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

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по курсу

«Data Science Pro»

Слушатель Поникаровских Андрей Александрович

Москва, 2025

Содержание

Введение

Темой данной выпускной квалификационной работы является прогнозирование конечных свойств новых материалов (композиционных материалов).

Современные требования к материалам в аэрокосмической, автомобильной, энергетической и строительной отраслях всё чаще выходят за рамки возможностей традиционных металлов, сплавов и полимеров. В ответ на эти вызовы широкое распространение получили композиционные материалы — искусственные многокомпонентные системы, состоящие из двух или более фаз, физически разделяемых на макроуровне или микроуровне, но совместно работающих как единое целое.

Композиты, как правило, включают матрицу (связующее вещество, например, полимерную, металлическую или керамическую) и наполнитель (армирующий элемент — волокна, частицы, ткани), который обеспечивает основные механические характеристики. Матрица и наполнитель разделены границей (поверхностью) раздела. Это неоднородные по химическому составу и структуре материалы. Наполнитель равномерно распределен в матрице и имеет заданную пространственную ориентацию.

Благодаря синергетическому сочетанию компонентов, композиционные материалы обладают уникальным набором свойств: высокой удельной прочностью и жёсткостью, коррозионной стойкостью, малой плотностью, возможностью управления анизотропией свойств и адаптацией под конкретные условия эксплуатации. Композиционные материалы характеризуются совокупностью свойств, не присущих каждому в отдельности взятому компоненту. За счет выбора 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.

Существует возможность получить композиты с уникальными эксплуатационными свойствами. Этим обусловлено широкое применение

композиционных материалов в различных областях техники. Композиционные материалы используются:

- в авиационной, ракетной и космической технике;
- в металлургии;
- в горнорудной промышленности;
- в химической промышленности;
- в автомобильной промышленности;
- в сельскохозяйственном машиностроении;
- в электротехнической промышленности;
- в ядерной технике;
- в машиностроительной отрасли;
- в сварочной технике;
- в судостроительной промышленности;
- в медицинской промышленности;
- в строительстве;
- в бытовой технике.

Однако достижение заданных конечных эксплуатационных свойств композита — нетривиальная задача. Эти свойства чувствительно зависят от множества технологических и составных параметров:

- соотношения матрица-наполнитель,
- типа и ориентации армирующих волокон,
- плотности и шага нашивки (в случае трёхмерных преформ),
- содержания функциональных добавок (например, отвердителей или эпоксидных групп),
- режимов отверждения и термообработки.

Даже незначительные отклонения в составе или технологии могут привести к существенному разбросу в механических характеристиках, таких как модуль упругости при растяжении и прочность при растяжении, что критично для ответственных конструкций. Традиционный подход — экспериментальный

подбор параметров — требует значительных временных и материальных затрат, а также генерирует большое количество отходов. Стоимость производства композитного материала высока. Зная характеристики компонентов, невозможно рассчитать свойства композита. Значит, для получения заданных свойств требуется большое количество испытаний различных комбинаций.

В этой связи актуальной становится задача предиктивного моделирования свойств композиционных материалов на основе данных. Применение методов машинного обучения позволяет построить функциональные зависимости между входными технологическими параметрами и конечными характеристиками материала, минимизируя количество физических испытаний, сокращая время и затраты, и ускоряя разработку новых композитов с заданными свойствами.

Целью данной выпускной квалификационной работы является разработка и внедрение программного инструмента на основе моделей машинного обучения для прогнозирования ключевых механических свойств композиционных материалов, а также обратного прогнозирования оптимального состава по заданным характеристикам. Решение этой задачи способствует цифровизации материаловедения и повышению эффективности проектирования композитных изделий. Учитывая такое широкое распространение и высокую потребность в новых материалах, тема данной работы является актуальной.

В процессе исследовательской работы были разработаны несколько моделей, способных с высокой вероятностью прогнозировать модули упругости при растяжении и прочности при растяжении, а также были созданы несколько нейронных сетей, которые предлагают соотношение «матрица - наполнитель». На основе одной из нейронных сетей было создано дружелюбное и доступное пользовательское веб - приложение с высоким юзабилити на фреймворке Flask: оно размещено на github.com, а также на веб-хостинге render.com.

1 Аналитическая часть

1.1 Постановка задачи

В рамках данной работы рассматривается композиционный материал, в котором матрица выполнена из базальтопластика, а армирование обеспечивается углепластиковыми нашивками. От экспертов в области материаловедения был предоставлен датасет, включающий информацию о характеристиках матрицы и наполнителя, технологических параметрах производства, а также измеренных свойствах готового композита. Поставлена задача — построить предиктивные модели, способные оценивать одни свойства композита на основе других входных признаков. Кроме того, необходимо разработать веб-приложение, обеспечивающее удобный и интуитивно понятный интерфейс для использования этих моделей экспертами в предметной области.

Датасет состоит из двух файлов:

1) X_bp.xlsx (с данными о параметрах базальтопластика), имеющий 10 признаков и индексный столбец, 1023 строк;
2) X_nup.xlsx (с данными нашивок углепластика), имеющий 3 признака и индексный столбец, 1040 строк.

Согласно заданию, файлы требуют объединения с типом INNER по индексному столбцу. После объединения часть строк из файла X_nup, а именно 17 строк таблицы способов компоновки композитов, не имеют соответствующих строк в таблице соотношений и свойств используемых компонентов композитов, поэтому были удалены. Дальнейшие исследования проводим с объединенным датасетом, содержащим 13 признаков и 1023 строк или объектов.

Описание признаков объединенного датасета приведено в таблице 1. Пропусков в данных нет. Дубликатов в данных нет. Большинство признаков имеют тип float64, то есть вещественный, являются непрерывными, количественными, кроме «Угол нашивки» - тип int64, целый, который принимает только два значения и будет рассматриваться как категориальный признак.

Таблица 1 — Описание признаков датасета

Название	Файл	Тип данных	Непустых значений	Уникальных значений
Соотношение матрица-наполнитель	X_bp	float64	1023	1014
Плотность, кг/м3	X_bp	float64	1023	1013
модуль упругости, ГПа	X_bp	float64	1023	1020
Количество отвердителя, м.%	X_bp	float64	1023	1005
Содержание эпоксидных групп,%_2	X_bp	float64	1023	1004
Температура вспышки, С_2	X_bp	float64	1023	1003
Поверхностная плотность, г/м2	X_bp	float64	1023	1004
Модуль упругости при растяжении, ГПа	X_bp	float64	1023	1004
Прочность при растяжении, МПа	X_bp	float64	1023	1004
Потребление смолы, г/м2	X_bp	float64	1023	1003
Угол нашивки, град	X_nup	int64	1023	2
Шаг нашивки	X_nup	float64	1023	989
Плотность нашивки	X_nup	float64	1023	988

Описательная статистика, приведенная на рисунке 1, позволяет принять решение о необходимости нормализации и оценить распределение целевых переменных (например, прочности при растяжении) — это влияет на выбор метрик и алгоритмов, а также в веб-интерфейсе реализовать проверку входных данных на соответствие допустимым диапазонам (например, «Плотность: 1700–2300

кг/м³»). Эти границы определяются именно на основе описательной статистики и экспертных знаний — чтобы пользователь не вводил физически нереалистичные значения, которые модель не сможет корректно обработать. Описательная статистика — это основа для обеспечения физической корректности, надёжности моделей и удобства взаимодействия с приложением.

Описательная статистика

```
# Посмотрю описательную статистику
df_descr = df.describe().T
df_descr['median'] = df.median()
df_descr.style.format(precision=4)
```

	count	mean	std	min	25%	50%	75%	max	median
Соотношение матрица-наполнитель	1023.0000	2.9304	0.9132	0.3894	2.3179	2.9069	3.5527	5.5917	2.9069
Плотность, кг/м3	1023.0000	1975.7349	73.7292	1731.7646	1924.1555	1977.6217	2021.3744	2207.7735	1977.6217
модуль упругости, ГПа	1023.0000	739.9232	330.2316	2.4369	500.0475	739.6643	961.8125	1911.5365	739.6643
Количество отвердителя, м.%	1023.0000	110.5708	28.2959	17.7403	92.4435	110.5648	129.7304	198.9532	110.5648
Содержание эпоксидных групп,%_2	1023.0000	22.2444	2.4063	14.2550	20.6080	22.2307	23.9619	33.0000	22.2307
Температура вспышки, С_2	1023.0000	285.8822	40.9433	100.0000	259.0665	285.8968	313.0021	413.2734	285.8968
Поверхностная плотность, г/м2	1023.0000	482.7318	281.3147	0.6037	266.8166	451.8644	693.2250	1399.5424	451.8644
Модуль упругости при растяжении, ГПа	1023.0000	73.3286	3.1190	64.0541	71.2450	73.2688	75.3566	82.6821	73.2688
Прочность при растяжении, МПа	1023.0000	2466.9228	485.6280	1036.8566	2135.8504	2459.5245	2767.1931	3848.4367	2459.5245
Потребление смолы, г/м2	1023.0000	218.4231	59.7359	33.8030	179.6275	219.1989	257.4817	414.5906	219.1989
Угол нашивки, град	1023.0000	44.2522	45.0158	0.0000	0.0000	0.0000	90.0000	90.0000	0.0000
Шаг нашивки	1023.0000	6.8992	2.5635	0.0000	5.0800	6.9161	8.5863	14.4405	6.9161
Плотность нашивки	1023.0000	57.1389	12.3510	0.0000	49.7992	57.3419	64.9450	103.9889	57.3419

Рисунок 1 - Описательная статистика

Гистограммы распределения переменных приведены на рисунке 2, диаграммы «ящик с усами» приведены на рисунке 3. По ним видно, что все признаки, кроме «Угол нашивки», имеют нормальное распределение и принимают неотрицательные значения. «Угол нашивки» принимает значения: 0 или 90. Описательная статистика в численном виде отражает то, что мы видим на гистограммах.

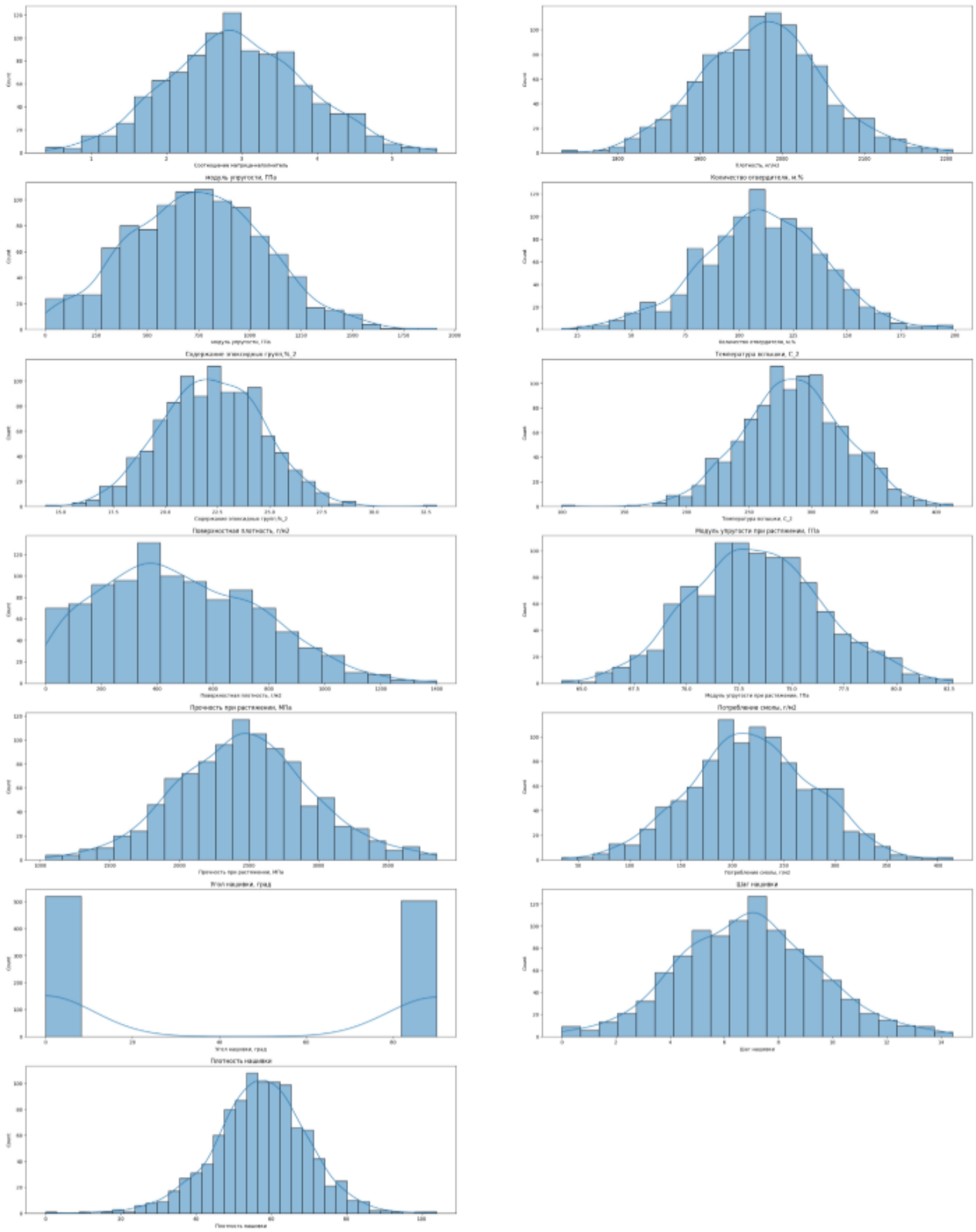

Рисунок 2 - Гистограммы распределения переменных

По условиям задания датасет был предварительно подготовлен вследствие чего имеет место отсутствие пропусков. В сырых данных пропуски и значения некорректных типов как правило присутствуют.

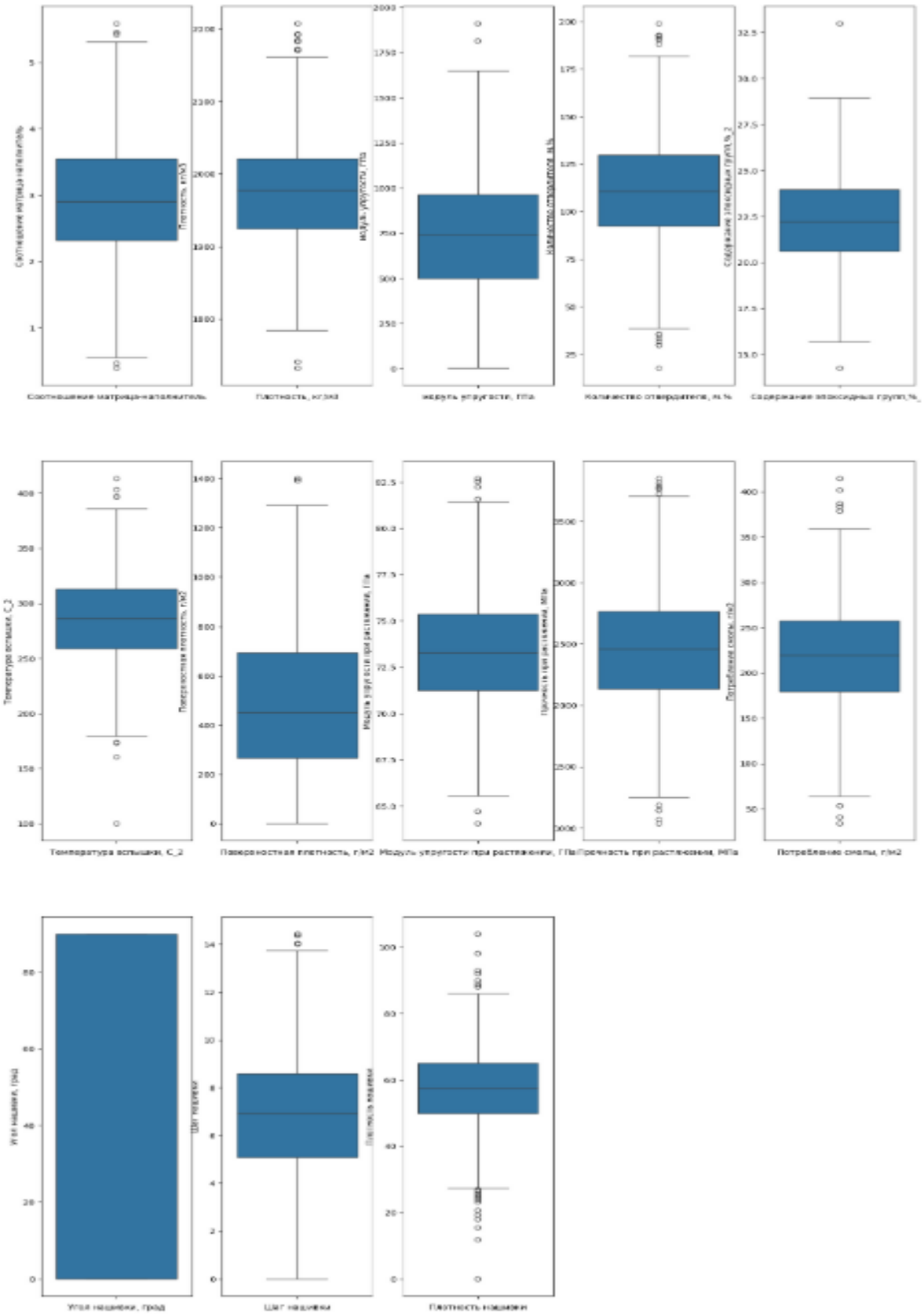

Рисунок 3 - Диаграммы «ящик с усами»

В задании целевыми переменными указаны:

1) модуль упругости при растяжении, Гпа;
2) прочность при растяжении, МПа;
3) соотношение матрица-наполнитель.

1.2 Описание используемых методов

Данная задача в рамках классификации категорий машинного обучения относится к машинному обучению с учителем и традиционно это задача регрессии. Цель любого алгоритма обучения с учителем — определить функцию потерь и минимизировать её, поэтому для наилучшего решения в процессе исследования были применены следующие методы:

- метод опорных векторов;
- случайный лес;
- линейная регрессия;
- градиентный бустинг;
- К-ближайших соседей;
- дерево решений;
- стохастический градиентный спуск;
- Lasso;
- Ridge;
- DummyRegressor.

Предсказание значений вещественной, непрерывной переменной — это задача регрессии. Эта зависимая переменная должна иметь связь с одной или несколькими независимыми переменными, называемых также предикторами или регрессорами. Регрессионный анализ помогает понять, как «типичное» значение зависимой переменной изменяется при изменении независимых переменных.

В настоящее время разработано много методов регрессионного анализа. Например, простая и множественная линейная регрессия. Эти модели являются

параметрическими в том смысле, что функция регрессии определяется конечным числом неизвестных параметров, которые оцениваются на основе данных.

1.2.1 DummyRegressor

DummyRegressor - это простая базовая модель («заглушка») для задач регрессии, которая не учится на данных в обычном смысле, а вместо этого делает предсказания на основе заранее заданной стратегии. Она используется в основном для:

1) сравнения с более сложными моделями (чтобы понять, действительно ли ваша модель лучше, чем случайное или тривиальное предсказание);
2) проверки корректности реализации более сложных алгоритмов;
3) установления базового уровня (baseline) производительности.

В библиотеке scikit-learn (sklearn.dummy.DummyRegressor) со стратегией «mean» всегда предсказывает среднее значение целевой переменной из обучающей выборки. Если другая «умная» модель показывает результаты, близкие к DummyRegressor, это сигнал, что:

1) модель не учится (возможно, ошибка в коде или признаках);
2) данные не содержат полезной информации для предсказания;
3) необходимо пересмотреть подход (инженерия признаков, выбор модели и т.д.).

Таким образом, DummyRegressor — это важный инструмент для валидации и интерпретации результатов в машинном обучении.

1.2.2 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).

$$Y = b0 + b1 * x1 + b2 * x2 + \cdots + bn * xn, \quad (2)$$

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 рекомендуется начать подбор подходящей модели.

Линейная регрессия реализована в библиотеке scikit-learn в sklearn.linear_model.LinearRegression на языке python.

1.2.3 Лассо (LASSO) и гребневая (Ridge) регрессия

Метод регрессии лассо (LASSO, Least Absolute Shrinkage and Selection Operator) — это вариация линейной регрессии, специально адаптированная для данных, которые имеют сильную корреляцию признаков друг с другом. Это простой метод, позволяющий уменьшить сложность модели и предотвратить переопределение, которое может возникнуть в результате простой линейной регрессии.

LASSO использует сжатие коэффициентов (shrinkage) и этим пытается уменьшить сложность данных, искривляя пространство, на котором они лежат. В

этом процессе лассо автоматически помогает устранить или исказить сильно коррелированные и избыточные функции в методе с низкой дисперсией.

Регрессия лассо использует регуляризацию L1, то есть взвешивает ошибки по их абсолютному значению. Данный метод вводит дополнительное слагаемое регуляризации в оптимизацию модели. Это даёт более устойчивое решение. В регрессии лассо добавляется условие смещения в функцию оптимизации для того, чтобы уменьшить коллинеарность и, следовательно, дисперсию модели. Но вместо квадратичного смещения, используется смещение абсолютного значения. Лассо регрессия хорошо прогнозирует модели временных рядов на основе регрессии, таким как авторегрессии.

Достоинства метода: легко полностью избавляется от шумов в данных; быстро работает; не очень энергоёмко; способно полностью убрать признак из датасета; доступно обнуляет значения коэффициентов.

Недостатки метода: выбор модели не помогает и обычно вредит; часто страдает качество прогнозирования; выдаёт ложное срабатывание результата; случайным образом выбирает одну из коллинеарных переменных; не оценивает правильность формы взаимосвязи между независимой и зависимой переменными; не всегда лучше, чем пошаговая регрессия.

Гребневая регрессия или ридж-регрессия — так же вариация линейной регрессии, очень похожая на регрессию LASSO. Она так же применяет сжатие и хорошо работает для данных, которые демонстрируют сильную мультиколлинеарность.

Самое большое различие между ними в том, что гребневая регрессия использует регуляризацию L2, которая взвешивает ошибки по их квадрату, чтобы сильнее наказывать за более значительные ошибки.

Регуляризация позволяет интерпретировать модели. Если коэффициент стал 0 (для Lasso) или близким к 0 (для Ridge), значит данный входной признак не является значимым.

Эти методы реализованы в библиотеке scikit-learn в sklearn.linear_model.Lasso и sklearn.linear_model.Ridge.

1.2.4 Метод опорных векторов для регрессии

Метод опорных векторов (support vector machine, SVM) — один из наиболее популярных методов машинного обучения. Этот бинарный линейный классификатор был выбран, потому что он хорошо работает на небольших датасетах. Он создает гиперплоскость или набор гиперплоскостей в многомерном пространстве, которые могут быть использованы для решения задач классификации и регрессии. Это контролируемое обучение моделей с учителем с использованием схожих алгоритмов для анализа данных и распознавания шаблонов. 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:

1) линейная;
2) полиномиальная;
3) гауссовская (rbf).

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

Достоинства метода: для классификации достаточно небольшого набора данных. При правильной работе модели, построенной на тестовом множестве, вполне возможно применение данного метода на реальных данных. Эффективен при большом количестве гиперпараметров. Способен обрабатывать случаи, когда гиперпараметров больше, чем количество наблюдений. Существует возможность гибко настраивать разделяющую функцию. Алгоритм максимизирует

разделяющую полосу, которая, как подушка безопасности, позволяет уменьшить количество ошибок классификации.

Недостатки метода: неустойчивость к шуму, поэтому в работе была проведена тщательнейшая работа с выбросами, иначе в обучающих данных шумы становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости; для больших наборов данных требуется долгое время обучения; достаточно сложно подбирать полезные преобразования данных; параметры модели сложно интерпретировать, поэтому были рассмотрены и другие методы.

Вариация метода для регрессии называется SVR (Support Vector Regression). В библиотеке scikit-learn реализацию SVR можно найти в sklearn.svm.SVR.

1.2.5 Метод k-ближайших соседей

Еще один метод классификации, который адаптирован для регрессии - метод k-ближайших соседей (k Nearest Neighbors), ищет ближайшие объекты с известными значения целевой переменной и основывается на хранении данных в памяти для сравнения с новыми элементами. Алгоритм находит расстояния между запросом и всеми примерами в данных, выбирая определенное количество примеров (k), наиболее близких к запросу, затем голосует за наиболее часто встречающуюся метку (в случае задачи классификации) или усредняет метки (в случае задачи регрессии). На интуитивном уровне суть метода проста: посмотри на соседей вокруг, какие из них преобладают, таковым 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, эвклидово расстояние для количественных признаков или расстояние Хэмминга для категориальных.

Достоинства метода: прост в реализации и понимании полученных результатов; имеет низкую чувствительность к выбросам; не требует построения

модели; допускает настройку нескольких параметров; позволяет делать дополнительные допущения; универсален; находит лучшее решение из возможных; решает задачи небольшой размерности.

Недостатки метода: замедляется с ростом объёма данных; не создаёт правил; не обобщает предыдущий опыт; основывается на всем массиве доступных исторических данных; невозможно сказать, на каком основании строятся ответы; сложно выбрать близость метрики; имеет высокую зависимость результатов классификации от выбранной метрики; полностью перебирает всю обучающую выборку при распознавании; имеет вычислительную трудоёмкость.

Этот метод — пример непараметрической регрессии, реализован в sklearn.neighbors.KneighborsRegressor.

1.2.6 Деревья решений

Деревья решений (Decision Trees) - еще один непараметрический метод, применяемый и для классификации, и для регрессии. Деревья решений используются в самых разных областях человеческой деятельности и представляют собой иерархические древовидные структуры, состоящие из правил вида «Если ..., то ...».

Решающие правила автоматически генерируются в процессе обучения на обучающем множестве путем обобщения обучающих примеров. Поэтому их называют индуктивными правилами, а сам процесс обучения — индукцией деревьев решений.

Дерево состоит из элементов двух типов: узлов (node) и листьев (leaf). В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу. В результате проверки множество примеров, попавших в узел, разбивается на два подмножества: удовлетворяющие правилу и не удовлетворяющие ему. Затем к каждому подмножеству вновь применяется правило и процедура рекурсивно повторяется пока не будет достигнуто некоторое условие остановки алгоритма. В последнем узле проверка и разбиение не производится, и он объявляется листом.

В листе содержится не правило, а подмножество объектов, удовлетворяющих всем правилам ветви, которая заканчивается данным листом. Для классификации — это класс, ассоциируемый с узлом, а для регрессии — соответствующий листу интервал целевой переменной.

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут, по которому это будет сделано. Общее правило для классификации можно сформулировать так: выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, а количество объектов из других классов в каждом из этих множеств было как можно меньше. Для этого были выбраны различные критерии, например, теоретико-информационный и статистический.

Для регрессии критерием является дисперсия вокруг среднего. Минимизируя дисперсию вокруг среднего, мы ищем признаки, разбивающие выборку таким образом, что значения целевого признака в каждом листе примерно равны.

Достоинства метода в том, что деревья решений помогают визуализировать процесс принятия решения и сделать правильный выбор в ситуациях, когда результаты одного решения влияют на результаты следующих решений; создаются по понятным правилам; просты в применении и интерпретации; заполняют пропуски в данных наиболее вероятным решением; работают с разными переменными, не требовательны к подготовке данных; выделяют наиболее важные поля для прогнозирования; могут использоваться для извлечения правил на естественном языке; обладают высокой точностью работы.

Недостаток деревьев решений: ошибаются при классификации с большим количеством классов и небольшой обучающей выборкой; имеют нестабильный процесс (изменение в одном узле может привести к построению совсем другого дерева); имеет затратные вычисления; необходимо обращать внимание на размер; ограниченное число вариантов решения проблемы; склонность переобучаться. Переобучение в случае дерева решений – это точное распознавание примеров, участвующих в обучении, и полная несостоятельность на новых данных.

В худшем случае, дерево будет большой глубины и сложной структуры, а в каждом листе будет только один объект. Для решения этой проблемы используют разные критерии остановки алгоритма.

Деревья решений реализованы в sklearn.tree.DecisionTreeRegressor.

1.2.7 Случайный лес

Случайный лес (RandomForest) — представитель ансамблевых методов, это множество решающих деревьев. 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 случайным образом.

Преимущества случайного леса:

1) высокая точность предсказания;
2) внутренняя оценка обобщающей способности модели;
3) редко переобучается;
4) практически не чувствителен к выбросам в данных;
5) одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки, данные с большим числом признаков;
6) высокая параллелизуемость и масштабируемость.

Из недостатков можно отметить, что его построение занимает больше времени, не обладает возможностью экстраполяции; сложно интерпретируемый; может недообучаться; трудоёмко прогнозируемый; иногда работает хуже, чем линейные методы.

Метод реализован в sklearn.ensemble.RandomForestRegressor.

1.2.8 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.).

Достоинства метода: новые алгоритмы учатся на ошибках предыдущих; требуется меньше итераций, чтобы приблизиться к фактическим прогнозам; наблюдения выбираются на основе ошибки; прост в настройке темпа обучения и применения; легко интерпретируем.

Из недостатков алгоритма можно отметить необходимость тщательно выбирать критерии остановки, иначе это может привести к переобучению; затраты времени на вычисления; наблюдения с наибольшей ошибкой появляются чаще; слабее и менее гибок, чем нейронные сети; необходимость грамотного подбора гиперпараметров.

В данной работе использована реализация градиентного бустинга из библиотеки scikit-learn — sklearn.ensemble.GradientBoostingRegressor. В то время как существуют и другие реализации, некоторые из которых более мощные, например, XDGBoost.

1.2.9 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 единице 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, сигмоида.

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

1) входной слой — его размер соответствует входным параметрам;
2) скрытые слои — их количество и размерность определяем специалист;
3) 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. В настоящий момент это самый мощный, гибкий и широко применяемый инструмент в машинном обучении.

Области применения нейронных сетей:

1) большие табличные датасеты (например, в рекомендательных системах);
2) когда есть ресурсы на тонкую настройку и инфраструктуру (GPU, фреймворки: PyTorch, TensorFlow);
3) задачи, где важна максимальная точность, а не интерпретируемость.

Среди их достоинств выделяются следующие:

1) высокая гибкость: могут аппроксимировать любые непрерывные функции (теорема универсальной аппроксимации);
2) автоматическое извлечение нелинейных зависимостей и взаимодействий признаков;
3) хорошо масштабируются на большие объёмы данных;
4) поддерживают многозадачное обучение, сложные архитектуры, регуляризацию (dropout, L1/L2);
5) совместимы с эмбеддингами для категориальных признаков.

В то же время существуют недостатки:

1) требуют большого объёма данных для эффективного обучения;

2) высокая вычислительная сложность (обучение медленное без GPU);
3) чувствительны к масштабу признаков;
4) плохая интерпретируемость («чёрный ящик»);
5) сложная настройка: архитектура, функции активации, оптимизатор, learning rate и т.д.;
6) склонны к переобучению на малых данных без регуляризации.

Таким образом, на основе описанных выше основных методов машинного обучения сформирована итоговая сравнительная таблица (таблицы 2 и 3) с указанием среди прочих критериев оценки априорных предпосылок к работоспособности каждого метода.

Таблица 2 — Сравнительная таблица методов машинного обучения (1 часть)

Метод	Интерпретируемость	Скорость обучения	Скорость предсказания	Устойчивость к шуму
DummyRegressor	Нет (тривиален)	Мгновенно	Мгновенно	—
Линейная регрессия	Очень высокая	Очень быстро	Очень быстро	Низкая
Ridge	Высокая	Быстро	Быстро	Средняя
LASSO	Высокая (разреж.)	Быстро	Быстро	Средняя
Дерево решений	Очень высокая	Быстро	Очень быстро	Низкая
Случайный лес	Низкая	Умеренно	Быстро	Высокая
Градиентный бустинг	Низкая	Медленно	Быстро	Высокая
SVM/SVR	Очень низкая	Медленно ($O(n^2–n^3)$)	Умеренно	Низкая
k-NN	Низкая	Нет обучения	Очень медленно ($O(n)$)	Низкая
Нейронные сети (MLP)	Очень низкая	Медленно (особенно без GPU)	Быстро (после обучения)	Средняя (с регуляризацией)

Таблица 3 — Сравнительная таблица методов машинного обучения (2 часть)

Метод	Требует масштабирования	Работает с нелинейностями	Отбор признаков	Априорные предпосылки
DummyRegressor	Нет	Нет	Нет	Нет
Линейная регрессия	Нет (но желательно)	Нет	Нет	Линейность, гомоскедастичность, независимость ошибок
Ridge	Да	Нет	Нет	Линейность, мультиколлинеарность допустима
LASSO	Да	Нет	Да	Линейность, разреженность истинной модели
Дерево решений	Нет	Да	Косвенно	Нет
Случайный лес	Нет	Да	Да (важность)	Нет
Градиентный бустинг	Нет	Да	Да	Нет (но лучше на табличных данных)
SVM/SVR	Да	Да (через ядра)	Нет	Нормализация, умеренный размер выборки
k-NN	Да	Да (локально)	Нет	Низкая размерность, нормализация, локальная гладкость
Нейронные сети (MLP)	Да	Да (глубоко и гибко)	Нет (но можно через attention/важность)	Масштабирование, большой объём данных, числовые признаки

Таблица подсказывает нам, что начать анализ надо с DummyRegressor, чтобы установить нижнюю границу, затем проверить линейную модель + LASSO/Ridge, что будет быстро и интерпретируемо. После чего запустим Random Forest — надёжный baseline для нелинейных задач. В связи с тем, что датасет менее 1000 объектов, то применим SVM (с осторожностью) и k-NN (если мало признаков). Настроим Gradient Boosting - часто лучший выбор для табличных данных. В конце работы пробуем нейронные сети, хотя они не являются «серебряной пулей» для табличных данных: на небольших и средних датасетах градиентный бустинг часто превосходит нейросети по точности и стабильности. Кроме того, они требуют больше инженерии: нормализация, обработка пропусков, подбор архитектуры. А выигрывают они при очень больших объёмах данных, где их гибкость раскрывается полностью, а текущий датасет небольшой.

1.3 Разведочный анализ данных

Цель разведочного анализа данных — выявить закономерности в данных. Для корректной работы большинства моделей желательна сильная зависимость выходных переменных от входных и отсутствие зависимости между входными переменными. Вначале посчитаем среднее и медианное значение признаков, что приведено на рисунках 4 и 5 соответственно. Пропуски в заполнении данных отсутствуют.

```
df.mean()
```

Соотношение матрица-наполнитель	2.930366
Плотность, кг/м3	1975.734888
модуль упругости, ГПа	739.923233
Количество отвердителя, м.%	110.570769
Содержание эпоксидных групп,%_2	22.244390
Температура вспышки, С_2	285.882151
Поверхностная плотность, г/м2	482.731833
Модуль упругости при растяжении, ГПа	73.328571
Прочность при растяжении, МПа	2466.922843
Потребление смолы, г/м2	218.423144
Угол нашивки, град	44.252199
Шаг нашивки	6.899222
Плотность нашивки	57.153929
dtype: float64	

Рисунок 4 – Среднее значение признаков

```
df.median()
```

Соотношение матрица-наполнитель	2.906878
Плотность, кг/м3	1977.621657
модуль упругости, ГПа	739.664328
Количество отвердителя, м.%	110.564840
Содержание эпоксидных групп,%_2	22.230744
Температура вспышки, С_2	285.896812
Поверхностная плотность, г/м2	451.864365
Модуль упругости при растяжении, ГПа	73.268805
Прочность при растяжении, МПа	2459.524526
Потребление смолы, г/м2	219.198882
Угол нашивки, град	0.000000
Шаг нашивки	6.916144
Плотность нашивки	57.341920
dtype: float64	

Рисунок 5 – Медианное значение признаков

Попарные графики рассеяния точек приведены на рисунке 6.

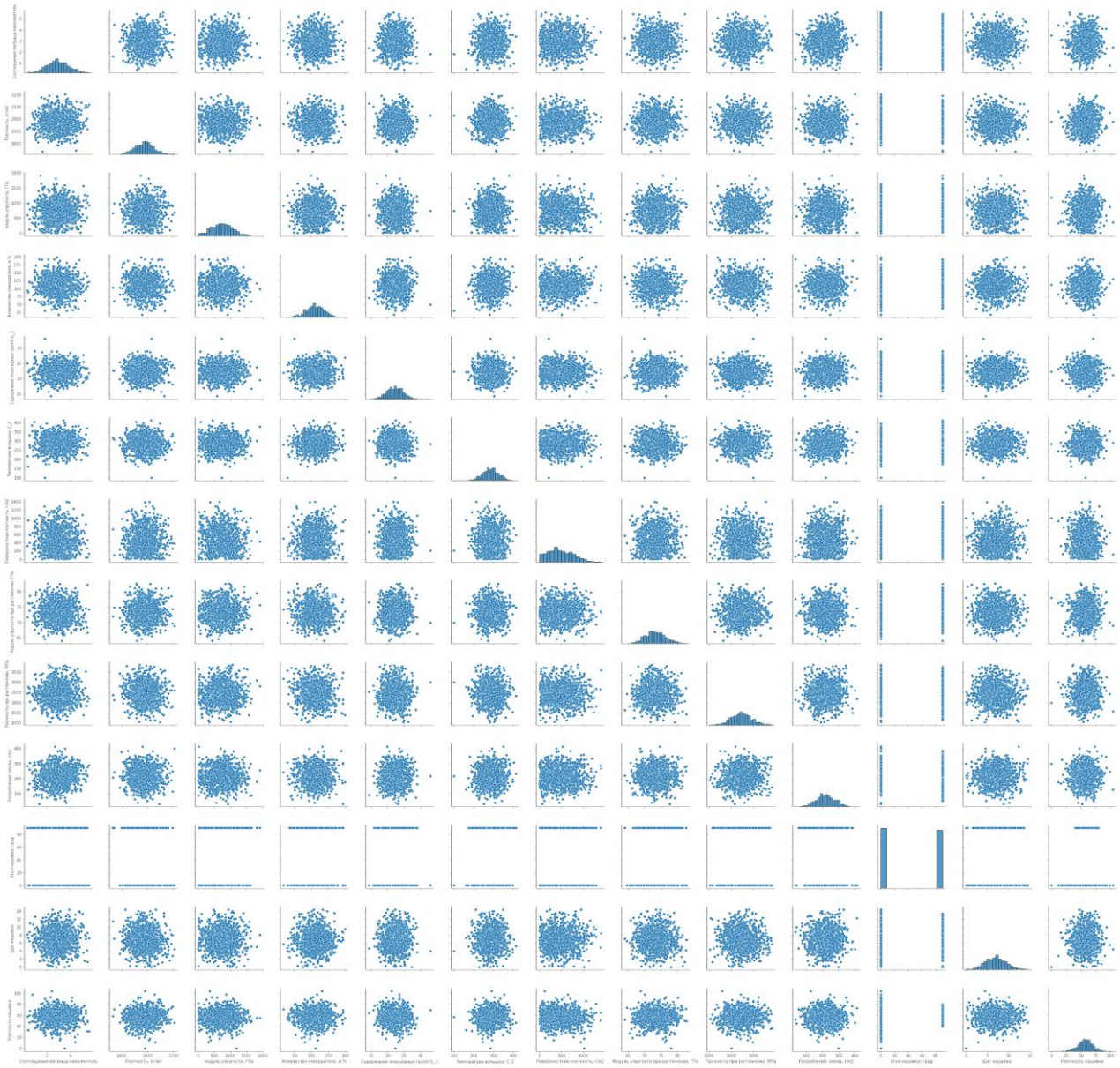

Рисунок 6 — Попарные графики рассеяния точек

По тепловой матрице корреляции, показанной на рисунке 7, видно, что все коэффициенты корреляции близки к нулю, что означает отсутствие линейной зависимости между признаками. Корреляционная матрица не выявила каких-либо зависимостей между признаками, если она имеется, то она очень низкая. Самая высокая зависимость между углом нашивки и плотностью нашивки (0,11). В результате проведенного разведочного анализы мы можем сделать следующие выводы. У почти у всех признаков имеется нормальное распределение и по имеющейся информации, данные являются предварительно обработанными.

```
corrmat = df.corr()

f, ax = plt.subplots(figsize =(15, 12))
sns.heatmap(corrmat, ax = ax, cmap ="YlGnBu", linewidths = 0.1, annot = True)
```

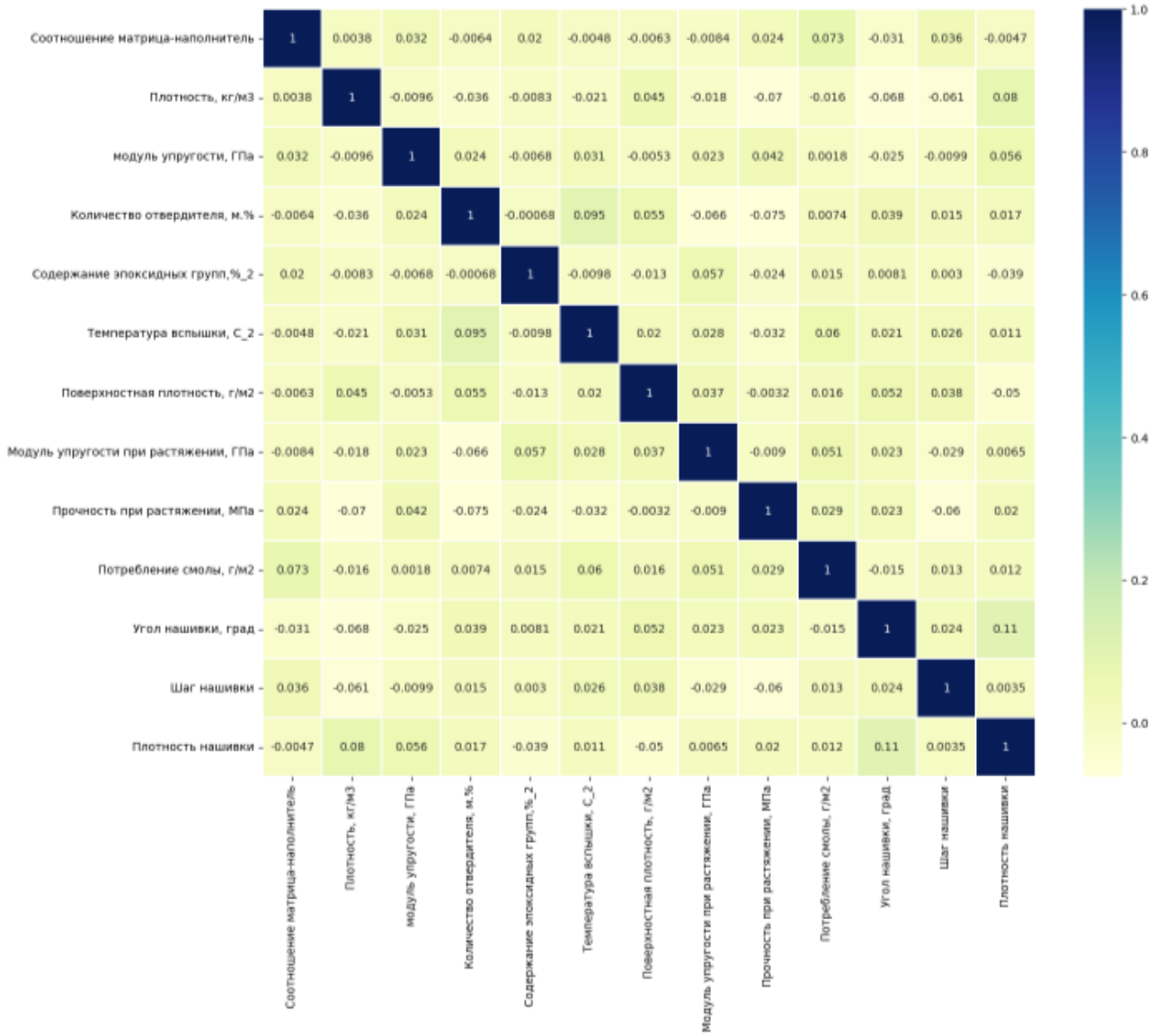

Рисунок 7 — Тепловая карта матрицы корреляции

Применены основные методы выявления выбросов для признаков с нормальным распределением:

1) метод трёх сигм (первоначально выявлено 24 выброса);

2) метод межквартильных расстояний (выявлено 93 выброса).

Пример выбросов показателя плотности на гистограмме распределения и диаграмме «ящик с усами» приведен на рисунке 8.

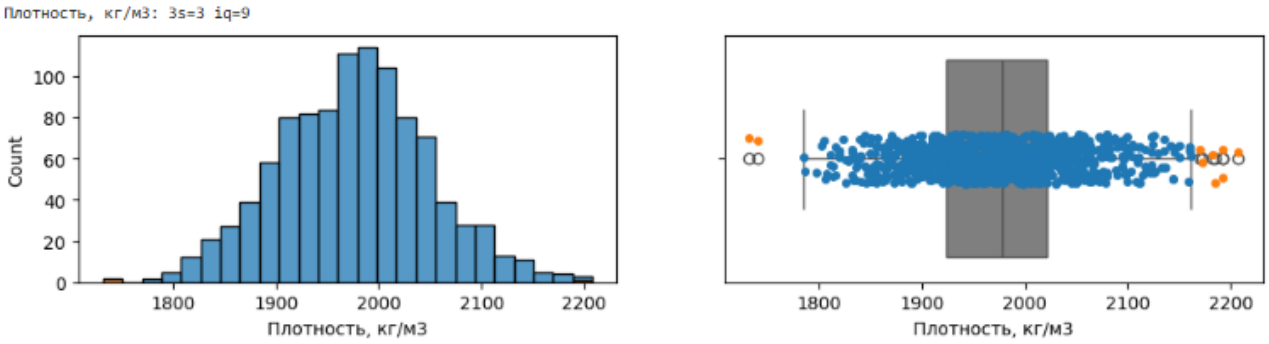

Рисунок 8 — Пример выбросов

Для более точного построения модели было решено максимально очистить датасет от выбросов, применяя метод межквартильных расстояний. Значения, определенные как выбросы, удаляем. После трех последовательных итераций метода межквартильных расстояний в датасете осталось 922 строки и 13 признаков-переменных.

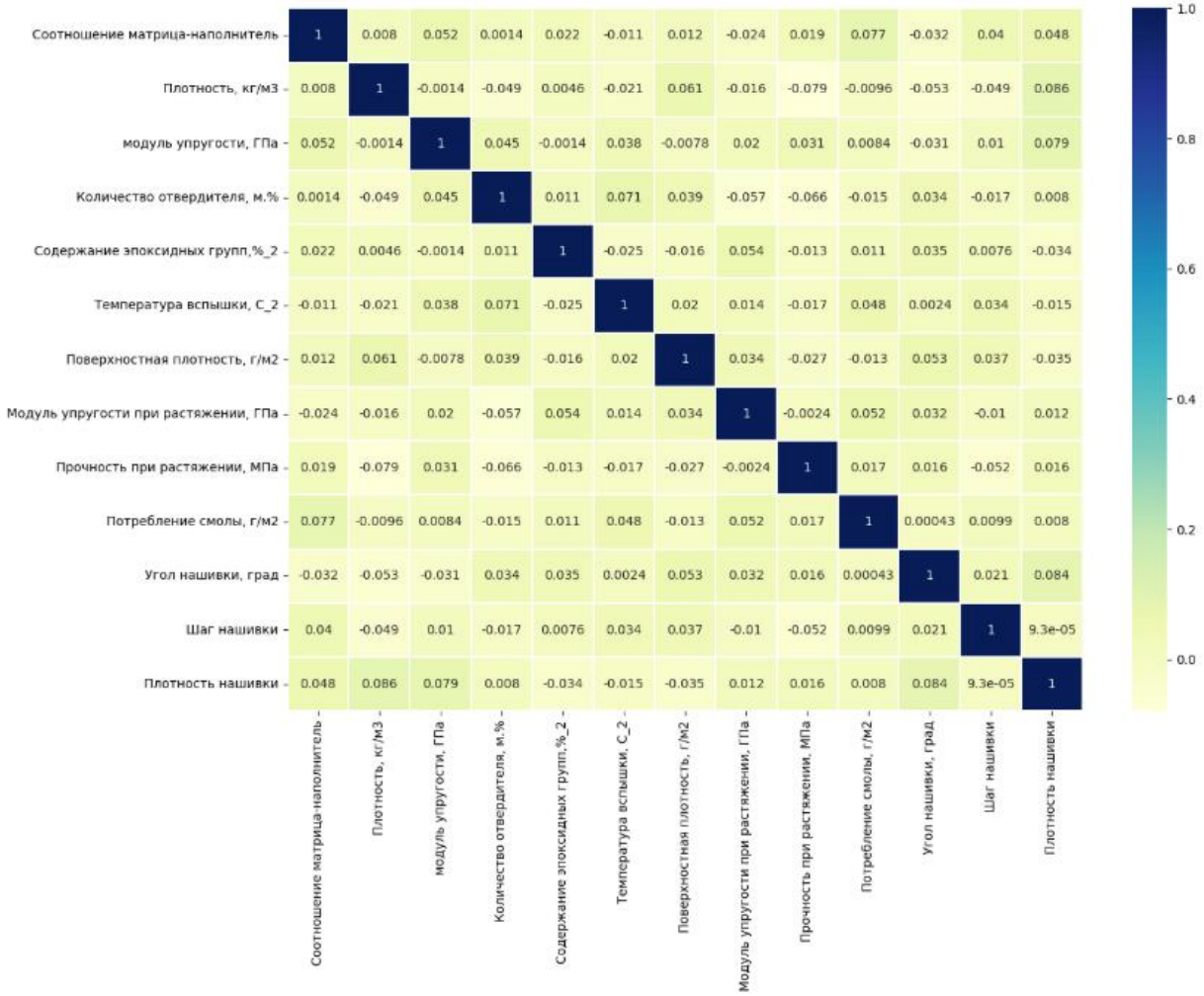

Рисунок 9 — Тепловая карта матрицы корреляции очищенных данных

На тепловой карте корреляционной матрицы очищенных данных, показанной на рисунке 9, видно, что корреляция изменилась незначительно. Существенных изменений нет, корреляция между признаками по-прежнему фактически отсутствует. Коэффициенты корреляции, близкие к нулю, показывают отсутствие линейной зависимости между признаками. Можно предположить, что применение линейных моделей регрессии не даст приемлемого результата.

1.3.1 Выбор признаков

Явных зависимостей между признаками не было обнаружено статистическими методами. В то же время исходя из проанализированной информации, выявлено, что признаки делятся на:

1) свойства матрицы;
2) свойства наполнителя;
3) свойства смеси и производственного процесса;
4) свойства готового композита.

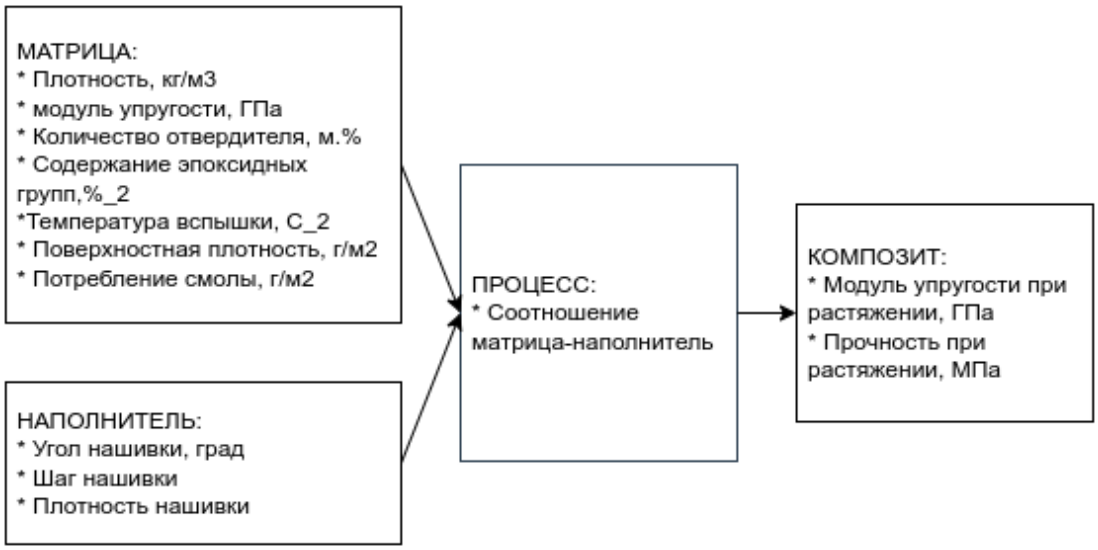

Рисунок 10 — Группы признаков с точки зрения предметной области

Распределение признаков в общем процессе по группам представлено на рисунке 10. Это позволило сделать вывод, что целевые признаки имеют зависимости вида (4), (5), (6).

$$\text{модуль упр} - \text{тип при растяжении}(\text{композит}) = f(\text{матрица}, \text{наполнитель}, \text{процесс}) \quad (4)$$

$$\text{прочность при растяжении}(\text{композит}) = f(\text{матрица}, \text{наполнитель}, \text{процесс}) \quad (5)$$

$$\text{соотн} - \text{е матрица} - \text{наполнитель}(\text{процесс}) = f(\text{матрица}, \text{наполнитель}, \text{композит}) \quad (6)$$

Для каждого из целевых признаков необходимо построить отдельную модель и решить три отдельные задачи.

1.3.2 Ход решения задачи

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

1) разделить данные на тренировочную и тестовую выборки. В задании указано, что на тестирование оставить 30% данных;
2) выполнить препроцессинг, то есть подготовку исходных данных;
3) выбрать базовую модель для определения нижней границы качества предсказания. Использую базовую модель, возвращающую среднее значение целевого признака. Лучшая модель по своим характеристикам должна быть лучше базовой;
4) взять несколько моделей с гиперпараметрами по умолчанию, и используя перекрестную проверку, посмотреть их метрики на тренировочной выборке;
5) подобрать для этих моделей гиперпараметры с помощью с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10;
6) сравнить метрики моделей после подбора гиперпараметров и выбрать лучшую;

7) получить предсказания лучшей и базовой моделей на тестовой выборке, сделать выводы;
8) сравнить качество работы лучшей модели на тренировочной и тестовой выборке.

1.3.3 Препроцессинг

Цель препроцессинга, или предварительной обработки данных — обеспечить корректную работу моделей. Его необходимо выполнять после разделения на тренировочную и тестовую выборку, как будто мы не знаем параметров тестовой выборки (минимум, максимум, матожидание, стандартное отклонение).

Препроцессинг для категориальных и количественных признаков выполняется по-разному. Категориальный признак один – «Угол нашивки, град». Он принимает значения 0 и 90. Модели отработают лучше, если мы превратим эти значения в 0 и 1 с помощью LabelEncoder или OrdinalEncoder.

Проблема вещественных признаков, которых большинство, заключается в том, что их значения лежат в разных диапазонах, в разных масштабах, что видно на рисунке 1. Необходимо провести одно из двух возможных преобразований:

1) нормализацию — приведение в диапазон от 0 до 1 с помощью MinMaxScaler;
2) стандартизацию — приведение к матожиданию 0, стандартному отклонению 1 с помощью StandartScaler.

В данной работе была выбрана стандартизация с помощью StandardScaler.

Для повторного использования препроцессинга в приложении для введенных данных реализована предварительная обработка в виде функции ColumnTransformer, что позволило сохранить и загрузить этот объект аналогично объекту модели. Выходные переменные не изменялись.

1.3.4 Перекрестная проверка

Для обеспечения статистической устойчивости метрик модели использована перекрестная проверка или кросс-валидация, при которой выборка разбивается необходимое количество раз на тестовую и валидационную. Модель обучается на тестовой выборке, затем выполняется расчет метрик качества на валидационной. В качестве результата получены средние метрики качества для всех валидационных выборок. Перекрестную проверку реализует функция cross_validate из библиотеки scikit-learn.

1.3.5 Поиск гиперпараметров по сетке

Поиск гиперпараметров по сетке реализует класс GridSearchCV из библиотеки scikit-learn. Он получает модель и набор гиперпараметров, поочередно передает их в модель, выполняет обучение и определяет лучшие комбинации гиперпараметры. Перекрестная проверка уже встроена в этот класс.

1.3.6 Метрики качества моделей

В качестве метрик качества, применяемых для регрессии, были использованы:

1) R2 или коэффициент детерминации, измеряющий 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;
2) RMSE (Root Mean Squared Error) или корень из средней квадратичной ошибки 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;
3) MAE (Mean Absolute Error) - средняя абсолютная ошибка принимает значениях в тех же единицах, что и целевая переменная;

4) MAPE (Mean Absolute Percentage Error) или средняя абсолютная процентная ошибка — безразмерный показатель, представляющий собой взвешенную версию MAE;
5) max error или максимальная ошибка данной модели в единицах измерения целевой переменной.

Метрики RMSE, MAE, MAPE, max error необходимо минимизировать.

R2 в норме принимает положительные значения. Эту метрику надо максимизировать. Отрицательные значение коэффициента детерминации означают плохую объясняющую способность модели.

2 Практическая часть

2.1 Разбиение и предобработка данных

2.1.1 Для прогнозирования модуля упругости при растяжении

Признаки датасета были разделены на входные и выходные, а строки - на тренировочное и тестовое множество. Размерности полученных наборов данных показаны на рисунке 11. Описательная статистика входных признаков до и после предобработки показана на рисунке 12. Описательная статистика выходного признака показана на рисунке 13.

```
x1_train: (645, 11) y1_train: (645, 1)
x1_test: (277, 11) y1_test: (277, 1)
```

Рисунок 11 - Размерности тренировочного и тестового множеств после разбиения для первой задачи

```
# Описательная статистика входных данных до предобработки
show_statistics(x1_train_raw)
```

	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	0.547391	1784.482245	2.436909	40.304806	15.695894	188.918674	1.894093	72.530873	0.000000	0.037639	28.661632
max	5.314144	2161.565216	1628.000000	179.645962	28.955094	385.697799	1288.691844	359.052220	90.000000	13.571921	86.012427
mean	2.925725	1973.514328	735.549446	111.418752	22.277356	286.441169	483.129553	217.871279	44.930233	6.948415	57.848569
std	0.906983	71.158440	324.420003	26.337565	2.378254	37.919237	282.682289	56.915998	45.034870	2.500522	11.048670

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

	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	-2.624284	-2.658557	-2.261517	-2.702191	-2.769498	-2.573843	-1.703711	-2.555577	-2.765878	-2.643720	0.000000
max	2.635411	2.644758	2.753046	2.592501	2.810012	2.619611	2.851921	2.482439	2.650905	2.551051	1.000000
mean	0.000000	-0.000000	-0.000000	0.000000	-0.000000	0.000000	-0.000000	-0.000000	0.000000	-0.000000	0.499225
std	1.000776	1.000776	1.000776	1.000776	1.000776	1.000776	1.000776	1.000776	1.000776	1.000776	0.500387

Рисунок 12 - Описательная статистика входных признаков до и после предобработки для первой задачи

	Модуль упругости при растяжении, ГПа
min	65.979990
max	81.203147
mean	73.342384
std	3.027444

Рисунок 13 - Описательная статистика выходного признака для первой 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 14. Описательная статистика входных признаков до и после предобработки показана на рисунке 15. Описательная статистика выходного признака показана на рисунке 16.

```
x2_train: (645, 11) y2_train: (645, 1)
x2_test: (277, 11) y2_test: (277, 1)
```

Рисунок 14 - Размерности тренировочного и тестового множеств после разбиения для второй задачи

```
# Описательная статистика входных данных до предобработки
show_statistics(x2_train_raw)
```

	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	0.547391	1784.482245	2.436909	40.304806	15.695894	188.918674	1.894093	72.530873	0.000000	0.037639	28.661632
max	5.314144	2161.565216	1628.000000	179.645962	28.955094	385.697799	1288.691844	359.052220	90.000000	13.571921	86.012427
mean	2.925725	1973.514328	735.549446	111.418752	22.277356	286.441169	483.129553	217.871279	44.930233	6.948415	57.848569
std	0.906983	71.158440	324.420003	26.337565	2.378254	37.919237	282.682289	56.915998	45.034870	2.500522	11.048670

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

	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	-2.748293	-2.700419	-2.194749	-2.534757	-2.613405	-2.263523	-1.740581	-2.531421	-2.693620	-2.506913	0.000000
max	2.742740	2.629675	2.659192	2.498459	2.864358	2.344673	2.936298	2.438852	2.623847	2.584929	1.000000
mean	-0.008585	-0.028437	-0.005671	0.033988	0.105589	0.020270	0.008474	-0.010207	0.021546	0.084425	0.499225
std	1.044793	1.005829	0.968720	0.951353	0.982526	0.887997	1.027411	0.987319	0.982427	0.980947	0.500387

Рисунок 15 - Описательная статистика входных признаков до и после предобработки для второй задачи

	Прочность при растяжении, МПа
min	1250.392802
max	3636.892992
mean	2466.696221
std	459.451353

Рисунок 16 - Описательная статистика выходного признака для второй 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 17. Описательная статистика входных признаков до и после предобработки показана на рисунке 18. Описательная статистика выходного признака показана на рисунке 19.

```
x3_train: (645, 12) y3_train: (645, 1)
x3_test: (277, 12) y3_test: (277, 1)
```

Рисунок 17 - Размерности тренировочного и тестового множеств после разбиения для третьей задачи

```
# Описательная статистика входных данных до предобработки
show_statistics(x3_train_raw)
```

	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
min	1784.482245	2.436909	40.304806	15.695894	188.918674	1.894093	65.979990	1250.392802	72.530873	0.000000	0.037639	28.661632
max	2161.565216	1628.000000	179.645962	28.955094	385.697799	1288.691844	81.203147	3636.892992	359.052220	90.000000	13.571921	86.012427
mean	1973.514328	735.549446	111.418752	22.277356	286.441169	483.129553	73.342384	2466.696221	217.871279	44.930233	6.948415	57.848569
std	71.158440	324.420003	26.337565	2.378254	37.919237	282.682289	3.027444	459.451353	56.915998	45.034870	2.500522	11.048670

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

	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	Угол нашивки, град
min	-2.700419	-2.194749	-2.534757	-2.613405	-2.263523	-1.740581	-2.394788	-2.729074	-2.531421	-2.693620	-2.506913	0.000000
max	2.629675	2.659192	2.498459	2.864358	2.344673	2.936298	2.645581	2.702995	2.438852	2.623847	2.584929	1.000000
mean	-0.028437	-0.005671	0.033988	0.105589	0.020270	0.008474	0.042892	0.039434	-0.010207	0.021546	0.084425	0.499225
std	1.005829	0.968720	0.951353	0.982526	0.887997	1.027411	1.002383	1.045787	0.987319	0.982427	0.980947	0.500387

Рисунок 18 - Описательная статистика входных признаков до и после предобработки для третьей задачи

	Соотношение матрица-наполнитель
min	0.547391
max	5.314144
mean	2.925725
std	0.906983

Рисунок 19 - Описательная статистика выходного признака для третьей задачи

2.2 Разработка и обучение моделей для прогнозирования модуля упругости при растяжении

Для подбора лучшей модели для этой задачи взяты следующие модели:

1) LinearRegression — линейная регрессия (раздел 1.2.2);
2) Ridge — гребневая регрессия (раздел 1.2.3);
3) Lasso — лассо-регрессия (раздел 1.2.3);
4) SVR — метод опорных векторов (раздел 1.2.4);
5) KneighborsRegressor — метод ближайших соседей (раздел 1.2.5);
6) DecisionTreeRegressor — деревья решений (раздел 1.2.6);
7) RandomForestRegressor — случайный лес (раздел 1.2.7);
8) GradientBoostingRegressor – градиентный бустинг (раздел 1.2.8).

В качестве базовой модели взят DummyRegressor, возвращающий среднее значение целевого признака.

Метрики работы выбранных моделей с гиперпараметрами по умолчанию, полученные с помощью перекрестной проверки на тестовом множестве, приведены на рисунке 20. Ни одна из выбранных мной моделей не оказалась подходящей для наших данных. Коэффициент детерминации R2 близок к нулю для линейных моделей, при этом по методу LASSO его значение совпало со значением, полученным из базовой модели DummyRegressor. Значит, они не лучше базовой модели. И остальные метрики у них примерно совпадают с базовой моделью.

Гораздо хуже линейных моделей с гиперпараметрами по умолчанию отработали деревья решений, метод ближайших соседей и градиентный бустинг.

Случайный лес отработал лучше, чем одно дерево решений, и чуть лучше, чем метод опорных векторов, но хуже, чем все линейные модели.

	R2	RMSE	MAE	MAPE	max_error
DummyRegressor	-0.011479	3.018468	2.434764	0.033263	-7.216742
LinearRegression	-0.022599	3.034619	2.453669	0.033520	-7.222509
Ridge	-0.022517	3.034496	2.453574	0.033519	-7.222363
Lasso	-0.011479	3.018468	2.434764	0.033263	-7.216742
SVR	-0.085891	3.124768	2.524366	0.034452	-7.445850
KNeighborsRegressor	-0.226448	3.311413	2.628943	0.035925	-8.330975
DecisionTreeRegressor	-1.317465	4.505732	3.660577	0.050026	-11.406496
RandomForestRegressor	-0.081441	3.117797	2.525393	0.034500	-7.342254
GradientBoostingRegressor	-0.132211	3.184520	2.554379	0.034910	-7.971823

Рисунок 20 — Результаты моделей с гиперпараметрами по умолчанию

	R2	RMSE	MAE	MAPE	max_error
Ridge(alpha=700, positive=True, solver='lbfgs')	-0.007231	3.011776	2.432033	0.033226	-7.126265
Lasso(alpha=0.15)	-0.009012	3.014177	2.429802	0.033195	-7.183302
SVR(C=0.02)	-0.012616	3.020104	2.437300	0.033289	-7.228111
KNeighborsRegressor(n_neighbors=29)	-0.059927	3.087279	2.493132	0.034113	-7.254115
DecisionTreeRegressor(max_depth=1, max_features=1, random_state=3128, splitter='random')	-0.011451	3.018141	2.426078	0.033145	-7.182651
RandomForestRegressor(bootstrap=False, criterion='absolute_error', max_depth=2, max_features=1, n_estimators=50, random_state=3128)	-0.011578	3.018652	2.438736	0.033310	-7.189810

Рисунок 21 — Результаты моделей после подбора гиперпараметров

После выполнения подбора гиперпарметров по сетке с перекрестной проверкой, получены метрики, приведенные на рисунке 21. Сделан вывод, что, подбирая гиперпараметры, можно значительно улучшить предсказание выбранной модели. Все модели крайне плохо описывают исходные данные - не удалось добиться положительного значения R2. Самая лучшая модель (Ridge) дает коэффициент детерминации близкий к нулю (-0,007), что соответствует базовой модели. Линейные модели совпадают с базовой моделью. Их характеристики улучшились, но не значительно.

Метод опорных векторов в процессе подбора гиперпараметры лучшим ядром выбрал линейное и отработал аналогично линейным моделям.

Метод ближайших соседей увеличением количества соседей радикально улучшил качество работы. Но его лучшие результаты все равно немного, но отстают от линейных моделей.

Деревья решений при кропотливом подборе параметров отстают по результатам от линейных моделей, они не являются объясняющей зависимость моделью.

Собирая деревья в ансамбли, можно улучшать характеристики. Но подбор параметров для леса затруднен тем, что это затратный по времени процесс, в связи с чем, не удалось получить комбинацию параметров для леса, которая была был лучше дерева решений.

Поэтому в качестве лучшей модели была выбрана линейная модель Ridge. На рисунке 22 приведена визуализация работы лучшей модели на тестовом множестве.

Сложно визуализировать регрессию в многомерном пространстве. Но даже на таком графике мы видим, насколько не соответствует лучшая модель исходным данным и насколько она неудачна.

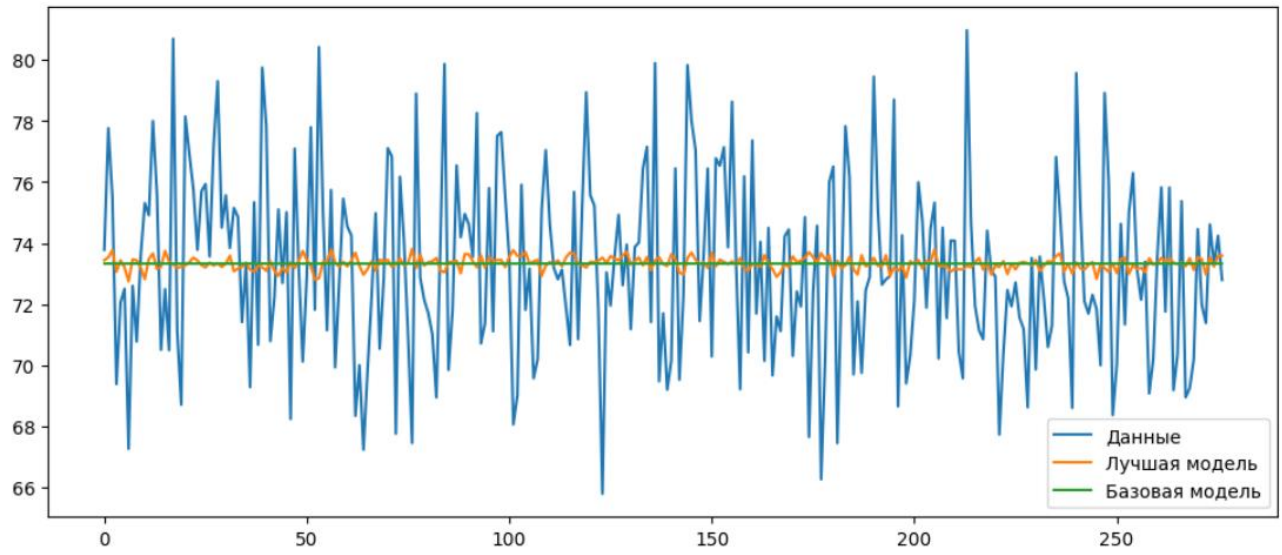

Рисунок 22 — Визуализация работы модели Ridge

Метрики работы лучшей модели на тестовом множестве и сравнении с базовой DummyRegressor отражены на рисунке 23. Они подтверждают: полученная модель хуже базовой. Результат исследования отрицательный, так как не удалось

получить модель, которая смогла бы оказать помощь в принятии решений специалисту предметной области.

	R2	RMSE	MAE	MAPE	max_error
Базовая модель	-0.001840	3.023023	2.480618	0.033942	7.628576
Лучшая модель (Ridge)	-0.009474	3.034520	2.475754	0.033873	7.730087

Рисунок 23 - Метрики работы лучшей модели на тестовом множестве

2.3 Разработка и обучение моделей для прогнозирования прочности при растяжении

Для подбора лучшей модели для прогнозирования прочности при растяжении были определены следующие модели:

1) LinearRegression — линейная регрессия (раздел 1.2.2);
2) Ridge — гребневая регрессия (раздел 1.2.3);
3) Lasso — лассо-регрессия (раздел 1.2.3);
4) SVR — метод опорных векторов (раздел 1.2.4);
5) DecisionTreeRegressor — деревья решений (раздел 1.2.6);
6) GradientBoostingRegressor — случайный лес (раздел 1.2.7).

В качестве базовой модели взят DummyRegressor, возвращающий среднее значение целевого признака.

Метрики работы выбранных моделей с гиперпараметрами по умолчанию, полученные с помощью перекрестной проверки на тестовом множестве, приведены на рисунке 24.

	R2	RMSE	MAE	MAPE	max_error
DummyRegressor	-0.014754	458.517408	366.664655	0.160630	-1095.981614
LinearRegression	-0.017429	459.567129	368.992333	0.161357	-1121.299260
Ridge	-0.017348	459.547973	368.974704	0.161350	-1121.197755
Lasso	-0.014644	458.916723	368.444713	0.161130	-1118.923935
SVR	-0.012796	458.101333	366.555234	0.160547	-1094.668346
DecisionTreeRegressor	-1.101336	648.893181	524.028712	0.223497	-1648.797617
GradientBoostingRegressor	-0.121957	482.055962	389.345452	0.169754	-1194.692003

Рисунок 24 — Результаты моделей с гиперпараметрами по умолчанию

Как видно из рисунка 24, ни одна из выбранных моделей не соответствует данным. R2 близок к нулю для линейных моделей и метода опорных векторов. Значит, они не лучше базовой модели. И остальные метрики у них примерно совпадают с базовой моделью. Метод регрессии опорных векторов с параметрами по умолчанию отработал лучше дерева и чуть лучше базовой модели.

Гораздо хуже линейных моделей с гиперпараметрами по умолчанию отработали деревья решений.

Градиентный бустинг с параметрами по умолчанию отработал лучше дерева, но хуже остальных моделей. Он тоже соответствует базовой модели.

После выполнения подбора гиперпарметров по сетке с перекрестной проверкой, получили метрики, приведенные на рисунке 25.

	R2	RMSE	MAE	MAPE	max_error
Ridge(alpha=990, solver='lsqr')	-0.008290	457.219991	366.133785	0.160339	-1092.435894
Lasso(alpha=20)	-0.005856	456.617587	365.960810	0.160195	-1090.504240
SVR(C=0.02, kernel='linear')	-0.012688	458.075057	366.548778	0.160541	-1094.267773
DecisionTreeRegressor(max_depth=1, max_features=3, random_state=3128)	-0.020410	459.733570	365.998753	0.160581	-1107.058869
GradientBoostingRegressor(loss='absolute_error', max_depth=1, max_features=11, n_estimators=50, random_state=3128)	-0.022204	460.206446	368.817375	0.161594	-1105.850304

Рисунок 25 — Результаты моделей после подбора гиперпараметров

С помощью подбора гиперпараметров не удалось получить модель, превосходящую базовую. Все модели крайне плохо описывают исходные данные. Не удалось добиться коэффициента детерминации, принимающего значение больше нуля. Линейные модели после подбора немного улучшили характеристики. При этом лучшая модель среди остальных получена при использовании метода Lasso.

Метод опорных векторов отработал чуть хуже линейных моделей.

Деревья решений после подбора параметров улучшили неудачный результат с параметрами по умолчанию.

Аналогично хуже результат, чем линейные модели, показал градиентный бустинг. Значения ошибок примерно такие же, как у дерева решений.

В качестве лучшей модели выбираю линейную модель Lasso. На рисунке 26 приведена визуализация работы лучшей модели на тестовом множестве.

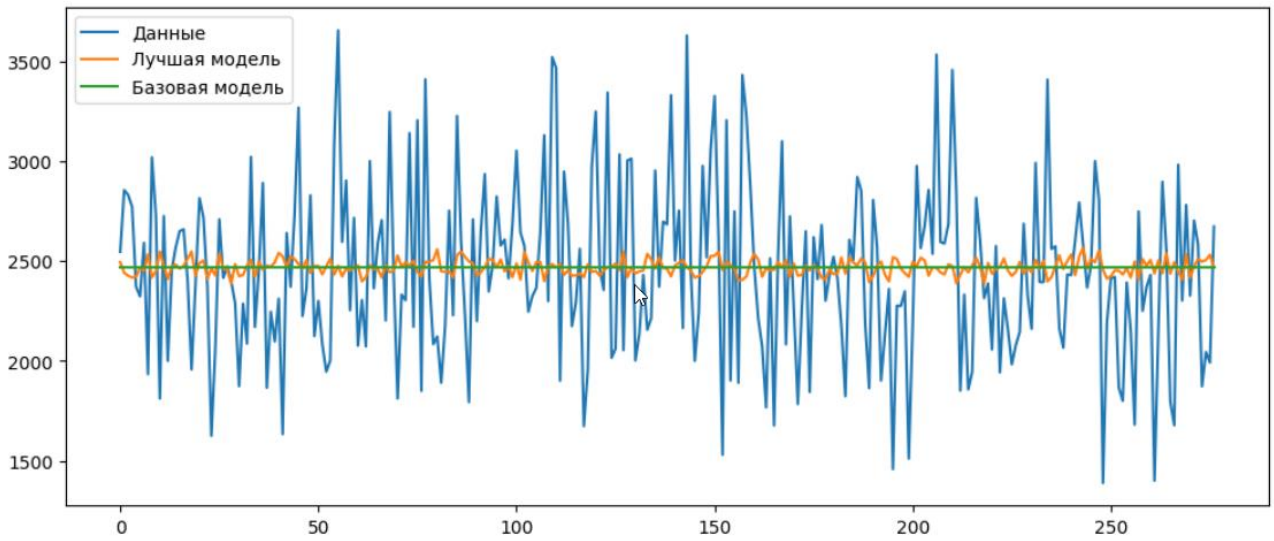

Рисунок 26 — Визуализация работы модели Lasso

Визуализируя результаты линейной модели Lasso с выбранными параметрами, мы видим насколько они плохи и далеки от исходных данных. Результаты выглядят аналогично, как те, которые получены линейным методом Ridge для модуля упругости при растяжении.

Метрики работы лучшей модели Lasso на тестовом множестве и сравнение с базовой отражены на рисунке 27. Линейная модель Lasso показывает результаты хуже базовой, результат исследования отрицательный. Не удалось получить модель, которая могла бы оказать помощь в принятии решений специалисту предметной области.

	R2	RMSE	MAE	MAPE	max_error
Базовая модель	-0.001555	439.676848	350.354301	0.151168	1187.738138
Лучшая модель (Lasso)	-0.011555	441.866376	350.404421	0.151146	1178.211769

Рисунок 27 - Метрики работы лучшей модели на тестовом множестве

2.4 Разработка нейронной сети для прогнозирования соотношения матрица-наполнитель

По заданию для соотношения матрица-наполнитель необходимо построить нейросеть. Но для сравнения нам также понадобится базовая модель DummyRegressor, возвращающая среднее целевого признака.

2.4.1 MLPRegressor из библиотеки scikit-learn

В ходе выполнения работы была построена нейронная сеть с помощью класса MLPRegressor следующей архитектуры:

1) слоев: 8;
2) нейронов на каждом слое: 24;
3) активационная функция: relu;
4) оптимизатор: adam;
5) пропорция разбиения данных на тестовые и валидационные: 30%;
6) ранняя остановка, если метрики на валидационной выборке не улучшаются;
7) количество итераций: 5000.

Нейросеть обучилась за 234 мс и 46 итераций. График обучения приведен на рисунке 28.

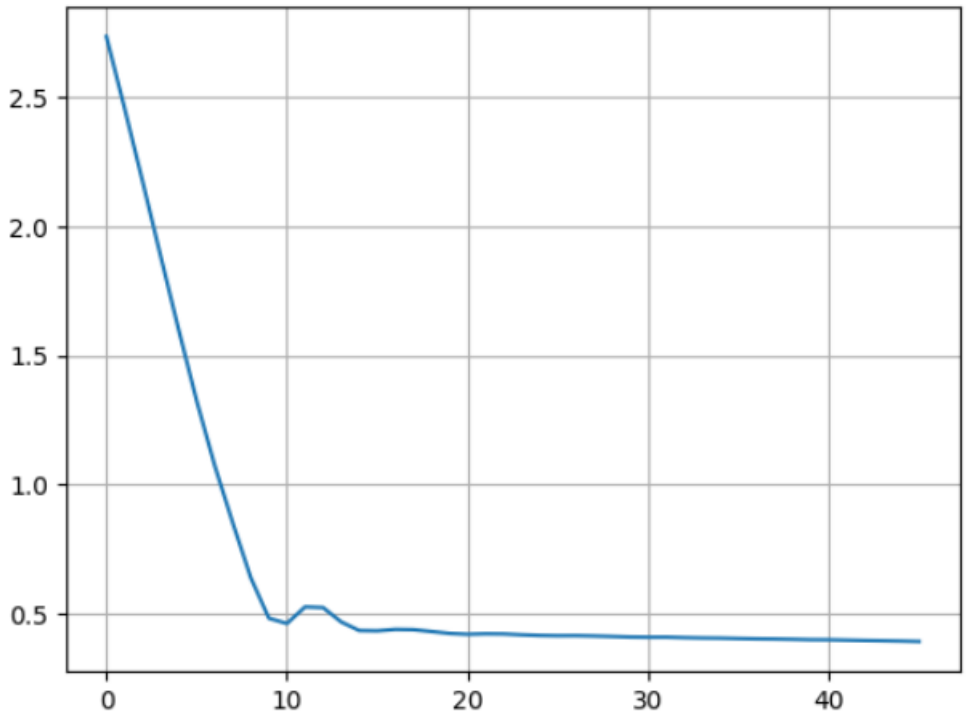

Рисунок 28 — График обучения MLPRegressor

Визуализация результатов, полученных нейросетью, приведены на рисунке 29. Видно, что нейросеть пыталась подстроится под исходные данные, но полностью подстроиться не получилось.

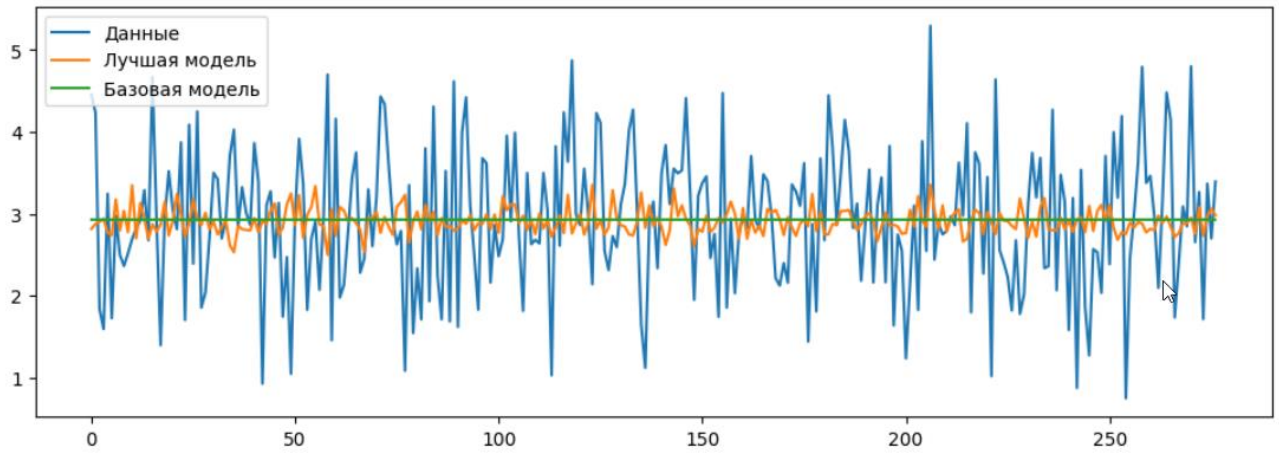

Рисунок 29 — Визуализация работы модели

Метрики работы нейросети MLPRegressor на тестовом множестве и сравнение с базовой моделью отражены на рисунке 30. Несмотря на красивый график на рисунке 29, метрики свидетельствуют об отсутствии результата, который можно внедрить. Ошибка нейросети составляет 30,6%, а значения её ошибок хуже, чем у базовой модели.

	R2	RMSE	MAE	MAPE	max_error
DummyRegressor	-0.000074	0.868130	0.691547	0.296759	2.370117
MLPRegressor	-0.064119	0.895496	0.722489	0.305808	2.201930

Рисунок 30 — Метрики работы нейросети MLPRegressor на тестовом множестве

2.4.2 Нейросеть из библиотеки TensorFlow

В ходе выполнения выпускной квалификационной работы была построена нейронная сеть с помощью класса keras.Sequential со следующими параметрами:

1) входной слой для 12 признаков;
2) выходной слой для 1 признака;
3) скрытых слоев: 8;
4) нейронов на каждом скрытом слое: 24;
5) активационная функция скрытых слоев: relu;
6) оптимизатор: Adam;

7) loss-функция: MeanAbsolutePercentageError.

Архитектура нейросети приведена на рисунках 31 и 32.

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: 4,537 (17.72 KB)

Trainable params: 4,537 (17.72 KB)

Non-trainable params: 0 (0.00 B)

Рисунок 31 — Архитектура нейросети в виде summary

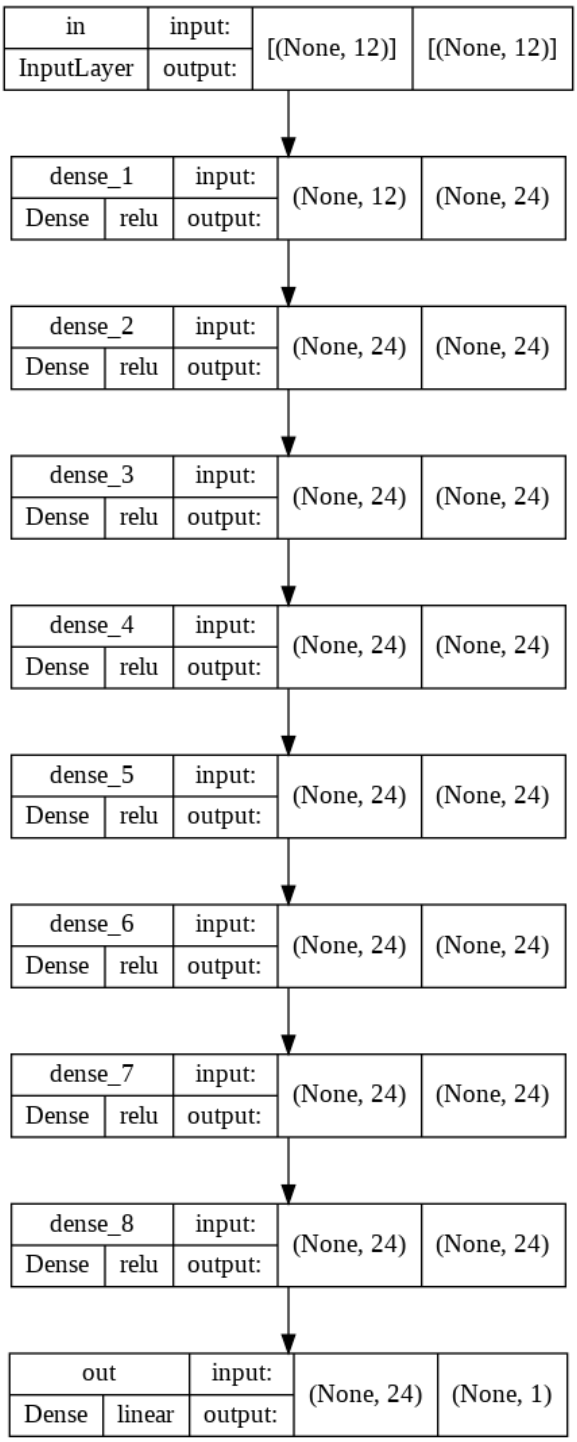

Рисунок 32 — Архитектура нейросети в виде графа

Обучение нейросети было запущено со следующими параметрами:

1) пропорция разбиения данных на тестовые и валидационные: 30%;
2) количество эпох: 50.
3) раннюю остановку не использую.

График обучения приведен на рисунке 33, ошибка — в таблице 4.

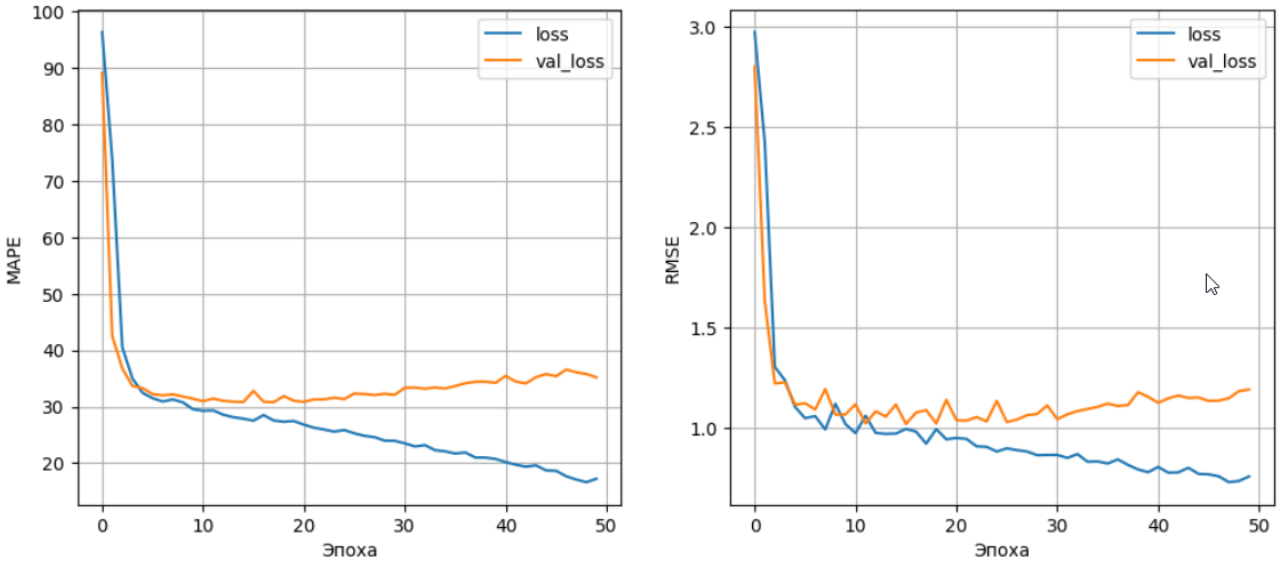

Рисунок 33 — График обучения нейросети

Видно, что примерно до восьмой эпохи обучение шло хорошо, а потом сеть начала переобучаться. Значение loss на тестовых выборках продолжило уменьшаться, а на валидационной выборки начало расти.

Одним из способов борьбы с переобучением может быть ранняя остановка обучения, если val_loss начинает расти. С этой целью в TensorFlow используются callbacks. Соответственно была взята нейросеть с той же архитектурой и запущено обучение с ранней остановкой. График обучения приведен на рисунке 34, а ошибка — в таблице 4. Очевидно, что решение проблемы переобучения повышает 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. Такие слои выключат 5% случайных нейронов на каждом слое.

График обучения приведен на рисунке 35, а ошибка — 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.

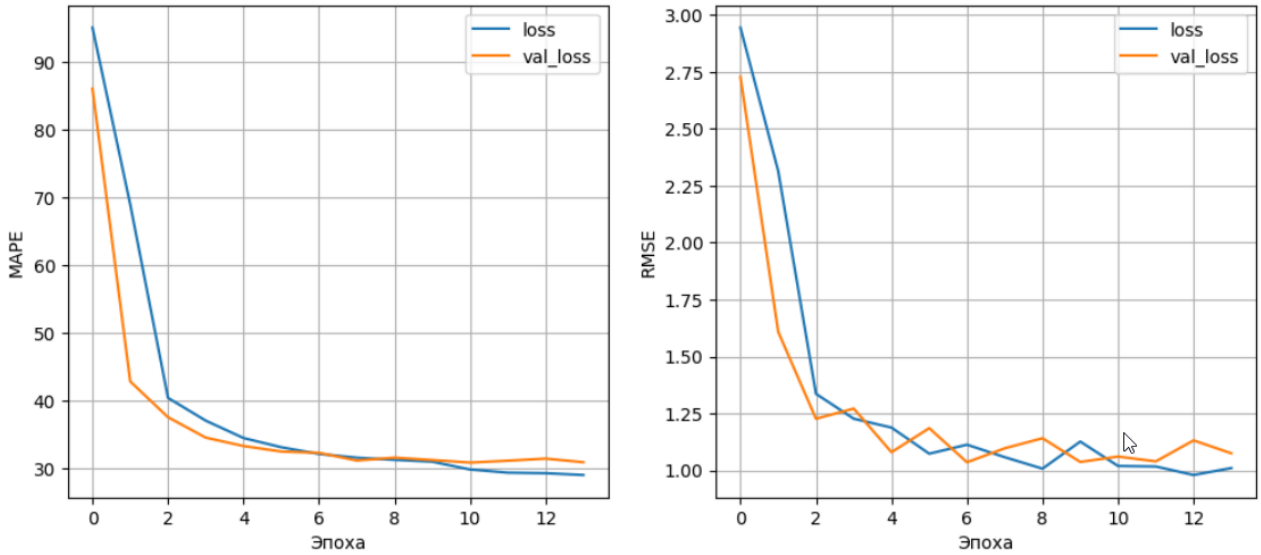

Рисунок 34 — График обучения нейросети с ранней остановкой

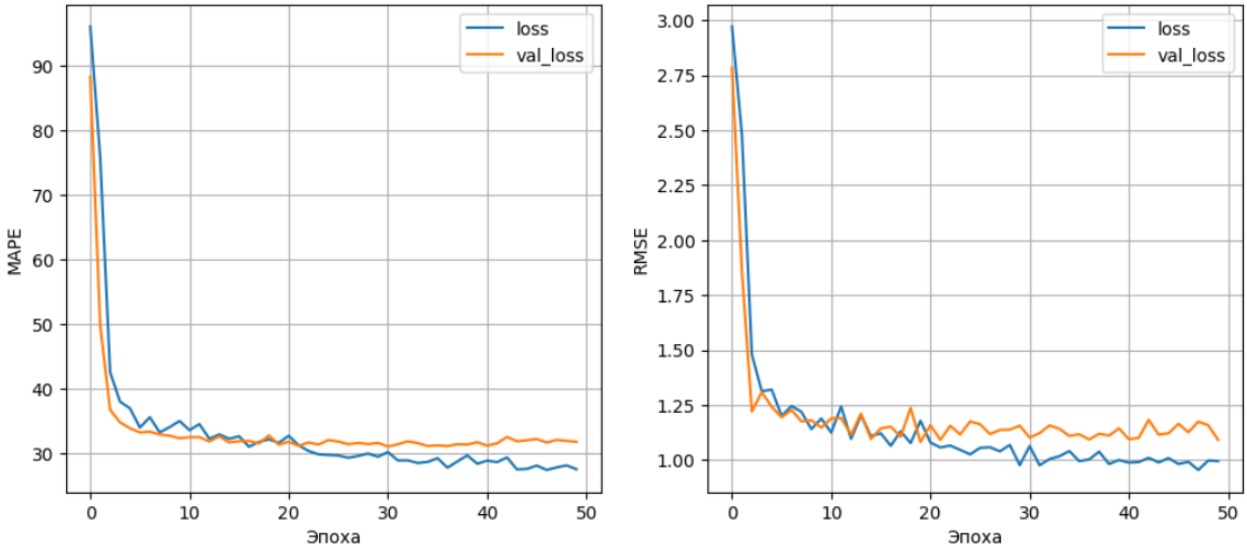

Рисунок 35 — График обучения нейросети с Dropout-слоем

Таблица 4. Борьба с переобучением нейросети

	Эпох	Ошибка на тестовых данных, %	Время обучения, с
Нейросеть переобученная	50	36,18	17,4
Нейросеть с ранней остановкой	14	31,14	7,55
Нейросеть с dropout-слоями	50	31,06	19,3

Визуализация результатов работы нейросетей отображена на рисунке 36, а их метрики — на рисунке 37.

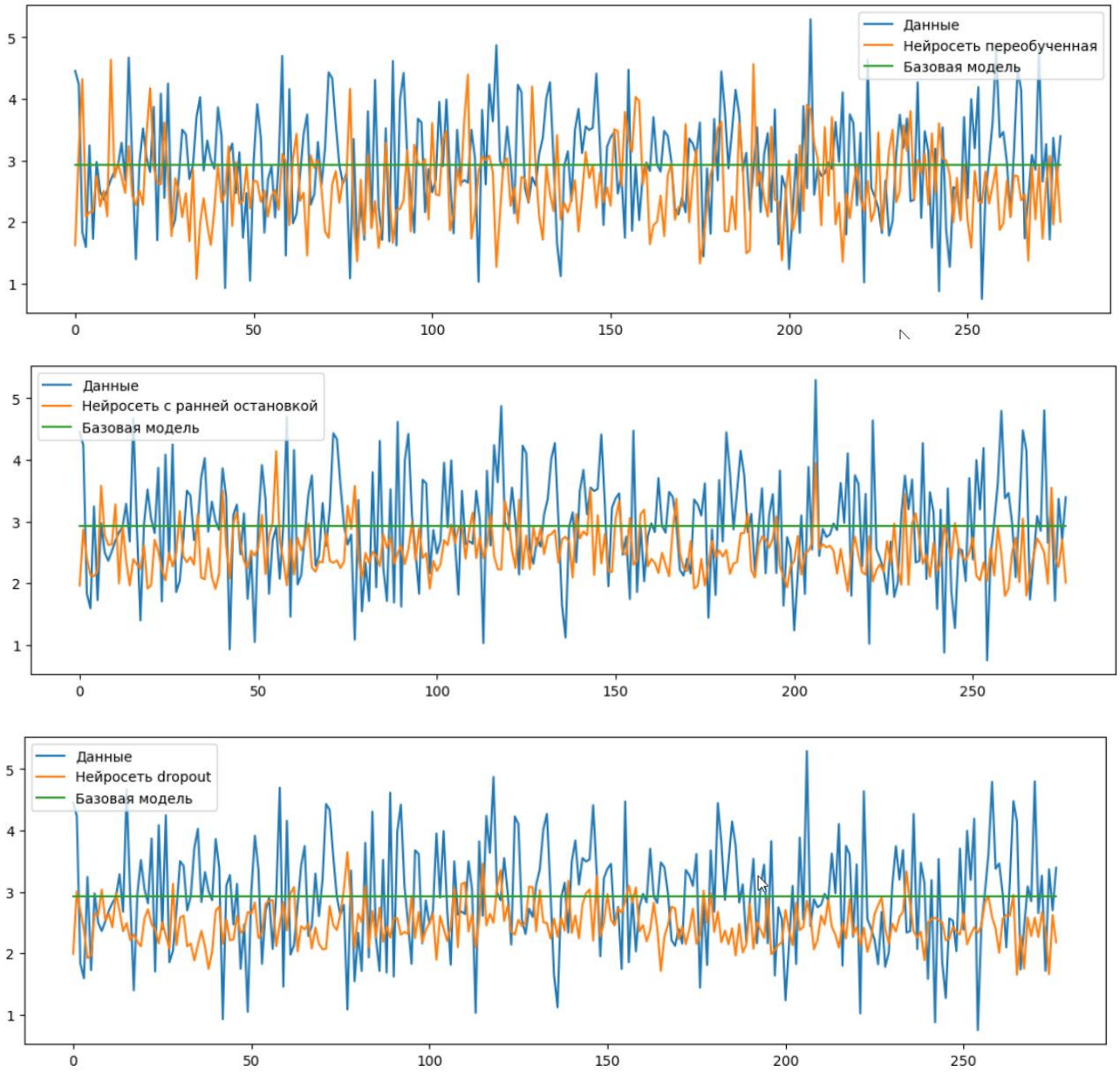

Рисунок 36 - Визуализация результатов работы нейросетей

	R2	RMSE	MAE	MAPE	max_error
DummyRegressor	-0.000074	0.868130	0.691547	0.296759	2.370117
Нейросеть переобученная	-0.830760	1.174585	0.957319	0.361797	3.604231
Нейросеть с ранней остановкой	-0.462496	1.049823	0.847659	0.311427	2.732753
Нейросеть dropout	-0.468733	1.052059	0.856727	0.310578	2.559960

Рисунок 37 -Метрики работы нейросетей на тестовом множестве

Согласно визуализации результатов, представленной на рисунке 36, нейросеть из библиотеки TensorFlow старалась подстроиться к данным. Выглядят результаты «похоже», но в то же время метрики неудовлетворительные. Лучшая обобщающая способность и меньшие значения ошибок на тестовом множестве оказались у нейросети, обученной с ранней остановкой. Но и она предсказывает гораздо хуже базовой модели.

2.5 Тестирование модели

В процессе выполнения работы осуществлено сравнение ошибки каждой модели на тренировочной и тестовой части выборки.

Модель для предсказания модуля упругости при растяжении – Ridge (alpha=700, positive=True, solver='lbfgs'). Сравнение ее ошибок показано на рисунке 38.

	R2	RMSE	MAE	MAPE	max_error
Модуль упругости, тренировочный	0.013531	3.004559	2.419611	0.033056	7.680848
Модуль упругости, тестовый	-0.009474	3.034520	2.475754	0.033873	7.730087

Рисунок 38 - Сравнение ошибок модели для модуля упругости при растяжении на тренировочном и тестовом датасете

Метод Ridge имеет ошибку на тренировочном датасете меньше, чем на тестовом, обучение незначительное имеет место. Но даже на тренировочном датасете он не нашёл закономерности во входных данных. Задачу решить в целом не удалось. Если модуль упругости при растяжении лежит в диапазоне [64,05-82,68], то наша модель делает предсказание с точностью ±7,73, работает не точнее среднего, и скорее бесполезна для применения в реальных условиях.

	R2	RMSE	MAE	MAPE	max_error
Прочность при растяжении, тренировочный	0.018043	454.934526	363.717442	0.159240	1295.547126
Прочность при растяжении, тестовый	-0.011555	441.866376	350.404421	0.151146	1178.211769

Рисунок 39 - Сравнение ошибок модели для прочности на тренировочном и тестовом датасете

Модель для предсказания прочности при растяжении - Lasso (alpha=20). Сравнение ее ошибок показано на рисунке 39.

Метод линейной регрессии Lasso имеет ошибку на тестовом датасете меньше, чем на тренировочном, а вот коэффициент детерминации отрицательный в отличие от положительного значения тренировочного датасета, хотя они оба близки к нулю. Ошибка на тестовом множестве незначительно меньше, чем на тренировочном. В целом эта задача также не решена.

Если прочность при растяжении лежит в диапазоне [1071,12-3848,44], то наша модель дает предсказание с точностью ±1178,21. Она работает не точнее среднего, и бесполезна для применения в реальных условиях.

Модель для предсказания соотношения матрица-наполнитель — нейросеть из TensorFlow, обученная с ранней остановкой. Сравнение ее ошибок показано на рисунке 40.

	R2	RMSE	MAE	MAPE	max_error
Соотношение матрица-наполнитель, тренировочный	-0.278531	1.024750	0.805735	0.292013	3.013253
Соотношение матрица-наполнитель, тестовый	-0.462496	1.049823	0.847659	0.311427	2.732753

Рисунок 40 - Сравнение ошибок модели для соотношения матрица-наполнитель на тренировочном и тестовом датасете.

У нейросети показатели для тестовой выборки сильнее отличаются в худшую сторону от показателей тренировочной. Это говорит о том, что она не нашла закономерностей, а стала учить данные из тестовой выборки. Возможно, требуется более тщательное и грамотное построение архитектуры нейронной сети, чтобы получить лучший результат. Но сейчас задача далека от решения.

Если соотношение матрица-наполнитель лежит в диапазоне [0,39-5,59], то наша модель может предсказать с точностью ±2,73. Она работает не точнее среднего, и бесполезна для применения в реальных условиях.

2.6 Разработка приложения

Разработанные модели имели низкую предсказательную способность, но тем не менее, в целях выполнения поставленной задачи был разработан функционал приложения. Можно предположить, что дальнейшие исследования позволят построить качественную модель и внедрить ее в готовое приложение.

В приложении были реализованы следующие функции:

1) выбор целевой переменной для предсказания;
2) ввод входных параметров;
3) проверка введенных параметров;
4) загрузка сохраненной модели, получение и отображение прогноза выходных параметров.

В ходе выполнения выпускной квалификационной работы было разработано веб-приложение с помощью языка Python, фреймворка Flask и шаблонизатора Jinja.

Задача решена успешно, скриншоты разработанного веб-приложения приведены в приложении А.

2.7 Создание удаленного репозитория

Для данного исследования был создан удаленный репозиторий на GitHub, который находится по адресу https://github.com/ponikarovskikhaa-lgtm/DataScience_bmstu_course. В репозиторий были загружены результаты работы: исследовательский notebook, код приложения (app.py), модели, препроцессинг, датасет, использованная литература, файл README.md, файлы с презентацией и пояснительной запиской.

Одновременно было осуществлено размещение приложения на вэб-хостинге сервиса render.com под адресу: https://datascience-bmstu-course-1-prod.onrender.com, предварительно выполнив следующие шаги:

1) регистрация на render.com;
2) создание в репозитории GitHub файла requirements.txt, содержащего все необходимые наименования и версии библиотек для работы приложения, в т.ч. библиотека Gunicorn;

3) загрузка в репозиторий вэб-хостинга render.com репозитория проекта с GitHub;
4) развертывание Flask-приложения;
5) проверка работоспособности приложения.

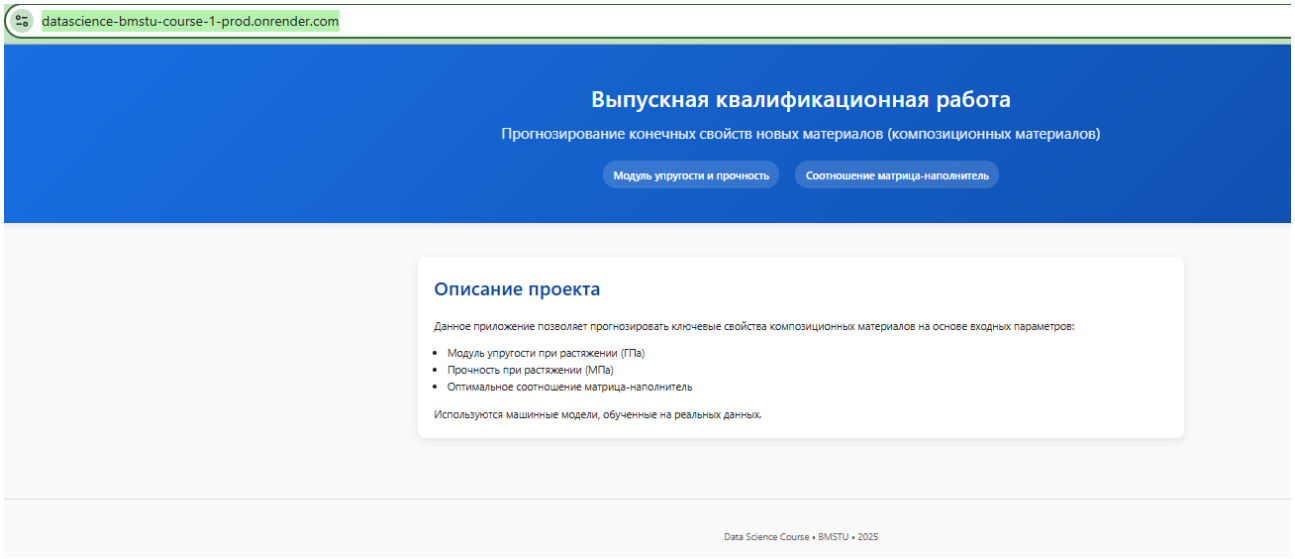

Рисунок 41 – Стартовый вэб-интерфейс размещенного приложения

На рисунке 41 показан стартовый интерфейс Flask-приложения по прогнозированию конечных свойств новых материалов (композиционных материалов). При клике на кнопку «Модуль упругости и прочность» происходит переход на страницу для задания входных параметров для расчета, что видно на рисунке 42. С этой страницы при клике на кнопку «Назад к выбору модели» можно вернуться на стартовую страницу.

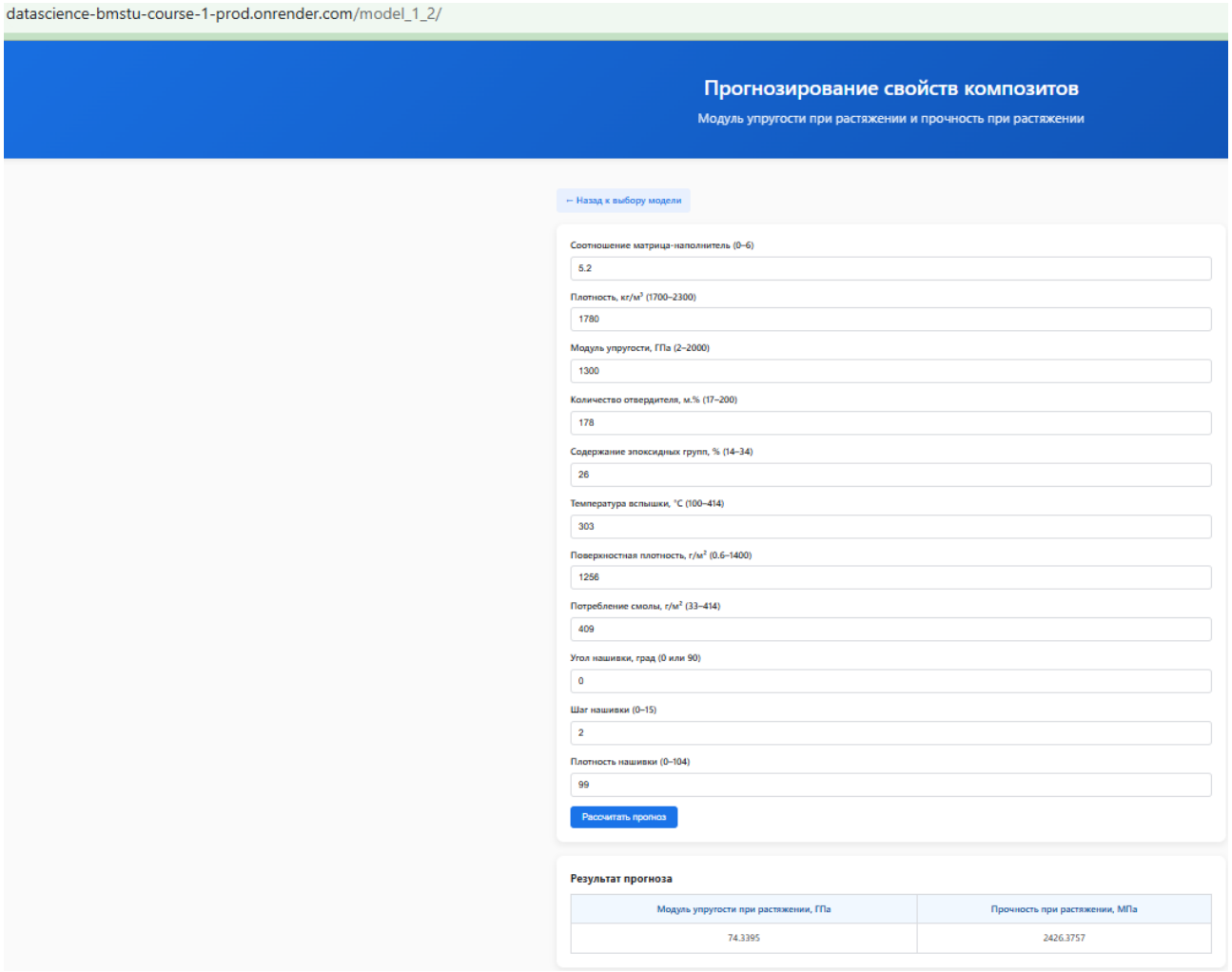

Рисунок 42 – Расчёт на вэб-хостинге прогноза модуля упругости и прочности при растяжении по введенным вручную входным параметрам

При клике на кнопку «Соотношение матрица-наполнитель» осуществляется переход на страницу для расчёта показателя соотношение матрица-наполнитель, исходя из введенных пользователем входных параметров матрицы и наполнителя. Интерфейс данной страницы и примером входных и предсказанного приложением результирующего показателя представлен на рисунке 43.

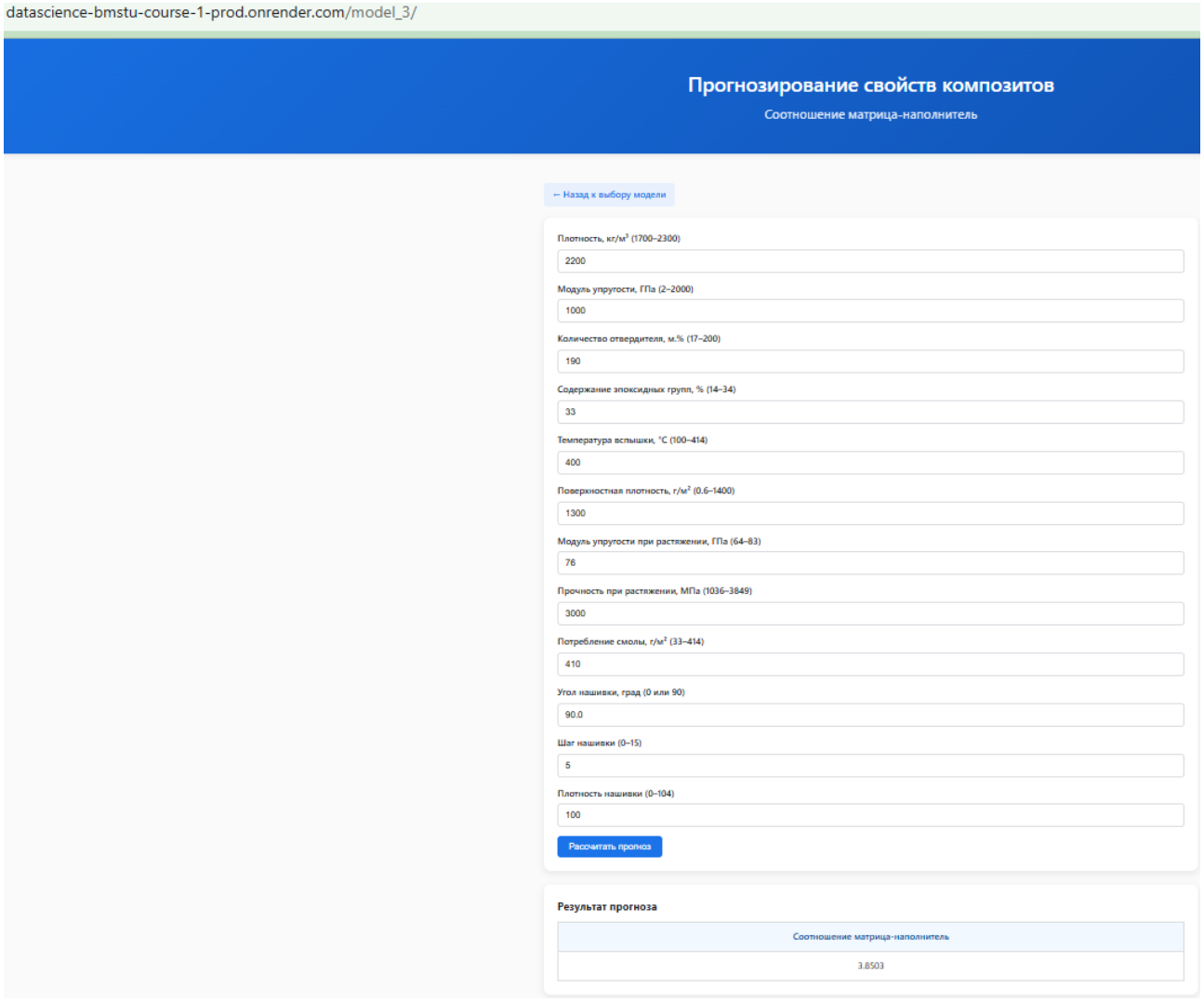

Рисунок 43 – Расчёт на вэб-хостинге прогноза соотношения матрица-наполнитель по введенным вручную входным параметрам

Заключение

В ходе выполнения данной выпускной квалификационной работы пройден практически весь Dataflow pipeline, рассмотрен значительный объём операций и задач, которые приходится выполнять специалисту по работе с данными.

Этот поток операций и задач включает:

1) изучение теоретических методов анализа данных и машинного обучения;
2) изучение основ предметной области, в которой решается задача;
3) извлечение и трансформацию данных. В связи с тем, что по условиям задания был предоставлен уже готовый набор данных, трудности работы с разными источниками и парсингом данных не возникли;
4) проведение разведочного анализа данных статистическими методами;
5) DataMining — извлечение признаков из датасета и их анализ;
6) разделение имеющихся размеченных данных на обучающую, валидационную, тестовую выборки;
7) выполнение предобработки (препроцессинга) данных для обеспечения корректной работы моделей;
8) построение аналитического решения, включающего в себя выбор алгоритма решения и модели, сравнение различных моделей, подбор гиперпараметров модели;
9) визуализация модели и оценка качества аналитического решения;
10) сохранение моделей;
11) разработка и тестирование приложения для поддержки принятия решений специалистом предметной области, которое использовало бы найденную модель;
12) внедрение решения и приложения в эксплуатацию.

В данный выпускной квалификационной работе использовались исходные данные реальной производственной задачи, а не учебные. Но тем не менее, использование различных методов машинного анализа не позволило решить

поставленную задачу прогнозирования конечных свойств новых композиционных материалов, а именно: на выходе были получены модели, которые бы неточно описывали закономерности предметной области. В то же время мною были применены полученные в ходе прохождения курса основные знания и навыки.

Возможные причины неудачи:

1) 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;
2) исследование предварительно обработанных данных. Возможно, на "сырых", не предварительно обработанных данных можно было бы получить более качественные модели, воспользовавшись другими методами очистки и подготовки;
3) недостаток моих знаний и опыта как исследователя. Нейросети являются самым современным подходам 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 могут быть:

1) консультация с экспертами в предметной области, которые могли бы поделиться знаниями, необходимыми для решения задачи;
2) углубление в изучение нейросетей, использование различной архитектуры, параметров обучения и т.д.;
3) осуществление отбора признаков разными методами, использование метода уменьшения размерности, например метод главных компонент;
4) после уменьшения размерности метод градиентного бустинга может улучшить свои результаты. Так же есть большой простор для подбора гиперпараметров для этого метода.

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 языку программирования python: – Режим доступа: https://docs.python.org/3.8/index.html.

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

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

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

8. Документация по библиотеке seaborn: – Режим доступа: https://seaborn.pydata.org/tutorial.html.

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

10. Документация по библиотеке keras: – Режим доступа: https://keras.io/api/.

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

12. Loginom Вики. Алгоритмы: – Режим доступа: https://wiki.loginom.ru/algorithms.html.

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

14. 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.

15. 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/.

16. Yury Kashnitsky. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес: – Режим доступа: https://habr.com/ru/company/ods/blog/324402/.

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

18. 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.

19. Бизли Д. Python. Подробный справочник: учебное пособие. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с., ил.

20. Рашка, Себастьян, Мирджалили, Вахид. Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2, 3-е изд.: Пер. с англ. – СПб.: ООО «Диалектика», 2020. – 848 с.: ил.

21. Справочник по композиционным материалам: в 2 - х кн. Кн. 2 / Под ред. Дж. Любина; Пер. с англ. Ф. Б. Геллера, М. М. Гельмонта; Под ред. Б. Э. Геллера - М.: Машиностроение, 1988. - 488 с.: ил.

22. Траск Эндрю. Грокаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.: ил.

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

24. Роббинс, Дженнифер. HTML5: карманный справочник, 5-е издание.:

Пер. с англ. - М.: ООО «И.Д. Вильямс»: 2015. - 192 с.: ил.

25. Реутов Ю.А.: Прогнозирование свойств полимерных композиционных материалов и оценка надёжности изделий из них, Диссертация на соискание учёной степени кандидата физико-математических наук, Томск 2016.

26. Плас Дж. Вандер, Python для сложных задач: наука о данных и машинное обучение. Санкт-Петербург: Питер, 2018, 576 с.

27. Материалы конференции: V Всероссийская научно-техническая конференция «Полимерные композиционные материалы и производственные технологии нового поколения», 19 ноября 2021 г.

28. Ларин А. А. Способы оценки работоспособности изделий из композиционных материалов методом компьютерной томографии, Москва, 2013, 148 с.

29. Гафаров, Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учеб. пособие /Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Издательство Казанского университета, 2018. – 121 с.

30. Абу-Хасан Махмуд, Масленникова Л. Л.: Прогнозирование свойств композиционных материалов с учётом наноразмера частиц и акцепторных свойств катионов твёрдых фаз, статья 2006 год.

31. Абросимов Н.А.: Методика построения разрешающей системы уравнений динамического деформирования композитных элементов конструкций (Учебно-методическое пособие), ННГУ, 2010.

32. Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru. 2020. - 412 с.: ил.

Приложение А. Скриншоты веб-приложения

Созданное приложение — это основной файл Flask (app.py – часть кода приведена на рисунке 44), папка templates, с шаблоном html – страницы (main.html – стартовая, model_1_2.html - прогноз модуля упругости и прочность, model_3.html - прогноз соотношения матрица-наполнитель, и features.html - тестовая страница для ввода всех признаков), папка models с сохранёнными моделями данных и функциями препроцессинга данных.

```python
@app.route('/model_1_2/', methods=['post', 'get'])
def model_1_2_page():
    # Необходимые признаки
    features = {
        'var1': 'Соотношение матрица-наполнитель',
        'var2': 'Плотность, кг/м3',
        'var3': 'модуль упругости, ГПа',
        'var4': 'Количество отвердителя, м.%',
        'var5': 'Содержание эпоксидных групп,%_2',
        'var6': 'Температура вспышки, С_2',
        'var7': 'Поверхностная плотность, г/м2',
        'var10': 'Потребление смолы, г/м2',
        'var11': 'Угол нашивки, град',
        'var12': 'Шаг нашивки',
        'var13': 'Плотность нашивки'
    }
    # Переменные для формы
    # params = {'var1': '', 'var2': '', 'var3': '', 'var4': '', 'var5': '', 'var6': '', 'var7': '',
    #           'var10': '', 'var11': '', 'var12': '', 'var13': ''}
    # тестовый пример 19, var8=73.62282622 var9=2519.45385534
    params = dict(zip(features.keys(), ['4.02912621359223', '1880.0', '622.0', '111.86',
                                        '22.2678571428571', '284.615384615384', '470.0', '220.0', '90.0',
                                        '4.0', '60.0']))
    #
    error = ''
    x = pd.DataFrame()
    var8 = ''
    var9 = ''
    # Получены данные из формы
    if request.method == 'POST':
        params = request.form.to_dict()
        data, error = get_data_from_form(features, params)
        if error == '':
            # Входные данные корректны, выполняется логика
            x = pd.DataFrame(data, index=[0])
            # для модуля упругости при растяжении
            preprocessor1 = load_pickle_obj('preprocessor1')
            model1 = load_pickle_obj('model1_best')
            x1 = preprocessor1.transform(x)
            y1 = model1.predict(x1)
            var8 = y1[0]
            # для прочности при растяжении
            preprocessor2 = load_pickle_obj('preprocessor2')
            model2 = load_pickle_obj('model2_best')
            x2 = preprocessor2.transform(x)
            y2 = model2.predict(x2)
            var9 = y2[0]
```

Рисунок 44 — Часть кода файла app.py приложения

После запуска в терминале приложения команды «python app.py», пользователь переходит на: http://127.0.0.1:3000/ (рис. 45).

```
PS C:\Users\Денис\Documents\Data Science\GIT\DataScience_bmstu_course> python app.py
src/models/ www
src/models/ www
 * Serving Flask app 'app'
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:3000
 * Running on http://127.0.0.1:3000
Press CTRL+C to quit
```

Рисунок 45 — Ссылка для открытия html-файла

Скриншоты веб-приложения, иллюстрирующие его работу, приведены на рисунках 46 -52. В ходе выполнения выпускной квалификационной работы были реализованы следующие функции:

1) выбор целевой переменной для предсказания (модуль упругости при растяжении и прочности при растяжении или соотношение матрица-наполнитель);
2) ввод входных параметров;
3) проверка введенных параметров;
4) загрузка сохраненной модели, получение и отображение прогноза выходных параметров.

При проверке введенных параметров считаем, что значения не могут быть пустыми, должны быть вещественными, не могут содержать некорректных символов и должны соответствовать допустимому диапазону.

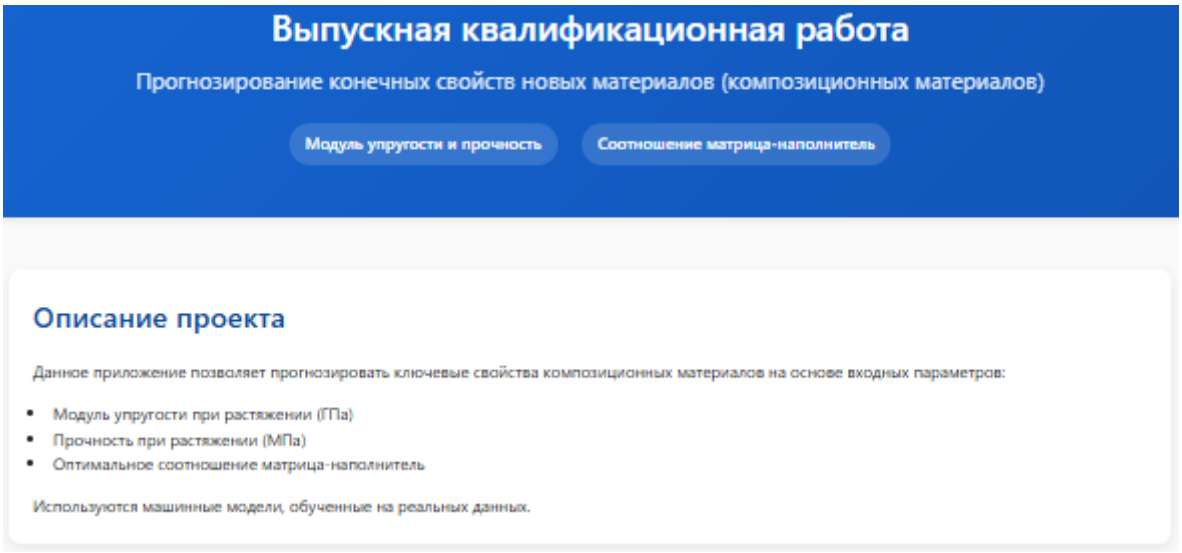

Рисунок 46 — Начальное окно, выбор целевых переменных

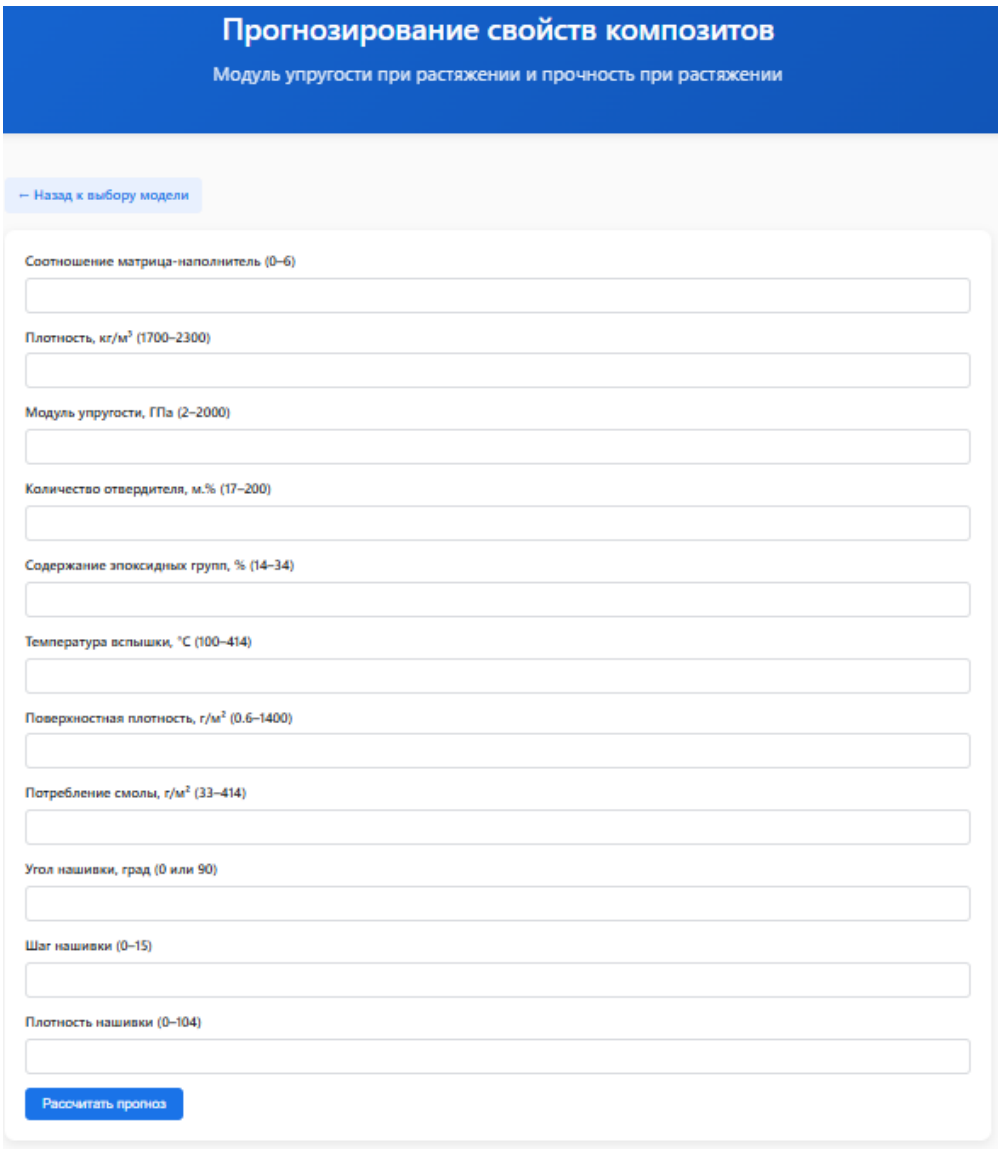

Рисунок 47 - Ввод входных параметров для прогнозирования модуля упругости при растяжении и прочности при растяжении

Рисунок 48 - Проверка входных параметров для прогнозирования модуля упругости при растяжении и прочности при растяжении

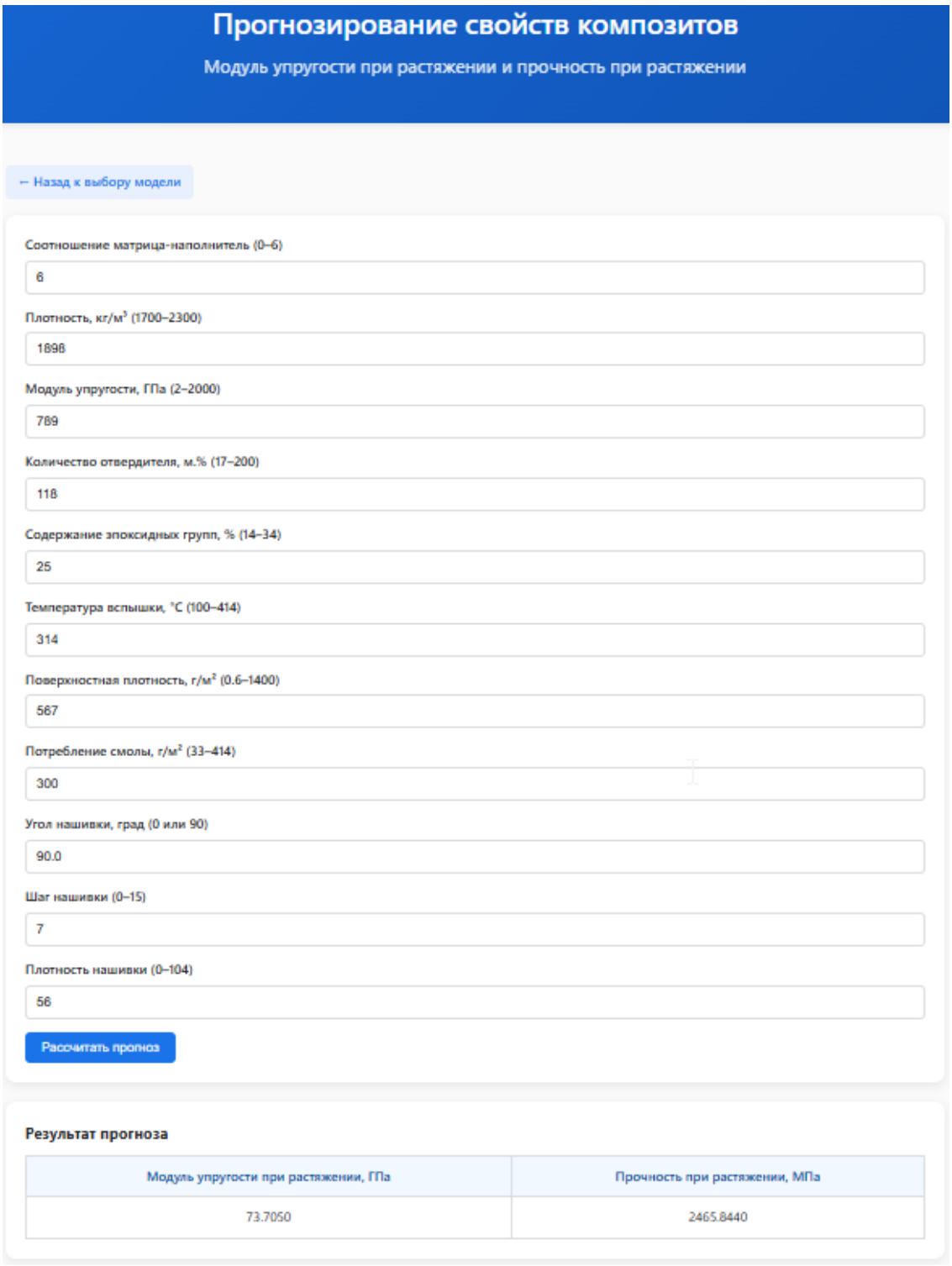

Рисунок 49 - Результат работы модели для модуля упругости при растяжении и прочности при растяжении

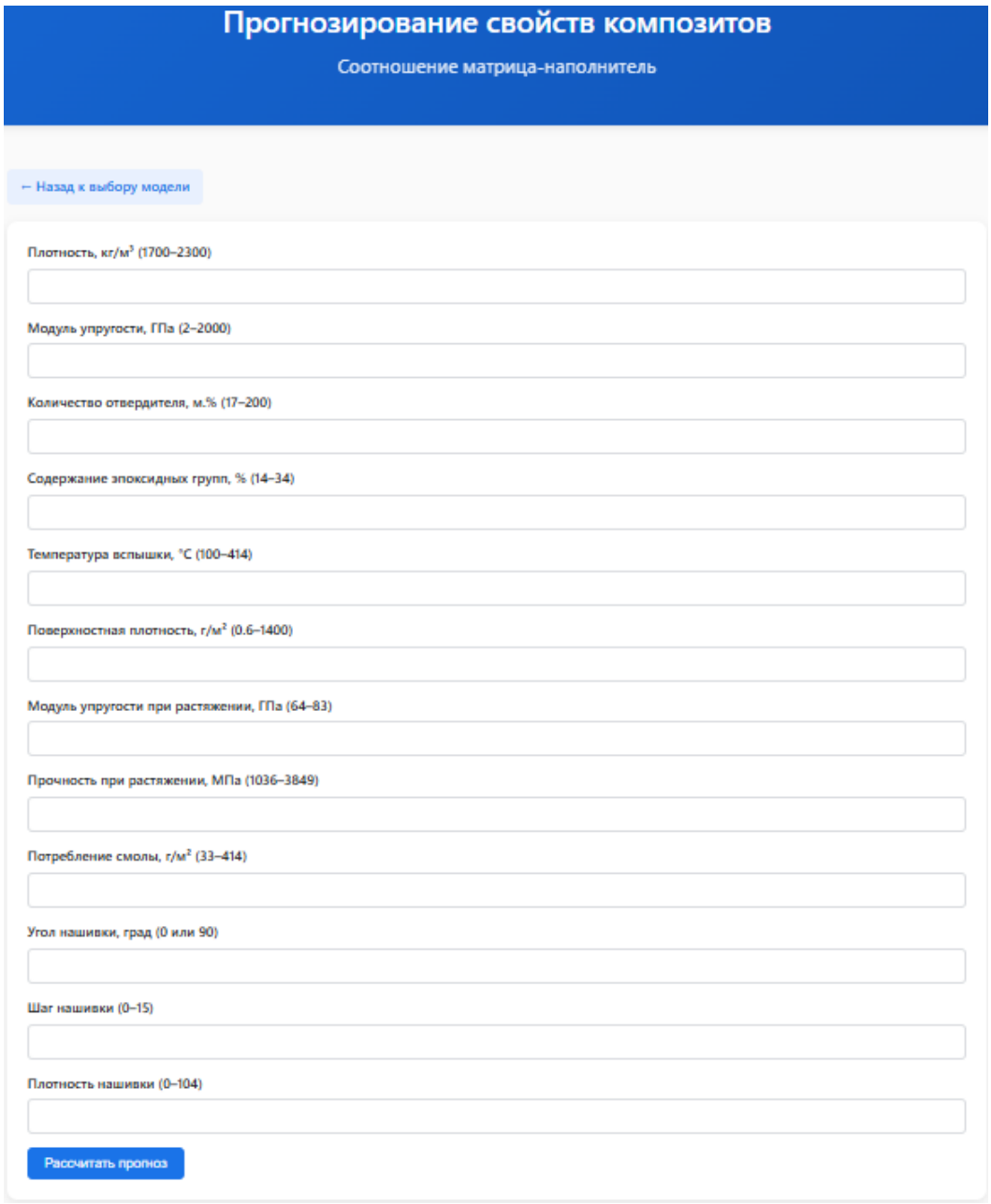

Рисунок 50 - Ввод входных параметров для прогнозирования соотношения матрица-наполнитель

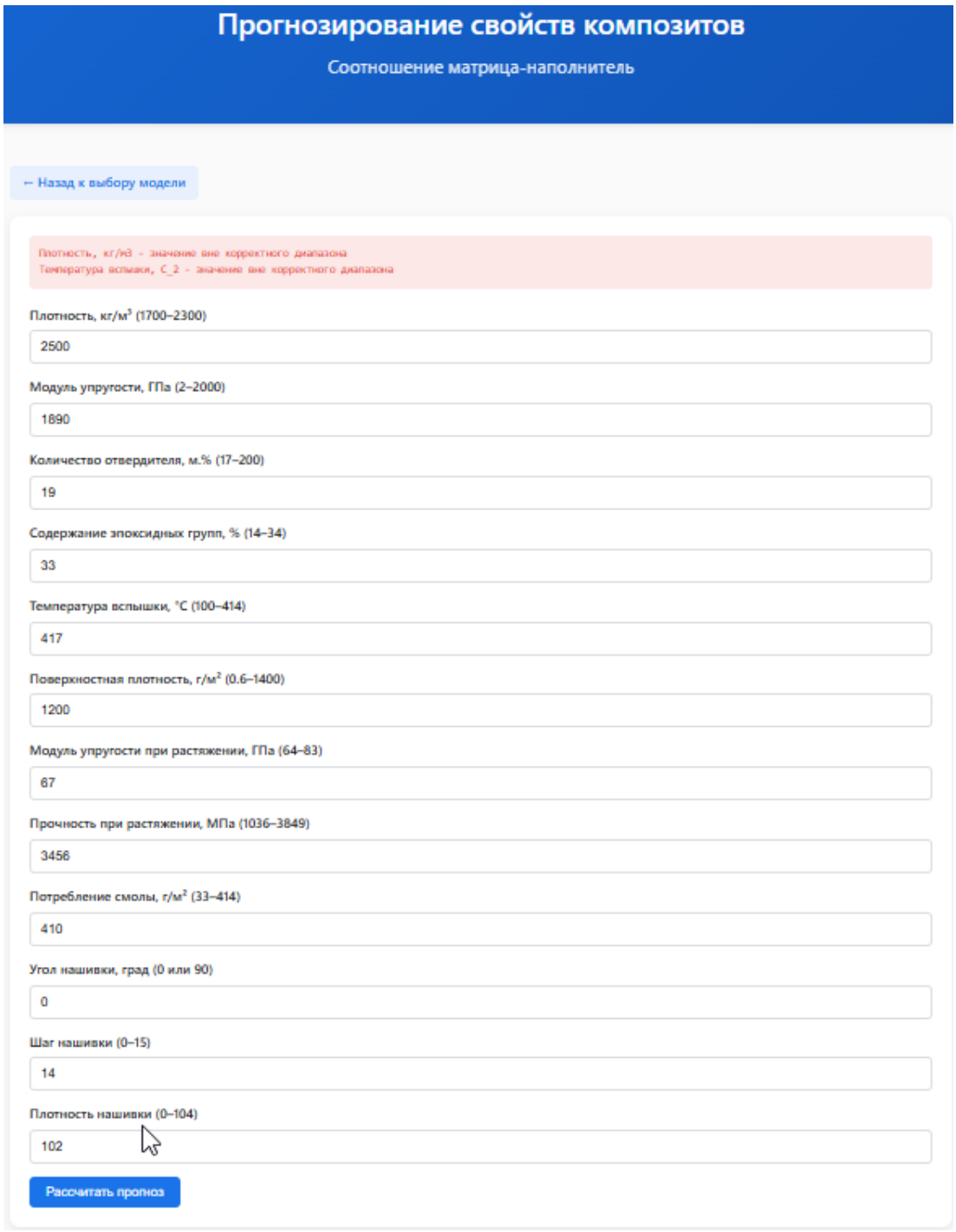

Рисунок 51 - Проверка входных параметров для прогнозирования соотношения матрица-наполнитель

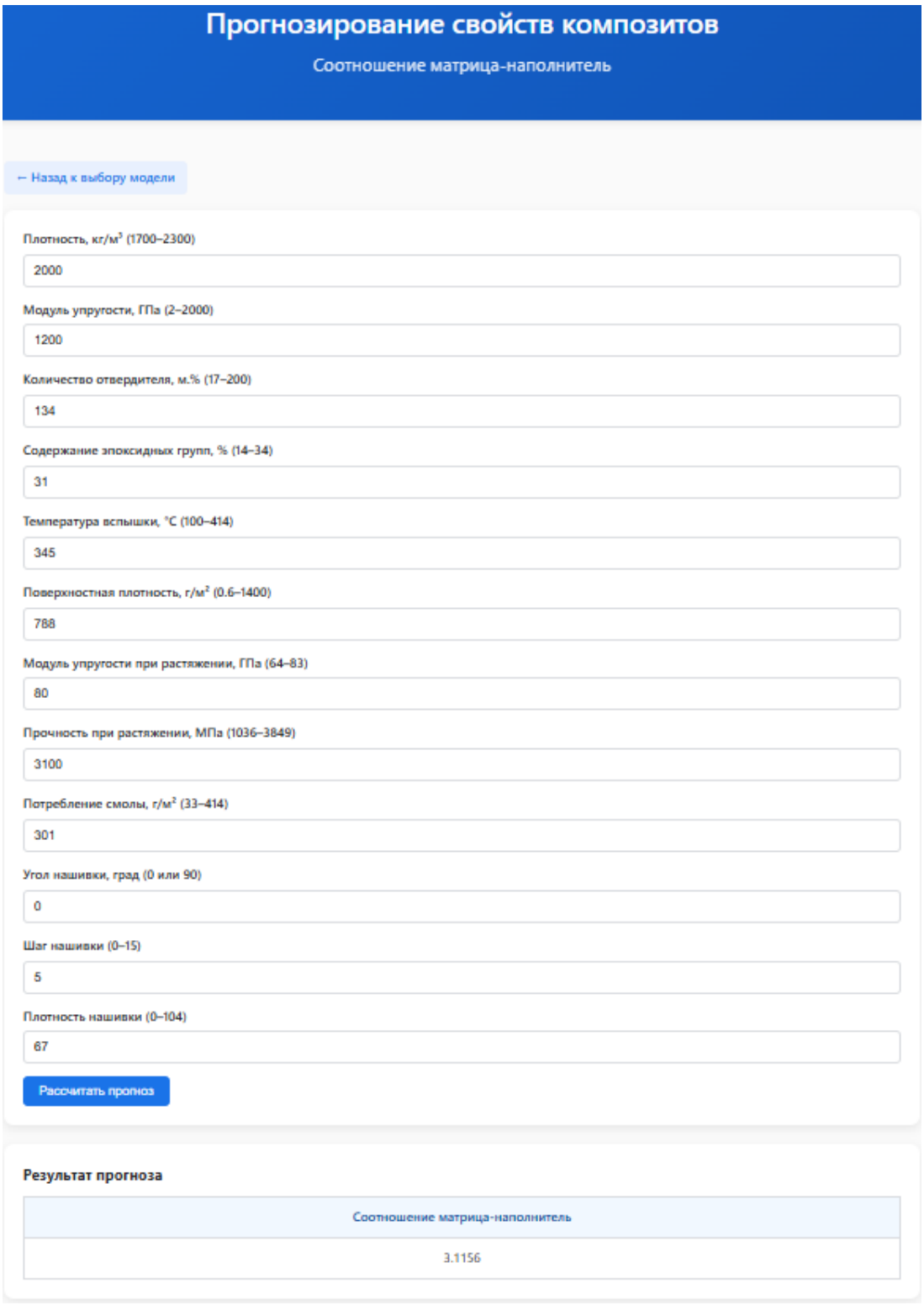

Рисунок 52 - Результат работы модели для соотношения матрица-наполнитель

В процессе выполнения работы проведено тестирование: модели, запущенные в jupyter notebook, где разрабатывалось аналитическое решение, и модели из приложения на одних и тех же данных возвращают одинаковый результат, что

свидетельствует о том, что загрузка моделей и подготовка параметров для моделей выполнены корректно.