросам, поскольку не использует возведение в квадрат.

- Средняя абсолютная процентная ошибка (MAPE) представляет собой безразмерный показатель, который выражает ошибку в процентах. MAPE является взвешенной версией MAE.

- Максимальная ошибка (max error) представляет собой наибольшую ошибку модели в единицах измерения целевой переменной.

RMSE, MAE, MAPE и max error обычно отображаются со знаком «-», чтобы корректно выделить лучшие модели - эти метрики должны быть минимизированы. В то же время, коэффициент детерминации (R2) должен быть максимизирован. Отрицательные значения коэффициента детерминации указывают на плохую объясняющую способность модели.

2. Практическая часть

2.1. Разбиение и предобработка данных

2.1.1 Для прогнозирования модуля упругости при растяжении

Для анализа данных были проведены следующие шаги: разделение признаков на входные и выходные, а также разделение строк на тренировочное и тестовое множество. Размерности полученных наборов данных представлены на рисунке 8. Для входных признаков была проведена описательная статистика до и после предобработки, результаты которой представлены на рисунке 9.

```
x1_train: (700, 11) y1_train: (700, 1)
x1_test: (300, 11) y1_test: (300, 1)
```

Рисунок 8 - Размерности тренировочного и тестового множеств после разбиения для 1-й задачи

```
In [40]: # Описательная статистика входных данных до предобработки
         show_statistics(x1_train_raw)
```

Out[40]:

	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
min	0.389403	1731.764635	2.436909	29.956150	14.254985	100.000000	1.894093	41.048278	0.000000	0.000000	0.000000
max	5.591742	2207.773481	1649.415706	198.953207	33.000000	413.273418	1399.542362	414.590628	90.000000	14.051383	98.202603
mean	2.896388	1975.820146	729.007510	109.933208	22.249999	286.503446	479.540882	219.657689	43.491620	6.884778	57.018189
std	0.913730	72.755603	323.480590	28.404954	2.410708	41.451246	277.257393	59.993071	45.006153	2.549085	12.724849

```
In [41]: # Описательная статистика входных данных после предобработки
         show_statistics(pd.DataFrame(x1_train, columns=(x1_continuous + x_categorical)))
```

Out[41]:

	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	Шаг нашивки	Плотность нашивки	Угол нашивки, град
min	-2.745601	-3.356802	-2.247673	-2.817571	-3.318777	-4.502490	-1.723960	-2.979249	-2.702770	-4.483986	0.000000
max	2.951900	3.190345	2.847316	3.136151	4.462388	3.060429	3.320541	3.251529	2.813407	3.238797	1.000000
mean	-0.000000	0.000000	-0.000000	-0.000000	-0.000000	0.000000	0.000000	0.000000	-0.000000	0.000000	0.483240
std	1.000699	1.000699	1.000699	1.000699	1.000699	1.000699	1.000699	1.000699	1.000699	1.000699	0.500068

Рисунок 9 - Описательная статистика входных признаков до и после предобработки для 1-й задачи

2.1.2 Для прогнозирования прочности при растяжении

Признаки датасета были разделены на входные и выходные, а строки - на тренировочное и тестовое множество. Размерности полученных наборов данных

показаны на рисунке 10. Описательная статистика входных признаков до и после предобработки показана на рисунке 11. Описательная статистика выходного признака показана на рисунке 12.

```
x2_train: (700, 11) y2_train: (700, 1)
x2_test: (300, 11) y2_test: (300, 1)
```

Рисунок 10 - Размерности тренировочного и тестового множеств после разбиения для 2-й задачи

In [100]:
```
# Описательная статистика входных данных до предобработки
show_statistics(x2_train_raw)
```

Out[100]:

	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
min	0.389403	1786.035636	4.339154	32.019222	15.881668	173.484920	1.668002	41.048278	0.000000	0.037639	20.571633
max	5.452959	2184.493200	1649.415706	192.851702	28.955094	403.652861	1291.340115	386.903431	90.000000	14.440522	92.963492
mean	2.920187	1975.744729	735.691153	110.174107	22.136375	285.585994	487.215559	218.527400	44.614286	6.983764	57.487818
std	0.897425	72.314674	326.522982	27.475716	2.376277	40.410590	270.036479	58.268571	45.030523	2.637260	11.910489

In [101]:
```
# Описательная статистика входных данных после предобработки
show_statistics(pd.DataFrame(x2_train, columns=(x2_continuous + x_categorical)))
```

Out[101]:

	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	Шаг нашивки	Плотность нашивки	Угол нашивки, град
min	-2.768251	-2.534001	-2.247658	-2.797457	-2.759118	-2.847876	-1.567569	-2.919113	-2.848029	-3.100593	0.000000
max	2.655332	2.820515	2.740297	2.804565	2.718414	2.935580	2.817645	2.798748	3.272282	3.097995	1.000000
mean	-0.057524	0.015331	-0.030162	-0.075212	-0.138508	-0.031099	0.083417	0.015066	0.103633	0.060373	0.495714
std	0.961233	0.971772	0.990034	0.957018	0.995618	1.015401	0.918193	0.963327	1.120668	1.019841	0.500339

Рисунок 11 - Описательная статистика входных признаков до и после предобработки для 2-й задачи

2.1.3 Для прогнозирования соотношения матрица-наполнитель

Признаки датасета были разделены на входные и выходные, а строки - на тренировочное и тестовое множество. Размерности полученных наборов данных показаны на рисунке 12. Описательная статистика входных признаков до и после предобработки показана на рисунке 13.

```
x3_train: (700, 12) y3_train: (700, 1)
x3_test: (300, 12) y3_test: (300, 1)
```

Рисунок 12 - Размерности тренировочного и тестового множеств после разбиения для 3-й задачи

In [114]:
```
# Описательная статистика входных данных до предобработки
show_statistics(x3_train_raw)
```

Out[114]:

	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	П
min	1786.035636	4.339154	32.019222	15.881668	173.484920	1.668002	64.054061	1036.856605	41.048278	0.000000	0.037639	
max	2184.493200	1649.415706	192.851702	28.955094	403.652861	1291.340115	82.525773	3848.436732	386.903431	90.000000	14.440522	
mean	1975.744729	735.691153	110.174107	22.136375	285.585994	487.215559	73.314828	2448.748866	218.527400	44.614286	6.983764	
std	72.314674	326.522982	27.475716	2.376277	40.410590	270.036479	3.234343	496.835835	58.268571	45.030523	2.637260	

In [115]:
```
# Описательная статистика входных данных после предобработки
show_statistics(pd.DataFrame(x3_train, columns=(x3_continuous + x_categorical)))
```

Out[115]:

	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	Шаг нашивки	Плотность нашивки	на
min	-2.534001	-2.247658	-2.797457	-2.759118	-2.847876	-1.567569	-3.296083	-3.227730	-2.919113	-2.848029	-3.100593	0.
max	2.820515	2.740297	2.804565	2.718414	2.935580	2.817645	3.270433	2.964245	2.798748	3.272282	3.097995	1.
mean	0.015331	-0.030162	-0.075212	-0.138508	-0.031099	0.083417	-0.003970	-0.118303	0.015066	0.103633	0.060373	0.
std	0.971772	0.990034	0.957018	0.995618	1.015401	0.918193	1.149778	1.094187	0.963327	1.120668	1.019841	0.

Рисунок 13 - Описательная статистика входных признаков до и после предобработки для 3-й задачи

2.2 Разработка и обучение моделей для прогнозирования модуля упругости при растяжении

Для выбора лучшей модели в данной задаче были рассмотрены следующие модели:

- Линейная регрессия (LinearRegression) (раздел 1.2.1)
- Гребневая регрессия (Ridge) (раздел 1.2.2)
- Лассо-регрессия (Lasso) (раздел 1.2.2)
- Метод опорных векторов (SVM) (раздел 1.2.3)
- Метод ближайших соседей (KneighborsRegressor) (раздел 1.2.4)
- Деревья решений (DecisionTreeRegressor) (раздел 1.2.5)
- Случайный лес (RandomForestRegressor) (раздел 1.2.6)

В качестве базовой модели использовался DummyRegressor, который возвращает среднее значение целевого признака.

Результаты работы выбранных моделей с гиперпараметрами по умолчанию, полученные с помощью перекрестной проверки на тестовом множестве, представлены на рисунке 14.

Ни одна из выбранных моделей не подходит для данных. Хуже всего отработали метод ближайших соседей и деревья решений. Остальные метрики по показателям примерно идентичны базовой модели.

Out[87]:

	R2	RMSE	MAE	MAPE	max_error
DummyRegressor	-0.008541	-3.214706	-2.615130	-0.035714	-7.743410
LinearRegression	-0.011794	-3.219895	-2.636183	-0.035999	-7.700395
Ridge	-0.011738	-3.219809	-2.636116	-0.035998	-7.700277
Lasso	-0.008541	-3.214706	-2.615130	-0.035714	-7.743410
SVR	-0.047251	-3.276116	-2.674892	-0.036522	-8.002309
KNeighborsRegressor	-0.226161	-3.547397	-2.852391	-0.038955	-9.150680
DecisionTreeRegressor	-1.062683	-4.573940	-3.622630	-0.049443	-12.208512
RandomForestRegressor	-0.054466	-3.290492	-2.669989	-0.036485	-8.233748

Рисунок 15 — Результаты моделей с гиперпараметрами по умолчанию

Out[94]:

	R2	RMSE	MAE	MAPE	max_error
Ridge(alpha=1500, positive=True, solver='lbfgs')	-0.005731	-3.210099	-2.616392	-0.035731	-7.739360
Lasso(alpha=0.5)	-0.008541	-3.214706	-2.615130	-0.035714	-7.743410
SVR(C=0.001, kernel='poly')	-0.010645	-3.218309	-2.615569	-0.035658	-7.811671
KNeighborsRegressor(n_neighbors=23)	-0.019579	-3.233235	-2.638854	-0.036030	-8.057815
DecisionTreeRegressor(criterion='absolute_error', max_depth=1, max_features=4, random_state=4096, splitter='random')	-0.009392	-3.216400	-2.612219	-0.035609	-7.826138
RandomForestRegressor(bootstrap=False, criterion='absolute_error', max_depth=2, max_features=1, random_state=4096)	-0.016234	-3.228020	-2.634513	-0.035930	-7.795222

Рисунок 16 — Результаты моделей после подбора гиперпараметров

После выполнения подбора гиперпарметров по сетке с перекрестной проверкой, получили метрики, приведенные на рисунке 18.

Из представленных результатов следуют следующие выводы:

Гребнивая линейная (Ридж) модель, а также Деревья решений (DecisionTreeRegressor) показали результаты близкие к линейной.

Результатов, которые могли бы объяснить какую-либо зависимость не выявлено.

Остальные модели отработали хуже. В качестве лучшей модели выбрано дерево решений. На рисунке 17 приведена визуализация работы лучшей модели на тестовом множестве.

Поэтому в качестве лучшей модели выбираю дерево решений. На рисунке 19 приведена визуализация работы лучшей модели на тестовом множестве.

В целом, модель неплохо отработала, имеются небольшие разбросы.

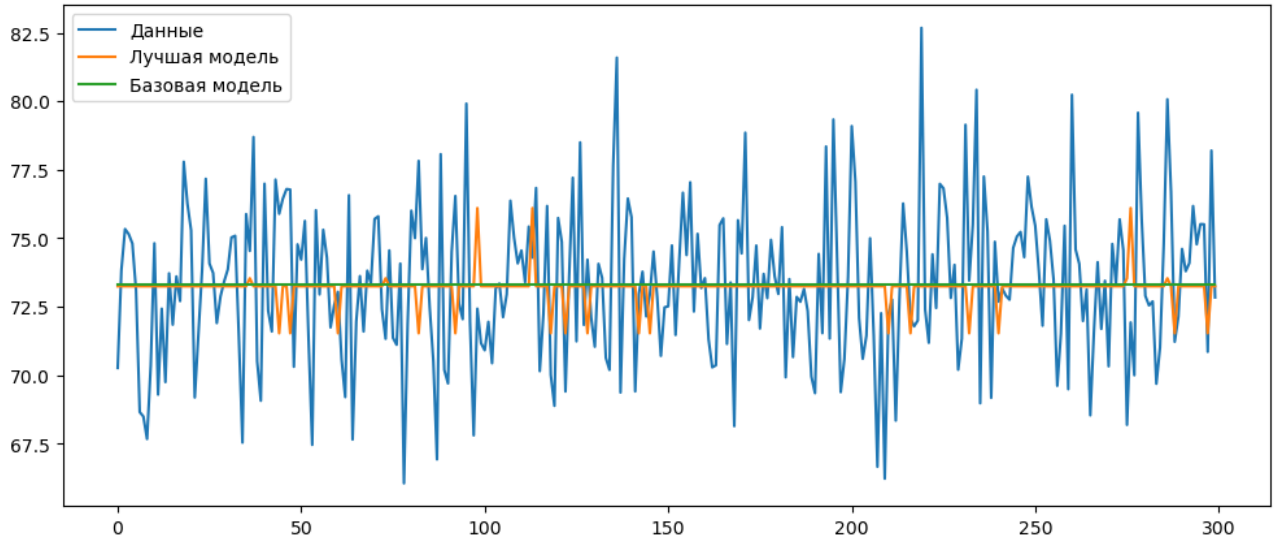

Рисунок 17 — Визуализация работы модели

Out[117]:

	R2	RMSE	MAE	MAPE	max_error
Базовая модель	-0.000016	-2.813038	-2.221060	-0.030326	-9.367223
Лучшая модель (дерево решений)	-0.033680	-2.859995	-2.265207	-0.030879	-9.434457

Рисунок 18 - Метрики работы лучшей модели на тестовом множестве

Метрики работы лучшей модели на тестовом множестве и сравнение с базовой отражены на рисунке 18. Полученная модель хуже базовой, несмотря на неплохие результаты, в любом случае данный результат нельзя считать положительным исходом. При проведении исследования не удалось найти модель,

которая могла бы предоставить ценную информацию для принятия решений специалисту в данной предметной области.

2.3 Для прогнозирования прочности при растяжении

Для выбора лучшей модели в данной задаче были рассмотрены следующие модели:

- Линейная регрессия (LinearRegression) (раздел 1.2.1)
- Гребневая регрессия (Ridge) (раздел 1.2.2)
- Лассо-регрессия (Lasso) (раздел 1.2.2)
- Метод опорных векторов (SVM) (раздел 1.2.3)
- Деревья решений (DecisionTreeRegressor) (раздел 1.2.5)
- Случайный лес (GradientBoostingRegressor) (раздел 1.2.7)

В качестве базовой модели использовался DummyRegressor, который возвращает среднее значение целевого признака.

Результаты работы выбранных моделей с гиперпараметрами по умолчанию, полученные с помощью перекрестной проверки на тестовом множестве, представлены на рисунке 19.

Out[99]:

	R2	RMSE	MAE	MAPE	max_error
DummyRegressor	-0.010129	-495.586751	-393.601038	-0.175633	-1286.268912
LinearRegression	-0.029041	-500.175134	-398.813241	-0.177633	-1282.696852
Ridge	-0.028982	-500.160854	-398.798869	-0.177627	-1282.687326
Lasso	-0.027703	-499.840800	-398.493511	-0.177517	-1281.565588
SVR	-0.009895	-495.542965	-393.980277	-0.175522	-1287.507718
DecisionTreeRegressor	-1.164180	-722.120215	-586.343204	-0.254223	-1676.702403
GradientBoostingRegressor	-0.069271	-509.375953	-407.996580	-0.181359	-1334.490077

Рисунок 19 — Результаты моделей с гиперпараметрами по умолчанию

Лучшие результаты показал метод опорных векторов, Градиентный бугстинг с параметрами по умолчанию отработал лучше дерева, но хуже SVM. Он тоже соответствует базовой модели.

После выполнения подбора гиперпарметров по сетке с перекрестной проверкой, получили метрики, приведенные на рисунке 22.

	R2	RMSE	MAE	MAPE	max_error
Ridge(alpha=990, positive=True, solver='lbfgs')	-0.013191	-496.310896	-394.202197	-0.175900	-1291.691176
Lasso(alpha=40)	-0.010129	-495.586751	-393.601038	-0.175633	-1286.268912
SVR(C=0.001, kernel='poly')	-0.009841	-495.519328	-393.918641	-0.175532	-1287.893713
DecisionTreeRegressor(criterion='poisson', max_depth=3, max_features=3, random_state=4096)	0.010102	-490.335976	-389.529097	-0.172543	-1254.310724
GradientBoostingRegressor(max_depth=2, max_features=1, n_estimators=50, random_state=4096)	-0.004168	-494.235639	-393.624265	-0.175098	-1276.953395

Рисунок 20 — Результаты моделей после подбора гиперпараметров

Применены несколько линейных моделей и ансамбли. Результаты по-прежнему крайне плохо описывают исходные данные и не могут применяться в приложении. Модель Деревья решений лучшая по показателям. Применён GradientBoosting, но показатели немного хуже, в качестве эксперимента визуализируем градиентный бустинг (Рисунок 21) и деревья решений (Рисунок 22).

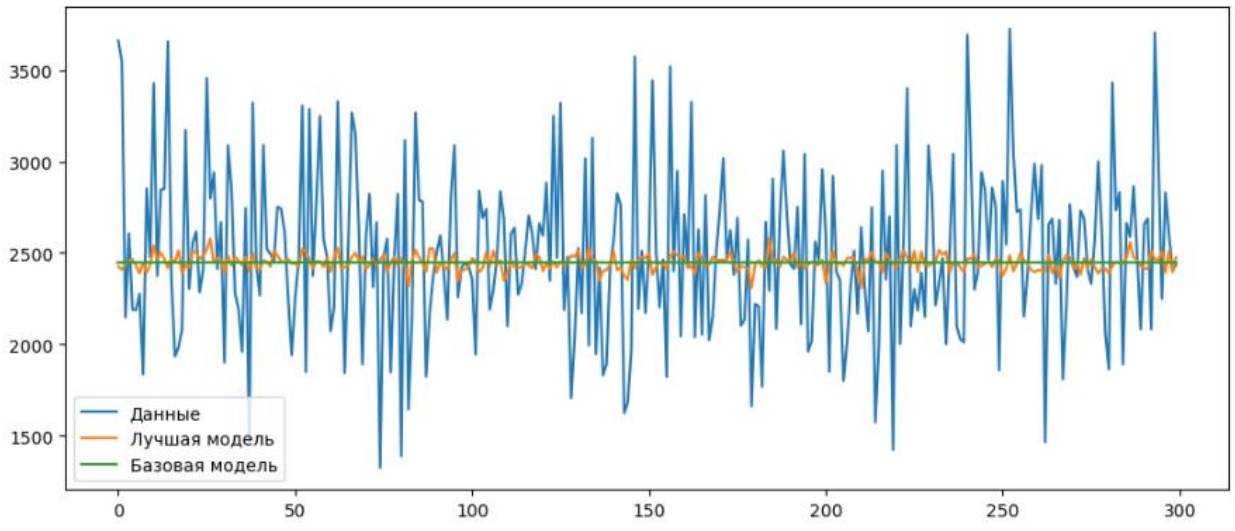

Рисунок 21 — Визуализация работы модели градиентный бустинг

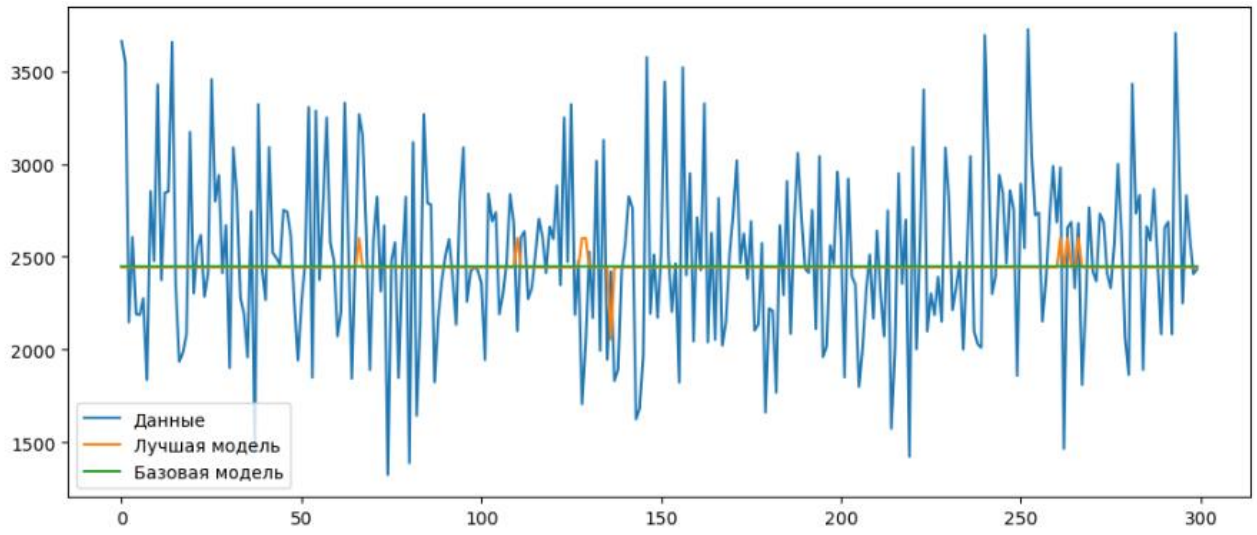

Рисунок 22 — Визуализация работы модели Деревья решений

Далее приведём метрики лучший моделей по каждому из 2 вариантов.

	R2	RMSE	MAE	MAPE	max_error
Базовая модель	-0.013996	-457.23484	-356.545181	-0.147661	-1276.441894
Лучшая модель (градиентный бустинг)	0.001457	-453.73746	-355.998769	-0.147096	-1239.629532

Рисунок 23 - Метрики работы лучшей модели градиентный бустинг

	R2	RMSE	MAE	MAPE	max_error
Базовая модель	-0.013996	-457.234840	-356.545181	-0.147661	-1276.441894
Лучшая модель (Деревья решений)	-0.020139	-458.617808	-357.596120	-0.148075	-1281.707872

Рисунок 24 - Метрики работы лучшей модели на тестовом множестве деревья решений

Метрики модели деревья решений ближе к базовой, хотя на градиентном бустинге коэффициент детерминации на градиентном бустинге лучше. Остальные модели отработали значительно хуже. Показателей, которые могли бы объяснить нам какую-либо зависимость – не выявлено.

2.4 Разработка нейронной сети для прогнозирования соотношения матрица-наполнитель

В соответствии с заданием, требуется построить нейросеть для соотношения матрица-наполнитель. Однако, для целей сравнения, также потребуется использовать базовую модель DummyRegressor, которая возвращает среднее значение целевого признака.

2.4.1 MLPRegressor из библиотеки sklearn

Создаём нейронную сеть, используя класс MLPRegressor, с следующей архитектурой:

- Количество слоев: 8;
- Количество нейронов на каждом слое: 24;
- Активационная функция: relu;
- Оптимизатор: adam;
- Пропорция разбиения данных на тестовую и валидационную выборки в соответствии с заданием: 30%;
- Ранняя остановка обучения, если метрики на валидационной выборке не улучшаются;
- Количество итераций: 5000.

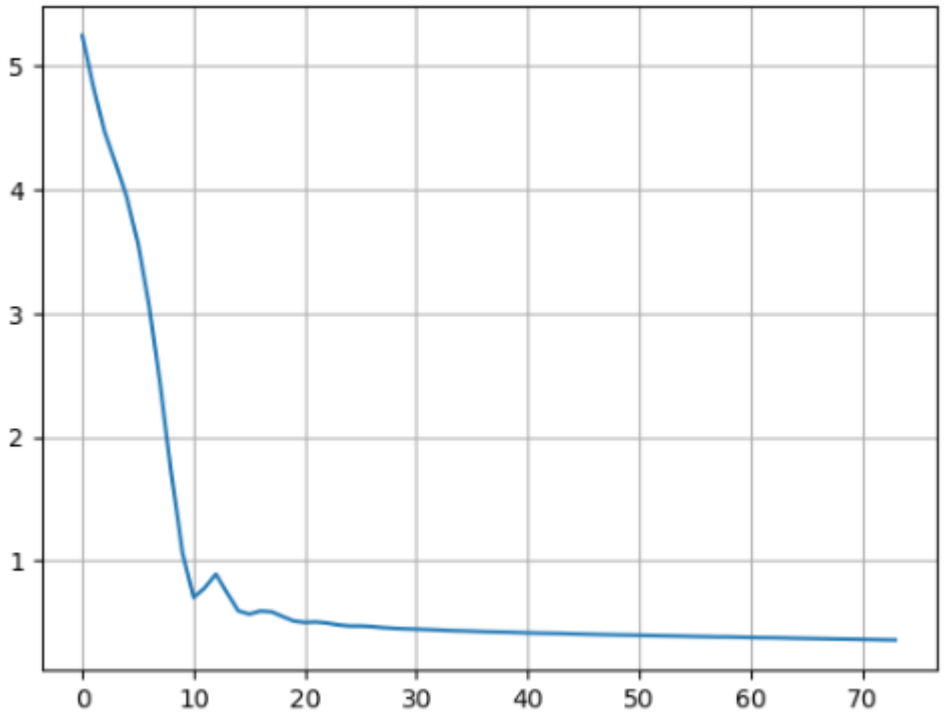

Рисунок 25 — График обучения MLPRegressor

Результаты, полученные нейросетью, визуализированы на рисунке 26. Очевидно, что нейросеть пыталась адаптироваться к исходным данным, однако результаты оказались не удовлетворительными.

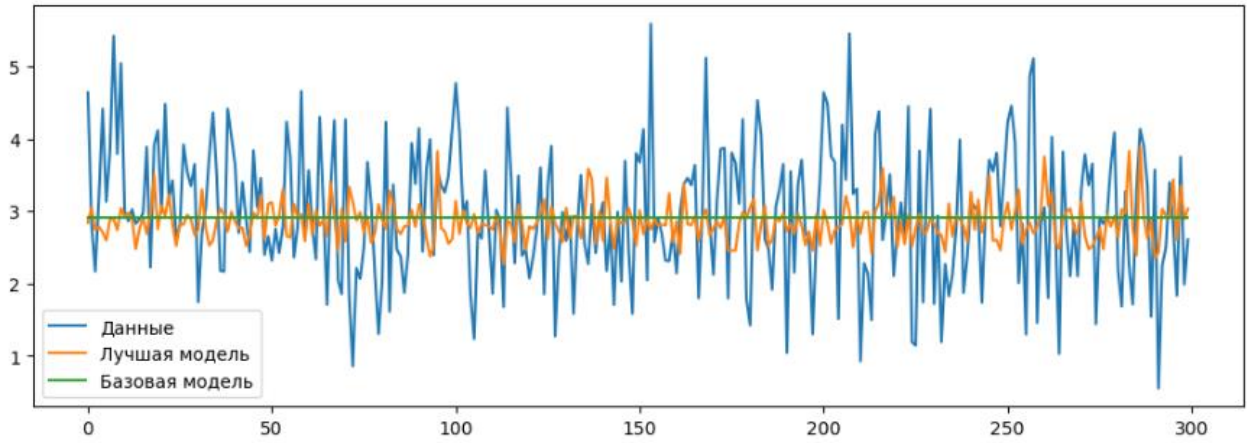

Рисунок 26 — Визуализация работы модели

Метрики работы нейросети MLPRegressor на тестовом множестве и сравнение с базовой моделью отражены на рисунке 27. Исходя из представленной визуализации видим, что MLPRegressor выдаёт ошибку практически по всем метрикам хуже, чем у базовой модели. Попробуем построить сеть с помощью TensorFlow.

	R2	RMSE	MAE	MAPE	max_error
DummyRegressor	-0.003309	-0.935162	-0.757702	-0.317726	-2.671554
MLPRegressor	-0.047629	-0.955594	-0.774743	-0.312176	-2.843771

Рисунок 27 — Метрики работы нейросети MLPRegressor на тестовом множестве

2.4.2 Нейросеть из библиотеки tensorflow

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

– входной слой для 12 признаков;

– выходной слой для 1 признака;

– скрытых слоев: 8;

– нейронов на каждом скрытом слое: 24;

– активационная функция скрытых слоев: relu;

– оптимизатор: Adam;

– loss-функция: MeanAbsolutePercentageError.

Архитектура нейросети приведена на рисунке 28.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_1 (Dense)             (None, 24)                312

 dense_2 (Dense)             (None, 24)                600

 dense_3 (Dense)             (None, 24)                600

 dense_4 (Dense)             (None, 24)                600

 dense_5 (Dense)             (None, 24)                600

 dense_6 (Dense)             (None, 24)                600

 dense_7 (Dense)             (None, 24)                600

 dense_8 (Dense)             (None, 24)                600

 out (Dense)                 (None, 1)                 25

=================================================================
Total params: 4537 (17.72 KB)
Trainable params: 4537 (17.72 KB)
Non-trainable params: 0 (0.00 Byte)
```

Рисунок 28 — Архитектура нейросети в виде summary

Запускаю бучение нейросети с использованием следующих параметров:

– пропорция разбиения данных на тестовые и валидационные: 30%;

– количество эпох: 50.

– раннюю остановку не используем.

График обучения приведен на рисунке 30, процент ошибки – 34.90, время обучения – 4.05 секунд.

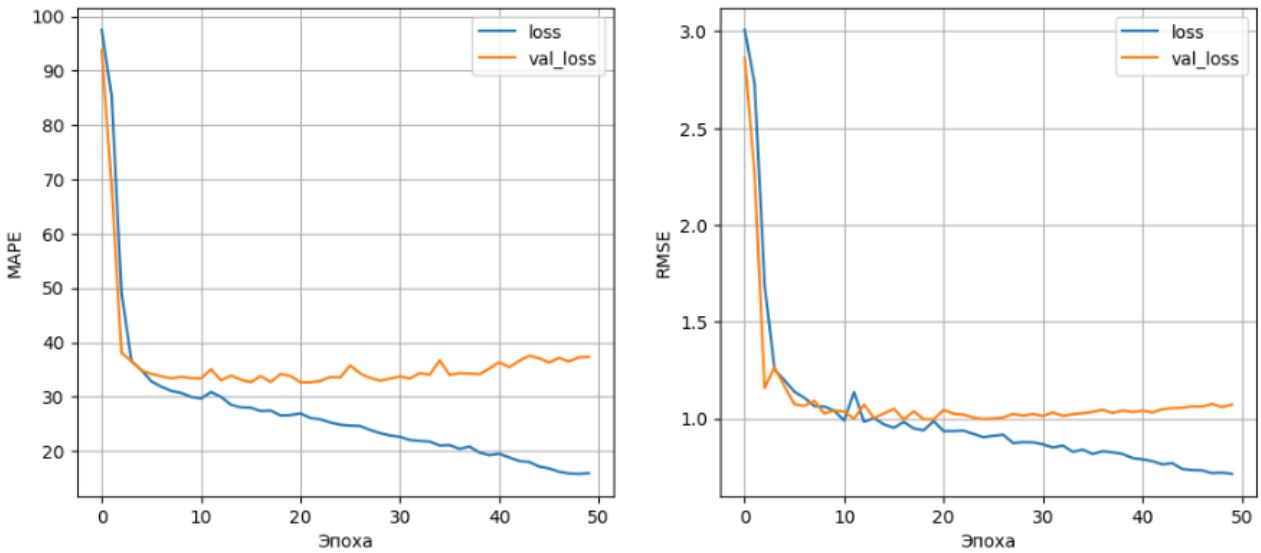

Рисунок 30 — График обучения нейросети

Следует отметить, что обучение шло хорошо примерно до 4-5 эпохи, а потом сеть начала переобучаться. Значение потерь (loss) на тестовой выборке продолжало уменьшаться, в то время как на валидационной выборке начало происходить увеличение.

Для борьбы с переобучением одним из методов является ранняя остановка обучения, основанная на следующем принципе: если значение функции потерь на валидационной выборке начинает расти, обучение останавливается. В TensorFlow для реализации этого метода используются обратные вызовы (callbacks).

Для применения ранней остановки, создается нейросеть с той же архитектурой, и запускается обучение с использованием этого метода. График обучения представлен на рисунке 31, где можно заметить, что использование ранней остановки позволяет улучшить точность модели на новых данных.

Еще одним методом борьбы с переобучением является добавление Dropout-слоев. Построим модель аналогичной архитектуры, только после каждого скрытого слоя добавим слой Dropout с параметром 0.05. Такой слои слои выключат 5% случайных нейронов на каждом слое.

График обучения приведен на рисунке 32, а все показатели ошибок сведены — в таблице 2. Видно, что Dropout-слои справились с переобучением.

Использование ранней остановки сокращает время на обучение модели, а использование Dropout увеличивает. Но уменьшается риск, что мы остановились слишком рано.

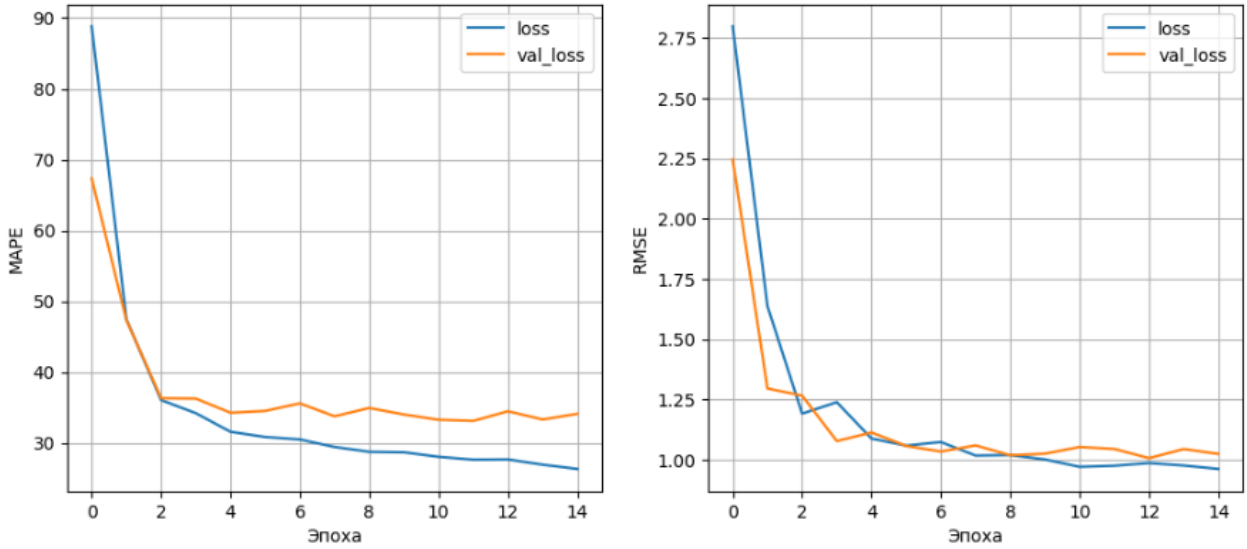

Рисунок 31 — График обучения нейросети с ранней остановкой

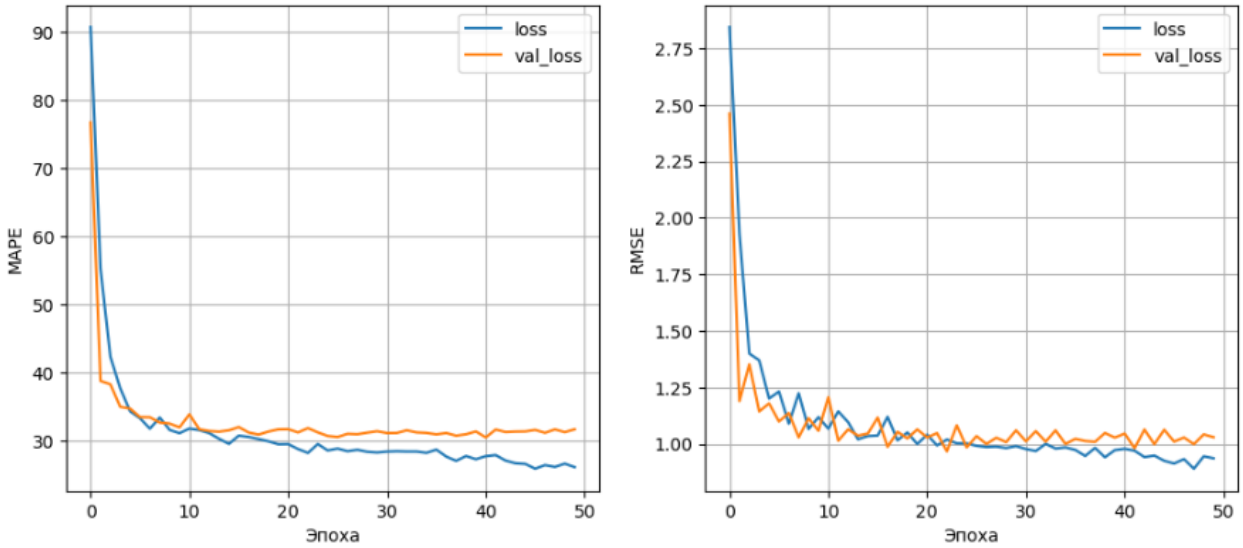

Рисунок 32 — График обучения нейросети с Dropout-слоем

Таблица 2. Борьба с переобучением нейросети

	Эпох	Ошибка на тестовых данных, %	Время обучения, с

Нейросеть переобученная	50	34.90	4.05
Нейросеть с ранней остановкой	12	31.14	1.86
Нейросеть с dropout-слоями	50	32.15	4.2

Визуализация результатов работы нейросетей отображена на рисунке 33, а их метрики — на рисунке 34.

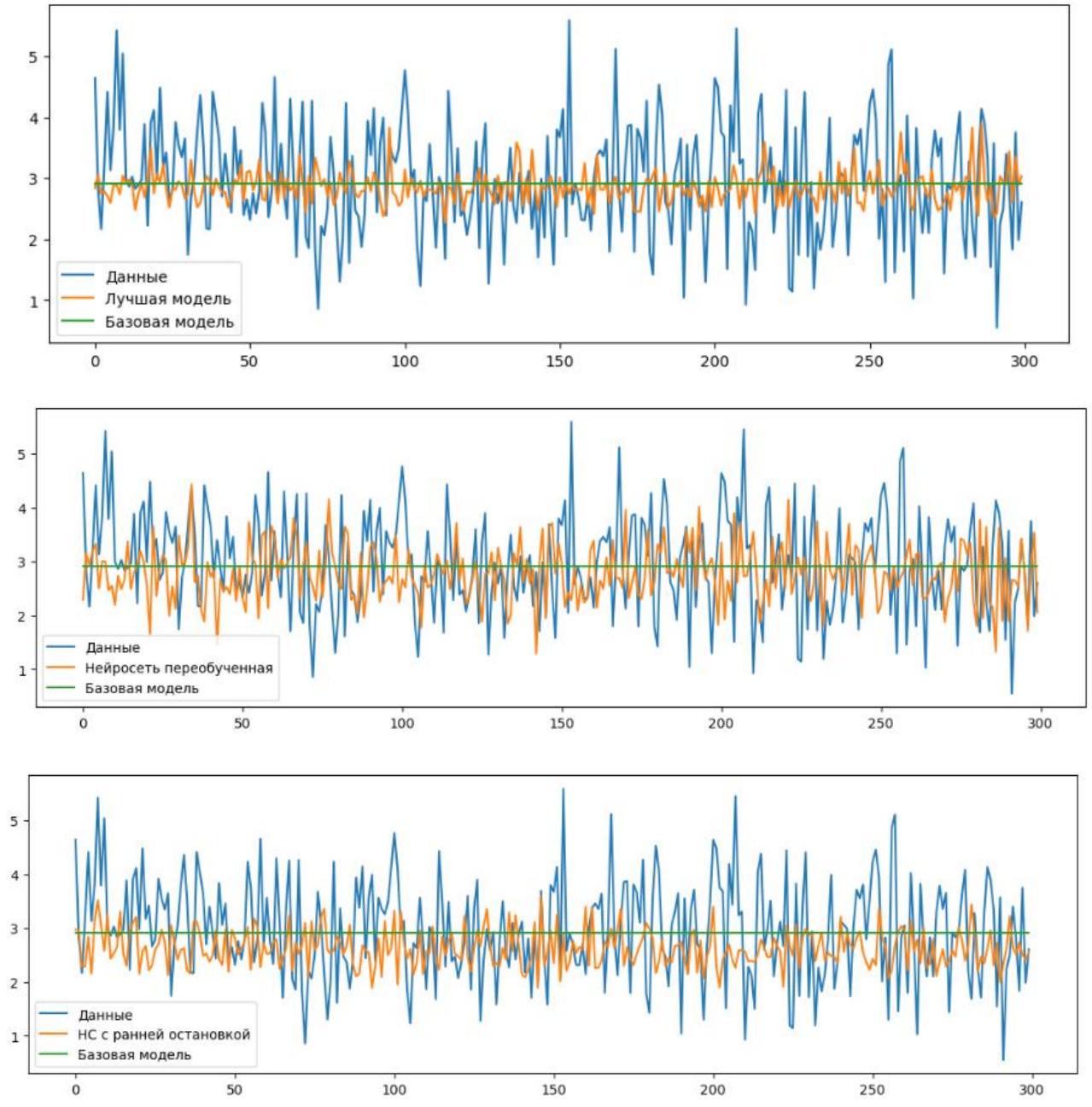

Рисунок 33 - Визуализация результатов работы нейросетей

	R2	RMSE	MAE	MAPE	max_error
DummyRegressor	-0.003309	-0.935162	-0.757702	-0.317726	-2.671554
НС переобученная	-0.446144	-1.122730	-0.890656	-0.355484	-3.279964
НС с ранней остановкой	-0.241443	-1.040238	-0.841552	-0.323412	-2.856988
НС dropout	-0.484030	-1.137341	-0.903828	-0.328172	-3.146087

Рисунок 34 -Метрики работы нейросетей на тестовом множестве

Визуализация результатов показывает, что нейросеть, созданная с использованием библиотеки TensorFlow, старалась подстроиться к данным. Нейросеть, обученная с использованием ранней остановки, показала лучшую обобщающую способность и имела меньшие значения ошибок на тестовом наборе данных. Однако, эта нейросеть предсказывает значительно хуже, чем базовая модель.