

**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

Петров Павел Алексеевич

Москва, 2022

Содержание

Введение	4
1. Аналитическая часть	6
1.1. Постановка задачи	6
1.2. Описание используемых методов	13
1.2.1 Линейная регрессия	14
1.2.2 Лассо (LASSO) и гребневая (Ridge) регрессия	15
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. Предобработка данных	28
2.2 Разработка и обучение моделей для прогнозирования модуля упру- ости при растяжении	31
2.3 Разработка и обучение модели для прогнозирования прочности при растяжении	34
2.4 Тестирование модели	
2.5 Разработка нейронной сети для прогнозирования соотношения мат-	

рица-наполнитель.....	7
2.5.1 MLPRegressor из библиотеки sklearn.....	37
2.5.2 Нейросеть из библиотеки tensorflow	39
2.6 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

Тема работы – это прогнозирование конечных свойств новых материалов (композиционных материалов).

Композиционные материал — многокомпонентный материал, изготовленный (человеком или природой) из двух или более компонентов с существенно различными физическими и/или химическими свойствами, которые, в сочетании, приводят к появлению нового материала с характеристиками, отличными от характеристик отдельных компонентов и не являющимися простой их суперпозицией. В составе композита принято выделять матрицу/матрицы и наполнитель/наполнители, последние выполняют функцию армирования (по аналогии с арматурой в таком композиционном строительном материале, как железобетон). В качестве наполнителей композитов как правило выступают углеродные или стеклянные волокна, а роль матрицы играет полимер. Сочетание разных компонентов позволяет улучшить характеристики материала и делает его одновременно лёгким и прочным. При этом отдельные компоненты остаются таковыми в структуре композитов, что отличает их от смесей и затвердевших растворов. Варьируя состав матрицы и наполнителя, их соотношение, ориентацию наполнителя, получают широкий спектр материалов с требуемым набором свойств. Многие композиты превосходят традиционные материалы и сплавы по своим механическим свойствам и в то же время они легче. Использование композитов обычно позволяет уменьшить массу конструкции при сохранении или улучшении её механических характеристик. 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.

Возможно 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.

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

Стоимость производства композитного материала высока. Зная характеристики компонентов, невозможно рассчитать свойства композита. Значит для получения заданных свойств требуется большое количество испытаний различных комбинаций. Сократить время и затраты на создание определенного материала могла бы помочь система поддержки производственных решений, построенная на принципах машинного обучения.

1. Аналитическая часть

1.1. Постановка задачи

Предмет исследования данной работы – композит с матрицей из базальтопластика и нашивками из углепластика.

Цель работы – разработка моделей, которые могут в достаточной степени предсказывать необходимые характеристики композита.

Задачи:

- 1) проанализировать и предобработать данные из предоставляемого датасета;
- 2) создать, протестировать различные модели и выбрать лучшие из них;
- 3) развернуть приложение, использующее эти модели для предсказания необходимых характеристик композита.

Датасет представляет из себя два файла: X_br (составляющая из базальтопластика) и X_nip (составляющая из углепластика).

Файл X_br содержит 10 признаков и индекс; 1023 строки.

Файл X_nip содержит 3 признака и индекс; 1040 строк.

Эти два файла необходимо объединить по типу “INNER”. После такого объединения результирующий файл содержит 13 признаков и 1023 строки.

Признаки датасета после объединения файлов указаны в таблице 1. Как видно из таблицы, тип данных везде “float64”, пропусков нет. Несмотря на то, что все признаки формально числовые, признак «Угол нашивки» удобно рассматривать как категориальный, так как он принимает всего 2 значения.

Таблица 1 — Признаки 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	float64	1023	2
Шаг нашивки	X_nup	float64	1023	989
Плотность нашивки	X_nup	float64	1023	988

После построения гистограмм распределения и диаграмм «ящик с усами» можно сделать вывод, что значения всех переменных, кроме «Угла нашивки», распределены по нормальному закону. Гистограммы и диаграммы приведены на рисунках 1-3.

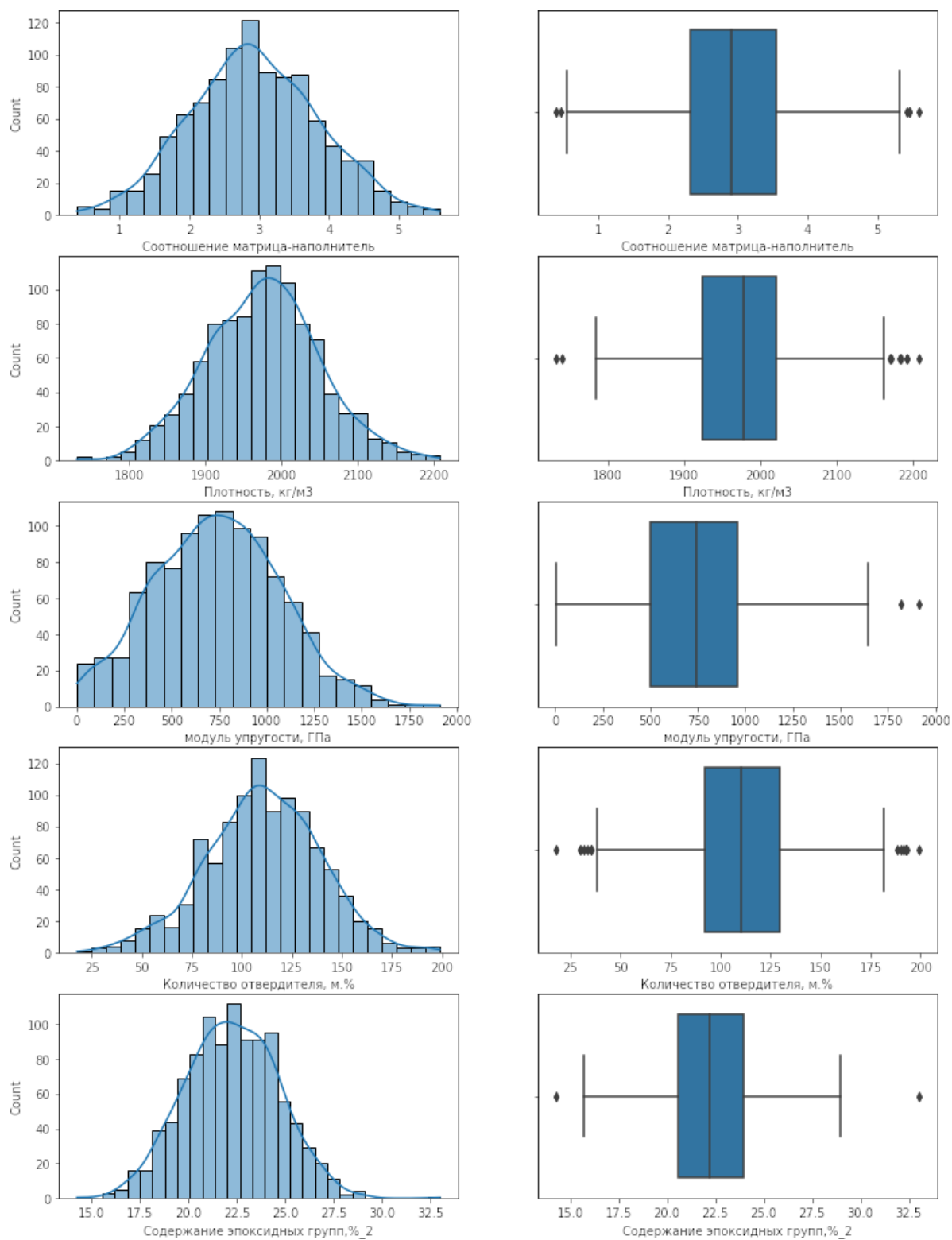


Рисунок 1 - Гистограммы распределения переменных
и диаграммы «ящик с усами»

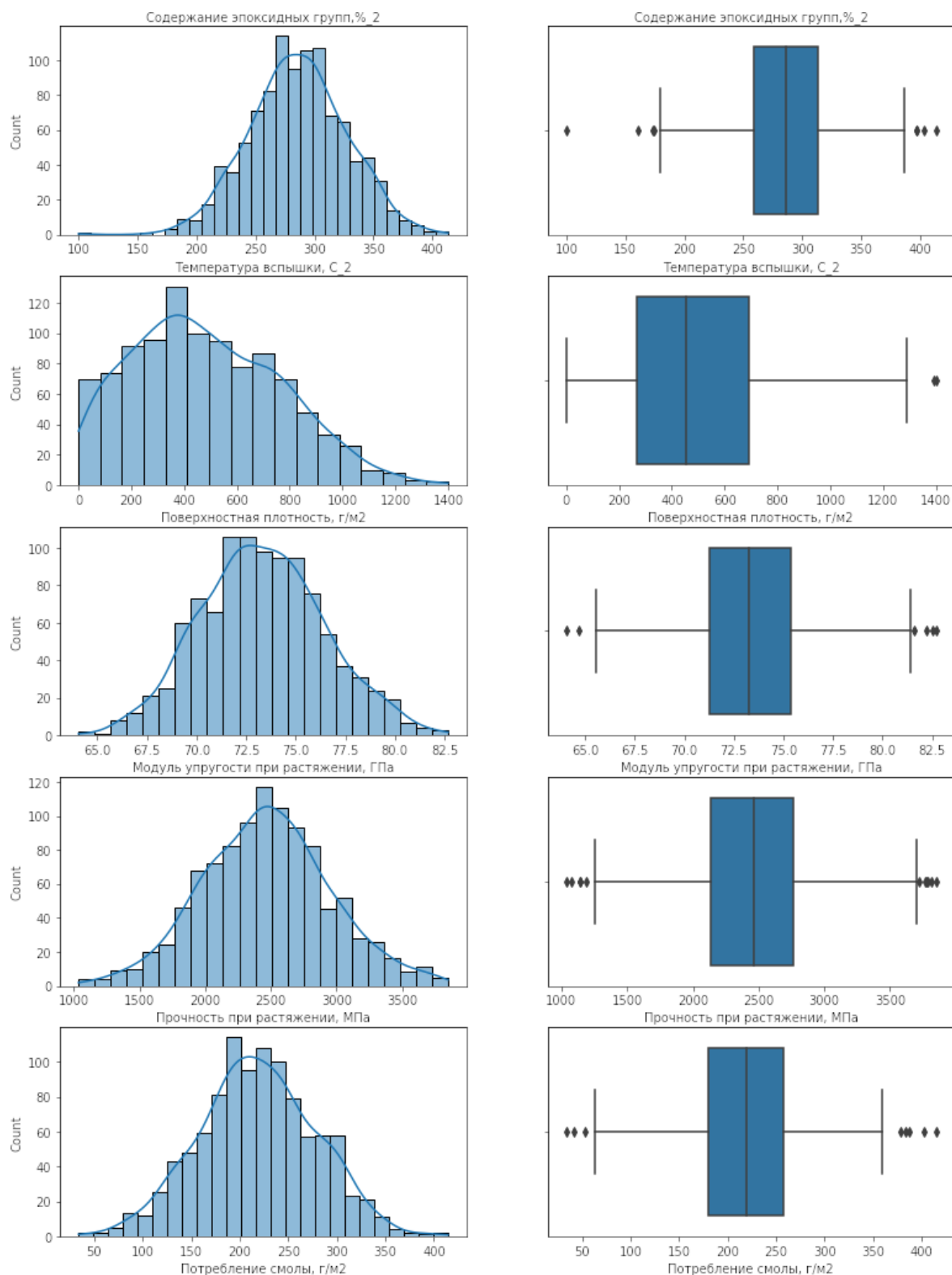


Рисунок 2 - Гистограммы распределения переменных
и диаграммы «ящик с усами»

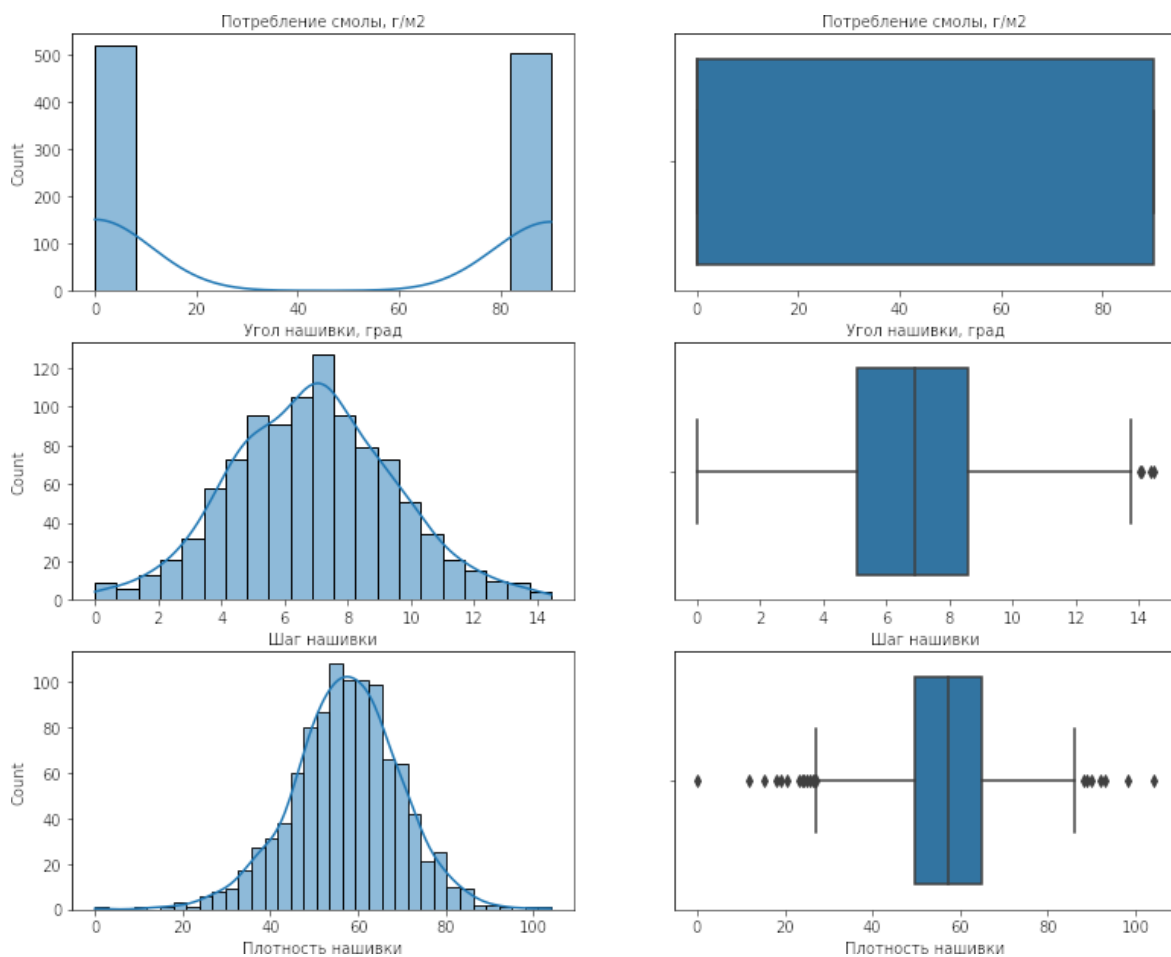


Рисунок 3 - Гистограммы распределения переменных
и диаграммы «ящик с усами»

Описательная статистика датасета представлена в таблице 2.

Попарные графики рассеяния точек приведены на рисунке 4.

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

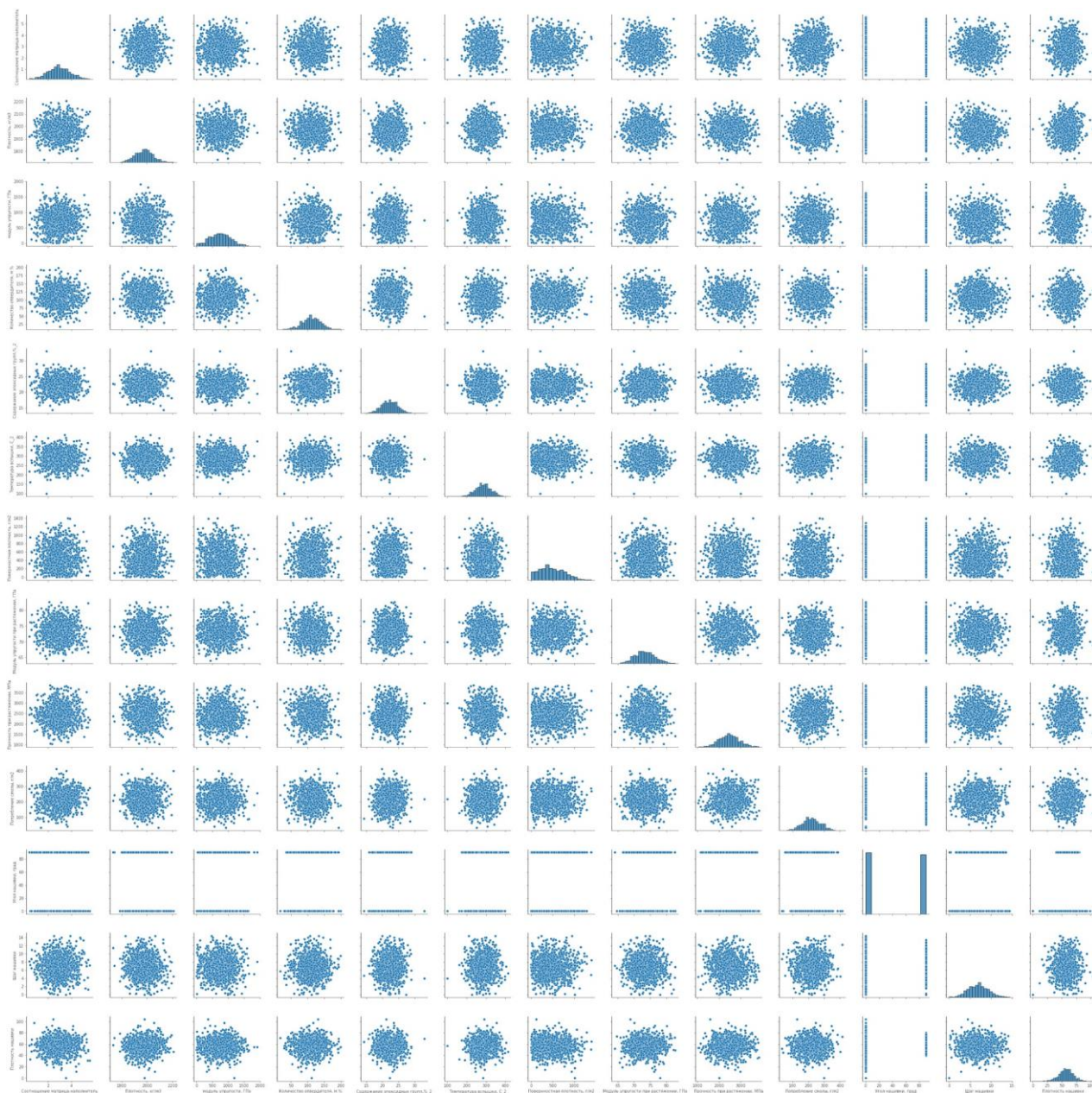


Рисунок 4 — Парные графики рассеяния точек

Для поиска выбросов используем метод трёх сигм. Получаем 24 выброса.

Пример выбросов на гистограмме распределения и диаграмме «ящиксами» приведен на рисунке 5.

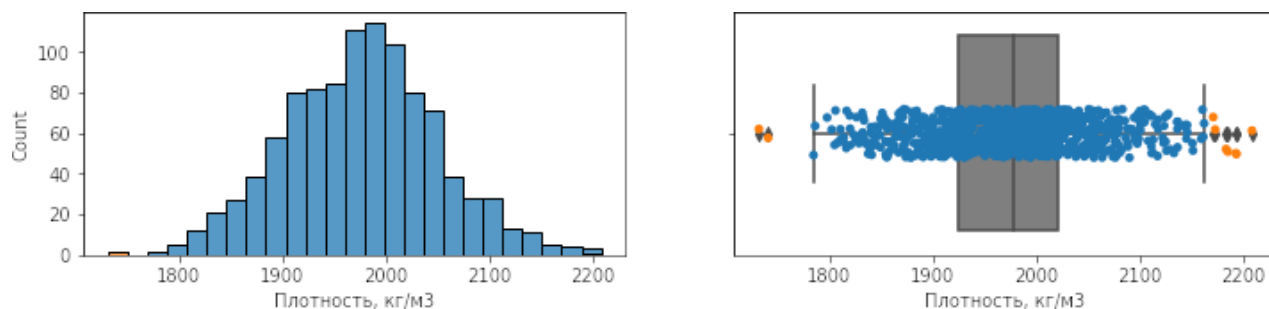


Рисунок 5 — Пример выбросов

После удаления выбросов в датасете стало 1000 строк, то есть датасет стал содержать 1000 строки 13 признаков.

Целевыми переменными для нас являются следующие признаки:

- модуль упругости при растяжении, ГПа;
- прочность при растяжении, МПа;
- соотношение матрица-наполнитель.

1.2. Описание используемых методов

Для предсказания используется метод регрессии и анализ на основе метода регрессии – регрессионный анализ. Существует множество методов регрессионного анализа, рассмотрим некоторые из них.

1.2.1 Линейная регрессия

В математической статистике линейная регрессия представляет собой метод аппроксимации зависимостей между входными и выходными переменными на основе линейной модели.

В регрессионном анализе входные (независимые) переменные называются также предикторными переменными или регрессорами, а зависимые переменные — критериальными.

Если рассматривается зависимость между одной входной и одной выходной переменными, то имеет место простая линейная регрессия. Для этого определяется уравнение регрессии $y=ax+b$ и строится соответствующая прямая,

известная как линия регрессии.

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:

$$Y = b_0 + b_1x_1 + b_2x_2 + \dots + b_nx_n,$$

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 была первым видом регрессионного анализа, который был тщательно изучен и начал широко использоваться в практических приложениях. Это связано с тем, что в линейных моделях оценивание параметров проще, а также с тем, что статистические свойства полученных оценок легче определить.

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

1.2.2 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 сильную корреляцию признаков друг с другом.

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.

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 `sklearn.linear_model.Lasso` и `sklearn.linear_model.Ridge`.

1.2.3 Метод опорных векторов для регрессии

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

Чаще всего он применяется в постановке бинарной классификации.

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

Если в исходном пространстве объекты линейно неразделимы, то выпол-

няется переход в пространство большей размерности.

Решается задача оптимизации.

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

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

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

Преимущество метода — его хорошая изученность.

Недостатки:

- чувствительность к выбросам;
- отсутствие интерпретируемости.

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

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

1.2.4 Метод k-ближайших соседей

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

Этот метод — пример непараметрической регрессии.

Он реализован в `sklearn.neighbors.KNeighborsRegressor`.

1.2.5 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.

Огромное преимущество деревьев решений в том, что они легко интерпретируемы, понятны человеку. Они могут использоваться для извлечения правил на естественном языке. Еще преимущества — высокая точность работы, не требовательность к подготовке данных.

Недостаток деревьев решений - 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`.

1.2.6 Случайный лес

Случайный лес (RandomForest) — представитель ансамблевых методов.

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

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

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:

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

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

Метод реализован в `sklearn.ensemble.RandomForestRegressor`.

1.2.7 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 соответствует 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 ра-

ботает на выборках с «табличными», неоднородными данными и способен эффективно находить нелинейные зависимости в данных различной природы. На настоящий момент это один из самых эффективных алгоритмов машинного обучения. Благодаря этому он широко применяется во многих конкурсах и промышленных задачах. Он проигрывает только нейросетям на однородных данных (изображения, звук и т. д.).

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

В этой работе я использую реализацию градиентного бустинга из библиотеки `sklearn` — `sklearn.ensemble.GradientBoostingRegressor`. Хотя существуют и другие реализации, некоторые из которых более мощные, например, `XDGBoost`.

1.2.8 Нейронная сеть

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

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

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

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

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

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

Прямое распространение — это процесс передачи входных значений в нейронную сеть и получения выходных данных, которые называются прогнозиру-

емым значением.

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

Для обновления весов в модели используются различные оптимизаторы.

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

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

1.3. Разведочный анализ данных

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.

На рисунке 4 мы видели график попарного рассеяния точек. По форме «облаков точек» мы не заметили зависимостей, которые станут основой работы моделей. Помочь выявить связь между признаками может матрица корреляции, приведенная на рисунке 6.

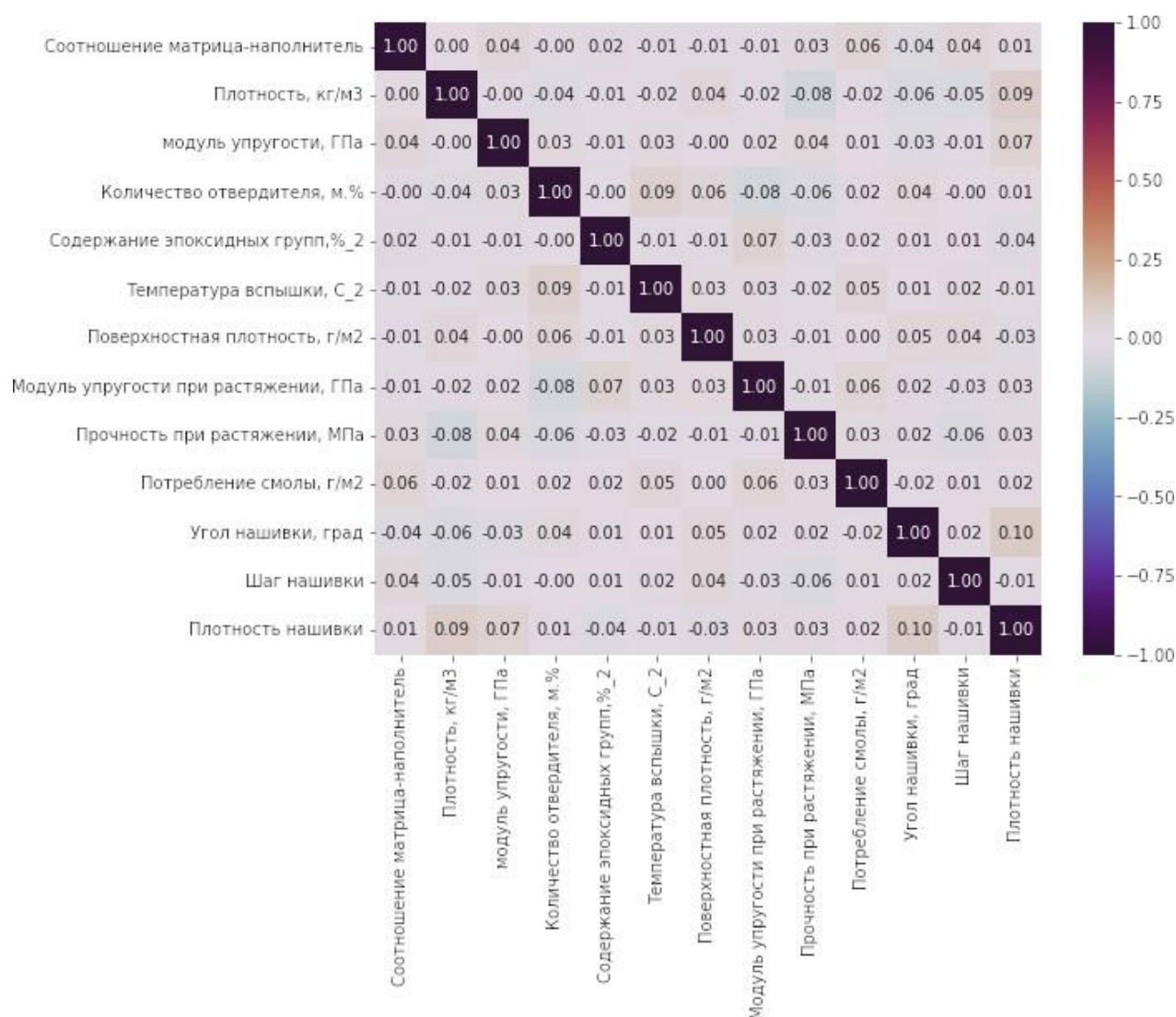


Рисунок 6 — Матрица корреляции

По матрице корреляции мы видим, что все коэффициенты корреляции близки к нулю, что означает отсутствие линейной зависимости между признаками.

1.3.1 Выбор признаков

Статистическими методами мы зависимостей между признаками не обнаружили. Хорошо было бы узнать, какие связи между признаками видит специалист по предметной области.

Можно предположить, что признаки делятся на:

- свойства матрицы;
- свойства наполнителя;
- свойства смеси и производственного процесса;
- свойства готового композита.



Рисунок 7 — Группы признаков
с точки зрения предметной области

1.3.2 Ход решения задачи

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

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

- сравнить метрики моделей после подбора гиперпараметров и выбрать лучшую;
- получить предсказания лучшей и базовой моделей на тестовой выборке, сделать выводы;
- сравнить качество работы лучшей модели на тренировочной и тестовой выборке.

1.3.2 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.

Вещественных количественных признаков у нас большинство. Проблема вещественных признаков в том, что их значения лежат в разных диапазонах, в разных масштабах. Это видно в таблице 2. Необходимо провести одно из двух возможных преобразований:

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

Буду использовать стандартизацию и StandardScaler.

Препроцессинг необходимо повторить в приложении для введенных данных. Поэтому удобно реализовать предварительную обработку с помощью

ColumnTransformer, а потом сохранить и загрузить этот объект аналогично объекту модели.

Выходные переменные никак не изменяю.

1.3.3 Перекрестная проверка

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

1.3.4 Поиск гиперпараметров по сетке

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

1.3.5 Метрики качества моделей

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

- R^2 или коэффициент детерминации измеряет долю дисперсии, объясненную моделью, в общей дисперсии целевой переменной. Если он близок к единице, то модель хорошо объясняет данные, если же он близок к нулю, то прогнозы сопоставимы по качеству с константным предсказанием;
- RMSE (Root Mean Squared Error) или корень из средней квадратичной ошибки принимает значения в тех же единицах, что и целевая переменная.

Метрика использует возведение в квадрат, поэтому хорошо обнаруживает грубые ошибки, но сильно чувствительна к выбросам;

- MAE (Mean Absolute Error) - средняя абсолютная ошибка так же принимает значения в тех же единицах, что и целевая переменная;

- MAPE (Mean Absolute Percentage Error) или средняя абсолютная процентная ошибка — безразмерный показатель, представляющий собой взвешенную версию MAE;

- max error или максимальная ошибка данной модели в единицах измерения целевой переменной.

RMSE, MAE, MAPE и max error принимают положительные значения. Но отображать я их буду со знаком «-». Так корректно отработает выделение цветом лучших моделей — эти метрики надо минимизировать.

R2 в норме принимает положительные значения. Эту метрику надо максимизировать. Отрицательные значения 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 8. Описательная статистика входных признаков до и после предобработки показана на рисунке 9. Описательная статистика выходного признака показана на рисунке 10.

```
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

# Описательная статистика входных данных до предобработки											
	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	1784.482245	4.339154	29.956150	15.881668	173.484920	1.668002	41.048278	0.000000	0.145034	20.571633
max	5.455566	2192.297637	1628.000000	192.851702	28.955094	403.652861	1291.340115	383.663401	90.000000	14.440522	89.876616
mean	2.907441	1972.568298	743.180523	110.927781	22.237431	286.158802	482.049370	218.246969	43.457143	6.890381	57.664603
std	0.908368	70.846531	326.626902	27.485979	2.347655	39.974223	273.399143	59.487795	45.005702	2.536808	11.880191

# Описательная статистика входных данных после предобработки											
	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	-2.774030	-2.656736	-2.263652	-2.948031	-2.709217	-2.820679	-1.758326	-2.980870	-2.660891	-3.124486	0.000000
max	2.807176	3.103701	2.710898	2.982703	2.863482	2.941347	2.962224	2.782667	2.978364	2.713344	1.000000
mean	-0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	-0.000000	-0.000000	-0.000000	0.000000	0.482857
std	1.000715	1.000715	1.000715	1.000715	1.000715	1.000715	1.000715	1.000715	1.000715	1.000715	0.500063

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

Описательная статистика выходной переменной

Модуль упругости при растяжении, ГПа

min	64.054061
max	82.682051
mean	73.354026
std	3.066086

Рисунок 10 - Описательная статистика выходного признака 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 11. Описательная статистика входных признаков до и после предобработки показана на рисунке 12. Описательная статистика выходного признака показана на рисунке 13.

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

Рисунок 11 - Размерности тренировочного и тестового множеств после разбиения для 2-й задачи

Описательная статистика входных данных до предобработки

	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	1784.482245	4.339154	29.956150	15.881668	173.484920	1.668002	41.048278	0.000000	0.145034	20.571633
max	5.455566	2192.297637	1628.000000	192.851702	28.955094	403.652861	1291.340115	383.663401	90.000000	14.440522	89.876616
mean	2.907441	1972.568298	743.180523	110.927781	22.237431	286.158802	482.049370	218.246969	43.457143	6.890381	57.664603
std	0.908368	70.846531	326.626902	27.485979	2.347655	39.974223	273.399143	59.487795	45.005702	2.536808	11.880191

Описательная статистика входных данных после предобработки

	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	-2.884928	-2.557447	-2.197166	-2.804896	-2.573362	-2.743752	-1.645837	-3.073555	-2.614982	-3.055713	0.000000
max	2.705821	2.722504	2.731448	2.862586	2.725119	2.894742	2.841009	2.868402	2.872154	2.860053	1.000000
mean	-0.106155	-0.122313	0.045583	0.012279	0.002542	0.016454	0.025438	-0.000407	-0.025869	0.110485	0.482857
std	1.002426	0.917244	0.991474	0.956296	0.951472	0.979261	0.951172	1.031694	0.973721	1.014075	0.500063

Рисунок 12 - Описательная статистика входных признаков до и после предобработки для 2-й задачи

Описательная статистика выходной переменной

Прочность при растяжении, МПа	
min	1071.123751
max	3848.436732
mean	2468.178562
std	487.297434

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

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

Рисунок 14 - Размерности тренировочного и тестового множеств после разбиения для 3-й задачи

Описательная статистика входных данных до предобработки

	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	4.339154	29.956150	15.881668	173.484920	1.668002	64.054061	1071.123751	41.048278	0.000000	0.145034	20.571633
max	2192.297637	1628.000000	192.851702	28.955094	403.652861	1291.340115	82.682051	3848.436732	383.663401	90.000000	14.440522	89.876616
mean	1972.568298	743.180523	110.927781	22.237431	286.158802	482.049370	73.354026	2468.178562	218.246969	43.457143	6.890381	57.664603
std	70.846531	326.626902	27.485979	2.347655	39.974223	273.399143	3.066086	487.297434	59.487795	45.005702	2.536808	11.880191

Описательная статистика входных данных после предобработки

	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	-2.557447	-2.197166	-2.804896	-2.573362	-2.743752	-1.645837	-2.850424	-2.890122	-3.073555	-2.614982	-3.055713	0.000000
max	2.722504	2.731448	2.862586	2.725119	2.894742	2.841009	2.933340	2.901172	2.868402	2.872154	2.860053	1.000000
mean	-0.122313	0.045583	0.012279	0.002542	0.016454	0.025438	0.037102	0.023037	-0.000407	-0.025869	0.110485	0.482857
std	0.917244	0.991474	0.956296	0.951472	0.979261	0.951172	0.951982	1.016120	1.031694	0.973721	1.014075	0.500063

Рисунок 15 - Описательная статистика входных признаков до и после предобработки для 3-й задачи

Описательная статистика выходной переменной

Соотношение матрица-наполнитель

min	0.389403
max	5.455566
mean	2.907441
std	0.908368

Рисунок 16 - Описательная статистика выходного признака для 3-й задачи

2.2 Разработка и обучение моделей для прогнозирования модуля упругости при растяжении

Для подбора лучшей модели для этой задачи были взяты следующие модели:

- LinearRegression — линейная регрессия (раздел 1.2.1);
- Ridge — гребневая регрессия (раздел 1.2.2);
- Lasso — лассо-регрессия (раздел 1.2.2);
- SVR — метод опорных векторов (раздел 1.2.3);
- KNeighborsRegressor — метод ближайших соседей (раздел 1.2.4);
- DecisionTreeRegressor — деревья решений (раздел 1.2.5);
- RandomForestRegressor — случайный лес (раздел 1.2.6)
- GradientBoostingRegressor — градиентный бустинг (раздел 1.2.7).

Метрики работы выбранных моделей с гиперпараметрами по умолчанию, полученные с помощью перекрестной проверки на тестовом множестве, приведены на рисунке 17.

Ни одна из выбранных мной моделей не оказалась подходящей для наших данных.

Коэффициент детерминации R^2 близок к 0 для линейных моделей и метода опорных векторов. 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 метод ближайших соседей и деревья решений.

Случайный лес отработал лучше, чем одно дерево решений, но хуже, чем линейные модели.

	R2	MAE	max_error
LinearRegression	-0.018532	-2.502366	-8.098392
Ridge	-0.018463	-2.502325	-8.097452
Lasso	-0.019376	-2.510495	-7.798105
SVR	-0.041456	-2.499637	-8.357012
KNeighborsRegressor	-0.238674	-2.725185	-8.823389
DecisionTreeRegressor	-1.034156	-3.589790	-11.822403
RandomForestRegressor	-0.075323	-2.555245	-8.380008
GradientBoostingRegressor	-0.098848	-2.593079	-8.632489

Рисунок 17 — Результаты моделей с гиперпараметрами по умолчанию

	R2	MAE	max_error
Ridge(alpha=80, positive=True, solver='lbfgs')	-0.016585	-2.494485	-7.850363
Lasso(alpha=0.05)	-0.012094	-2.500839	-7.965382
SVR(C=0.01, kernel='linear')	-0.017814	-2.500515	-8.061850
KNeighborsRegressor(n_neighbors=29)	-0.036593	-2.512539	-8.157406
DecisionTreeRegressor(max_depth=2, max_features=2, random_state=42)	-0.018267	-2.490189	-8.154902
RandomForestRegressor(bootstrap=False, criterion='absolute_error', max_depth=5, max_features=1, n_estimators=50, random_state=42)	-0.017821	-2.497013	-8.146335

Рисунок 18 — Результаты моделей после подбора гиперпараметров

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

Можно сделать 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 дает коэффициент детерминации близкий к нулю, 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 превзошли результат линейной модели. Но они не являются объясняющей зависимостью моделью.

Собирая деревья в ансамбли, можно улучшать характеристики. Но подбор параметров для леса затруднен тем, что это затратный по времени процесс. По этой причине мне не удалось получить комбинацию параметров для леса, которая была бы лучше деревьев решений.

Поэтому в качестве лучшей модели выбираю дерево решений. На рисунке 19 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.

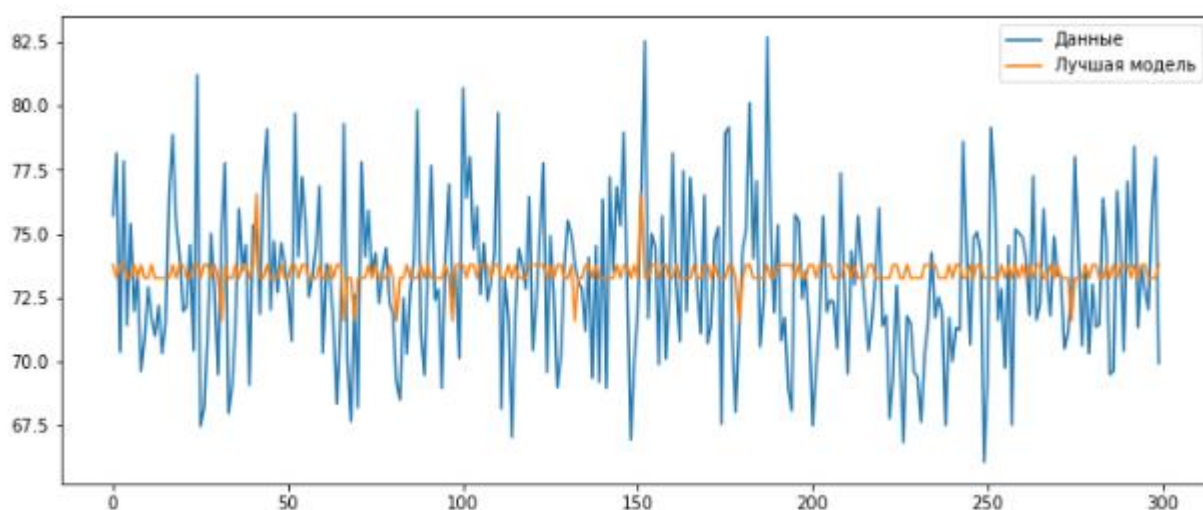


Рисунок 19 — Визуализация работы модели

	R2	MAE	max_error
Лучшая модель (дерево решений)	-0.035776	-2.610243	-8.152045

Рисунок 20 - Метрики работы лучшей модели на тестовом множестве

Метрики работы лучшей модели на тестовом множестве и сравнение с базовой отражены на рисунке 20. Они подтверждают: полученная модель хуже базовой. Результат исследования отрицательный. Не удалось получить модели, которая могла бы оказать помощь в принятии решений специалисту предметной области.

2.3 Для прогнозирования прочности при растяжении

Для подбора лучшей модели для этой задачи были взяты следующие модели:

- LinearRegression — линейная регрессия (раздел 1.2.1);
- Ridge — гребневая регрессия (раздел 1.2.2);
- Lasso — лассо-регрессия (раздел 1.2.2);
- SVR — метод опорных векторов (раздел 1.2.3);
- KNeighborsRegressor — метод ближайших соседей (раздел 1.2.4);
- DecisionTreeRegressor — деревья решений (раздел 1.2.5);
- RandomForestRegressor — случайный лес (раздел 1.2.6)
- GradientBoostingRegressor – градиентный бустинг (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 21.

	R2	MAE	max_error
LinearRegression	-0.014804	-391.262712	-1305.947015
Ridge	-0.014749	-391.248131	-1305.883895
Lasso	-0.013580	-390.926249	-1304.848543
SVR	-0.021077	-390.543237	-1279.655107
KNeighborsRegressor	-0.199452	-422.715060	-1379.897274
DecisionTreeRegressor	-1.097452	-561.980002	-1784.349498
RandomForestRegressor	-0.036247	-393.990948	-1325.856363
GradientBoostingRegressor	-0.033926	-398.082126	-1274.138037

Рисунок 21 — Результаты моделей с гиперпараметрами по умолчанию

Ни одна из выбранных мной моделей не соответствует данным.

R2 близок к 0 для линейных моделей и метода опорных векторов. 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.

Градиентный бустинг с параметрами по умолчанию отработал лучше дерева. Он тоже соответствует базовой модели.

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

	R2	MAE	max_error
Ridge(alpha=710, solver='lsqr')	-0.010113	-389.284538	-1291.069649
Lasso(alpha=20)	-0.010149	-389.214674	-1293.747766
SVR(C=0.02, kernel='linear')	-0.021227	-390.526504	-1279.607766
DecisionTreeRegressor(criterion='poisson', max_depth=3, max_features=10, random_state=42, splitter='random')	-0.014831	-388.216719	-1282.593572
GradientBoostingRegressor(max_depth=2, max_features=1, random_state=42)	0.009853	-386.455905	-1276.277562

Рисунок 22 — Результаты моделей после подбора гиперпараметров

Подбор гиперпараметров - интересный процесс. Но нам он не помог получить модель, превосходящую базовую. Все модели крайне плохо описывают исходные данные. Не удалось добиться коэффициента детерминации, большего нуля.

Линейные после подбора немного улучшили характеристики.

Метод опорных векторов отработал аналогично линейным моделям.

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

Но лучший результат дает градиентный бустинг. Значения ошибок примерно такие же, как у дерева решений. Но коэффициент детерминации немного больше, что показывает чуть лучшую объясняющую способность модели.

Поэтому в качестве лучшей модели выбираю градиентный бустинг. На рисунке 23 приведена визуализация работы лучшей модели на тестовом множестве.

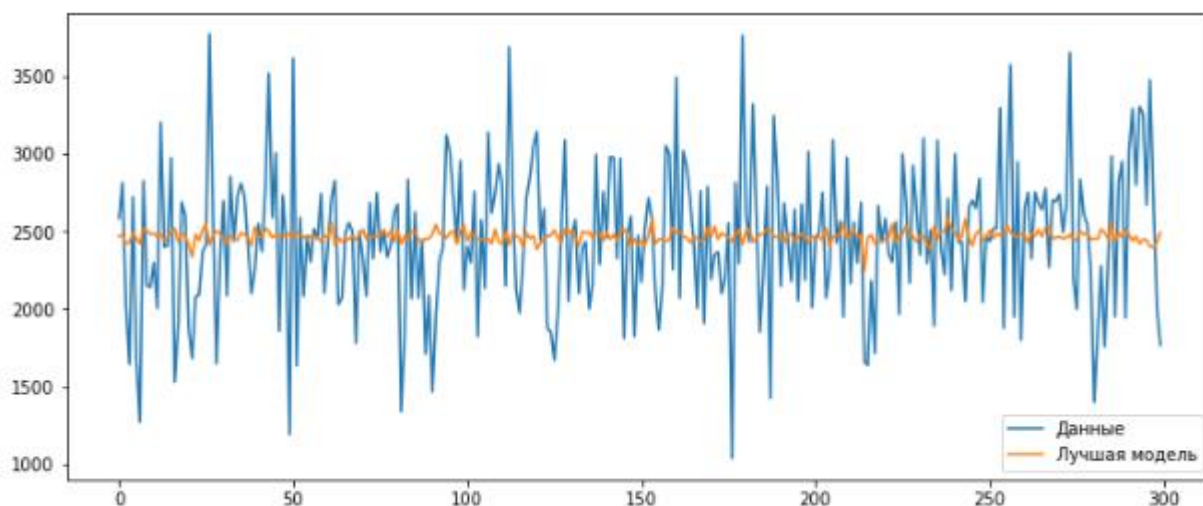


Рисунок 23 — Визуализация работы модели

Визуализируя результаты градиентного бустинга с выбранными параметрами, мы видим насколько они плохи и далеки от исходных данных. Но

результаты выглядят более "естественно", чем те, что получены деревом решений для модуля упругости при растяжении.

Метрики работы лучшей модели на тестовом множестве и сравнение с базовой отражены на рисунке 24. Несмотря на то, что градиентный бустинг показывает результаты чуть-чуть лучше базовой, результат исследования отрицательный. Не удалось получить модели, которая могла бы оказать помощь в принятии решений специалисту предметной области.

	R2	MAE	max_error
Лучшая модель (градиентный бустинг)	0.004028	-376.647056	-1384.841404

Рисунок 24 - 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 `sklearn`

Строю нейронную сеть с помощью класса `MLPRegressor` следующей архитектуры:

- слоев: 8;
- нейронов на каждом слое: 24;
- активационная функция: `relu`;
- оптимизатор: `adam`;
- пропорция разбиения данных на тестовые и валидационные: 30%;
- ранняя остановка, если метрики на валидационной выборке не улучшаются;

- количество итераций: 5000.

Нейросеть обучилась за 343мс и 33 итерации. График обучения приведен на рисунке 25.

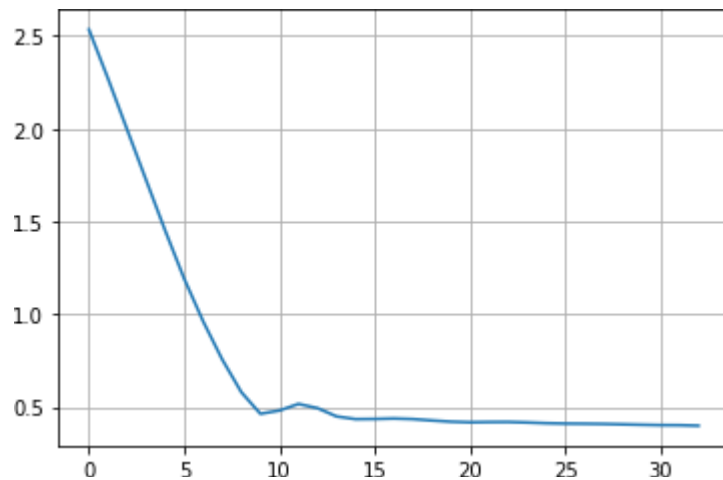


Рисунок 25 — График обучения MLPRegressor

Визуализация результатов, полученных нейросетью, приведены на рисунке 26. Видно, что нейросеть пыталась подстроиться под исходные данные, но хорошо не получилось.

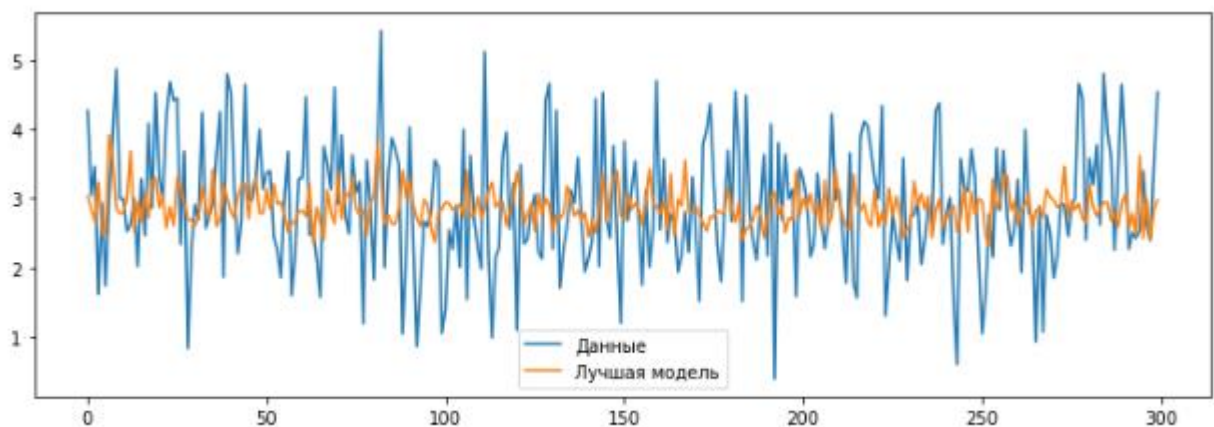


Рисунок 26 — Визуализация работы модели

Метрики работы нейросети MLPRegressor на тестовом множестве и сравнение с базовой моделью отражены на рисунке 27. Несмотря на красивый график с рисунка 26, метрики говорят об отсутствии результата, который можно

внедрить. Ошибка нейросети составляет 30,7%, а ее значения ошибок хуже, чем у базовой модели.

	R2	MAE	max_error
MLPRegressor	-0.052842	-0.751262	-2.790557

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 рисунках 28 и 29.

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		
Trainable params: 4,537		
Non-trainable params: 0		

Рисунок 28 — Архитектура нейросети в виде summary

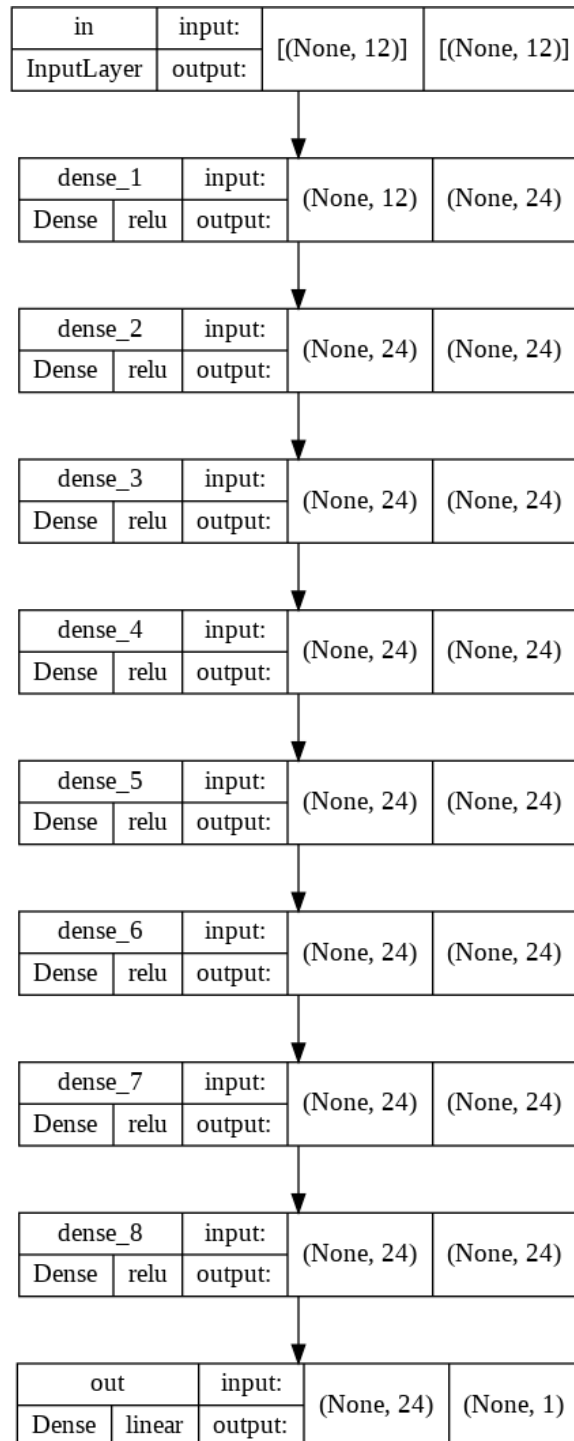


Рисунок 29 — Архитектура нейросети в виде графа

Запускаю обучение нейросети со следующими параметрами:

- пропорция разбиения данных на тестовые и валидационные: 30%;
- количество эпох: 50.

- раннюю остановку не использую.

График обучения приведен на рисунке 30, ошибка — в таблице 2.

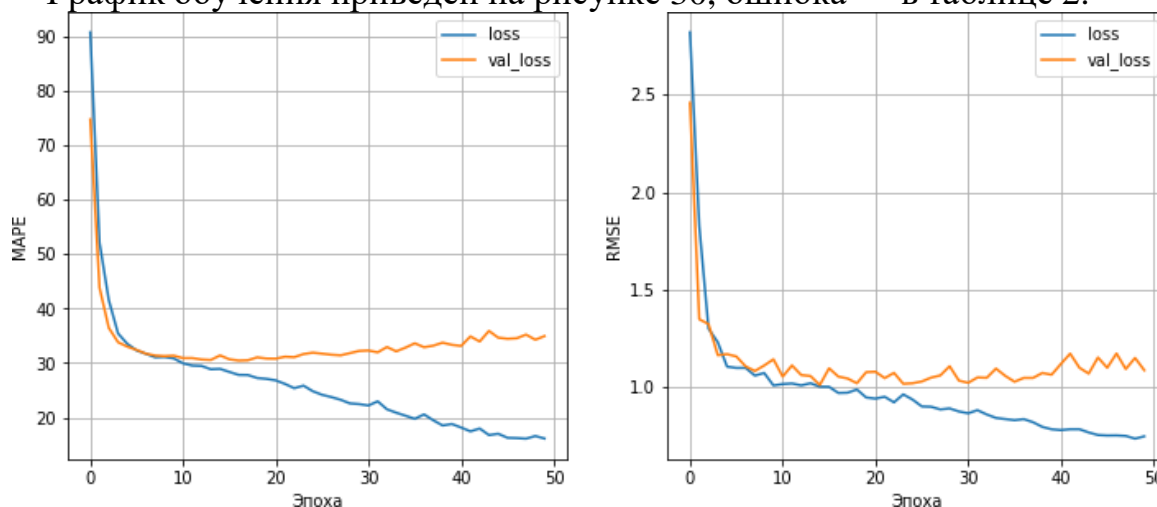


Рисунок 30 — График обучения нейросети

Видно, что примерно до 8 эпохи обучение шло хорошо, а потом сеть начала переобучаться. Значение loss на тестовых выборках продолжило уменьшаться, а на валидационной начало расти.

Одним из способов борьбы с переобучением может быть ранняя остановка обучения, если val_loss начинает расти. Для этого в tensorflow используются callbacks. Попробую взять нейросеть с той же архитектурой и запустить обучение с ранней остановкой. График обучения приведен на рисунке 31, а ошибка — в таблице 2. 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. Такой слой выключает 5% случайных нейронов на каждом слое.

График обучения приведен на рисунке 32, 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 останови-

лись слишком рано.

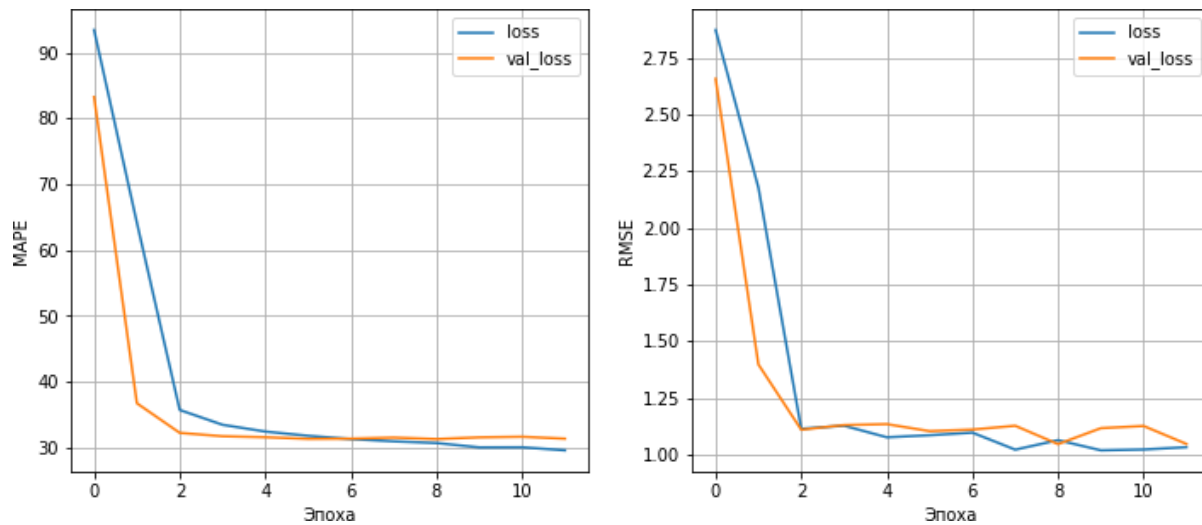


Рисунок 31 — График обучения нейросети с ранней остановкой

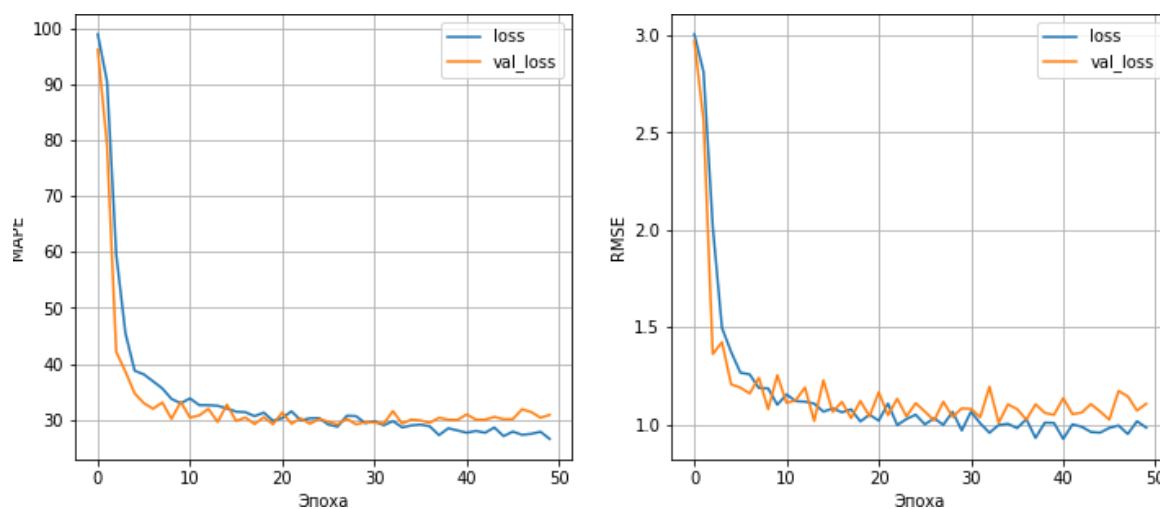


Рисунок 32 — График обучения нейросети с Dropout-слоем

Таблица 2. Борьба с переобучением нейросети

	Эпох	Ошибка на тестовых данных, %	Время обучения, с
Нейросеть переобученная	50	37.37	5.85
Нейросеть с ранней остановкой	12	31.28	2.25
Нейросеть с dropout-слоями	50	32.41	12.4

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

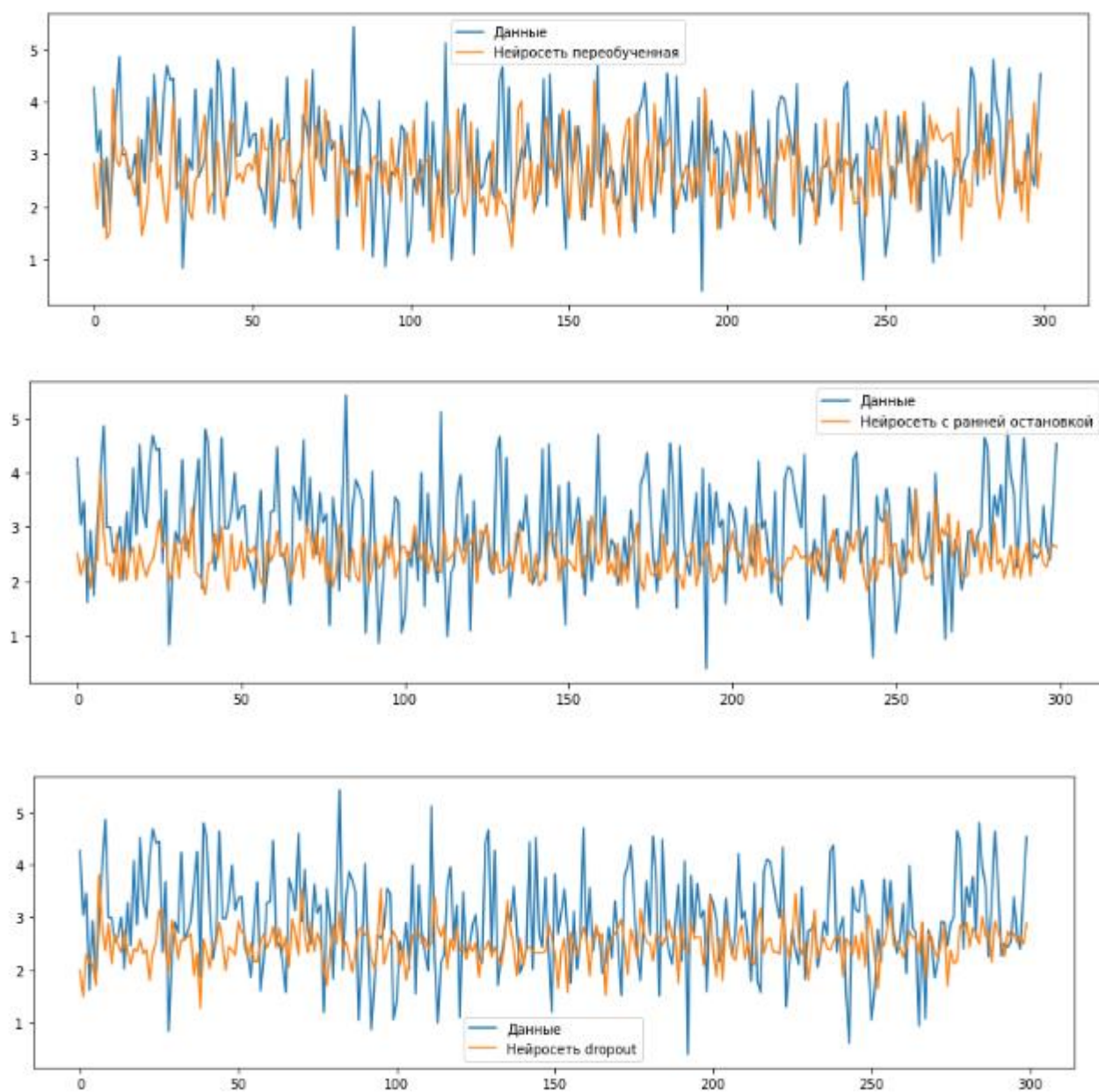


Рисунок 33 - Визуализация результатов работы нейросетей

	R2	MAE	max_error
Нейросеть переобученная	-0.624376	-0.938195	-2.868809
Нейросеть с ранней остановкой	-0.322407	-0.852214	-2.781806
Нейросеть dropout	-0.628132	-0.960385	-2.903841

Рисунок 34 -Метрики работы нейросетей на тестовом множестве

Визуализация результатов показывает, что нейросеть из библиотеки tensorflow старалась подстроиться к данным. Выглядят результаты «похоже», но метрики разочаровывают. 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

Согласно заданию, необходимо сравнить ошибку каждой модели на тренировочной и тестирующей части выборки.

Модель для предсказания модуля упругости при растяжении - DecisionTreeRegressor(criterion='absolute_error', max_depth=2, max_features=10, splitter='random'). Сравнение ее ошибок показано на рисунке 35.

	R2	MAE	max_error
Модуль упругости, тренировочный	0.017295	-2.410294	-9.008468
Модуль упругости, тестовый	-0.035776	-2.610243	-8.152045

Рисунок 35 - Сравнение ошибок модели для модуля упругости при растяжении на тренировочном и тестовом датасете

Дерево решений имеет ошибку на тренировочном датасете меньше, чем на тестовом, потому что чему-то все-таки научилось. Но даже на тренировоч-

ном датасете оно не нашло закономерности во входных данных. Задачу решить не удалось.

Если модуль упругости при растяжении лежит в диапазоне [64.05-82.68], то наша модель делает предсказание с точностью ± 8.15 . Она работает не точнее среднего, и бесполезна для применения в реальных условиях.

Модель для предсказания прочности при растяжении - GradientBoostingRegressor(max_depth=1, max_features=1, n_estimators=50). Сравнение ее ошибок показано на рисунке 36.

	R2	MAE	max_error
Прочность при растяжении, тренировочный	0.057141	-374.670333	-1383.885510
Прочность при растяжении, тестовый	0.004028	-376.647056	-1384.841404

Рисунок 36 - Сравнение ошибок модели для прочности на тренировочном и тестовом датасете

Градиентный бустинг - это прекрасный метод, который показал положительный, хоть и близкий к 0 коэффициент детерминации. Ошибка на тестовом множестве незначительно больше, чем на тренировочном. Значит, модель нашла следы зависимости, а не выучила данные. Но задача не решена.

Если прочность при растяжении лежит в диапазоне [1071.12-3848.44], то наша модель дает предсказание с точностью ± 1384.85 . 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 37.

	R2	MAE	max_error
Соотношение матрица-наполнитель, тренировочный	-0.212722	-0.787676	-3.084322
Соотношение матрица-наполнитель, тестовый	-0.322407	-0.852214	-2.781806

Рисунок 37 - 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 [0.39-5.46], то наша модель может предсказать с точностью ± 3.08 . Она работает не точнее среднего, и бесполезна для применения в реальных условиях.

2.6. Разработка приложения

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

В приложении необходимо реализовать следующие функции:

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

Решено разработать веб-приложение с помощью языка Python, фреймворка Flask и шаблонизатора Jinja.

Эту задачу получилось решить. 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/neurologist-engineer/VKR>. На него были загружены результаты работы: исследовательский notebook, код приложения.

Заключение

В ходе выполнения данной работы мы прошли практически весь Dataflow pipeline, рассмотрели большую часть операций и задач, которые приходится выполнять специалисту по работе с данными.

Этот поток операций и задач включает:

- изучение теоретических методов анализа данных и машинного обучения;
- изучение основ предметной области, в которой решается задача;
- извлечение и трансформацию данных. Здесь нам был предоставлен гото-

вый набор данных, поэтому через трудности работы с разными источниками и парсингом данных мы еще не соприкоснулись;

- проведение разведочного анализа данных статистическими методами;
- DataMining — извлечение признаков из датасета и их анализ;
- разделение имеющихся, в нашем случае размеченных, данных на обучающую, валидационную, тестовую выборки;
- выполнение предобработки (препроцессинга) данных для обеспечения корректной работы моделей;
- построение аналитического решения. Это включает выбор алгоритма решения и модели, сравнение различных моделей, подбор гиперпараметров модели;
- визуализация модели и оценка качества аналитического решения;
- сохранение моделей;
- разработка и тестирование приложения для поддержки принятия решений специалистом предметной области, которое использовало бы найденную модель;
- внедрение решения и приложения в эксплуатацию. Этот блок задач мы тоже пока не затронули.

В этой работе мы имели дело не с учебными наборами данных, которые дают хорошо изученные решения, а с реальной производственной задачей. И к сожалению, не смогли поставленную задачу решить — не получили моделей, которые бы описывали закономерности предметной области.

Возможные причины неудачи:

- нечеткая постановка задачи, отсутствие дополнительной информации о зависимости признаков с точки зрения физики процесса. Незначимые признаки являются для модели шумом, и мешают найти зависимость целевых от значимых входных признаков;
- исследование предварительно обработанных данных. Возможно, на "сырых", не предобработанных данных можно было бы получить более качественные модели, воспользовавшись другими методами очистки и подготовки;

- мой недостаток знаний и опыта. Нейросети являются самым современным подходом к решению такого рода задач. Они способны находить скрытые и нелинейные зависимости в данных. Но выбор оптимальной архитектуры нейросети является неочевидной задачей.

Дальнейшие возможные пути решения этой задачи могли бы быть:

- углубиться в изучение нейросетей, попробовать различные архитектуры, параметры обучения и т.д.;

- провести отбор признаков разными методами. Испробовать методы уменьшения размерности, например метод главных компонент;

- после уменьшения размерности градиентный бустинг может улучшить свои результаты. Так же есть большой простор для подбора гиперпараметров для этого метода;

- проконсультироваться у экспертов в предметной области. Возможно, они могли бы поделиться знаниями, необходимыми для решения задачи.

Библиографический список

- 1 Композиционные материалы : учебное пособие для вузов / Д. А. Иванов, А. И. Ситников, С. Д. Шляпин ; под редакцией А. А. Ильина. — Москва : Издательство Юрайт, 2019 — 253 с. — (Высшее образование). — Текст : непосредственный.
- 2 Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.
- 3 ГрасД. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.
- 4 Документация по 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>.

Приложение А. Скриншоты веб-приложения

Скриншоты веб-приложения, иллюстрирующие его работу, приведены на рисунках 38 - 44. Реализованы были следующие функции:

- выбор целевой переменной для предсказания (модуль упругости при растяжении и прочности при растяжении или соотношение матрица-наполнитель);
- ввод входных параметров;
- проверка введенных параметров;
- 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 38 — Начальное окно, выбор целевых переменных

Прогнозирование модуля упругости при растяжении и прочности при растяжении

Соотношение матрица-наполнитель (0..6)

Плотность, кг/м3 (1700...2300)

Модуль упругости, ГПа (2...2000)

Количество отвердителя, м.% (17...200)

Содержание эпоксидных групп, % 2 (14...34)

Температура вспышки, С_2 (100...414)

Поверхностная плотность, г/м2 (0.6...1400)

Потребление смолы, г/м2 (33...414)

Угол нашивки, град (0 или 90)

Шаг нашивки (0...15)

Плотность нашивки (0...104)

Рисунок 39 - Ввод входных параметров для прогнозирования модуля упругости при растяжении и прочности при растяжении

Прогнозирование модуля упругости при растяжении и прочности при растяжении

Содержание эпоксидных групп, % 2 - некорректное значение "a"
Количество отвердителя, м.% - значение вне корректного диапазона
Некоторые значения отсутствуют!

Соотношение матрица-наполнитель (0..6) | 3

Плотность, кг/м3 (1700...2300) | 1800

Модуль упругости, ГПа (2...2000) | 900

Количество отвердителя, м.% (17...200) | 0

Содержание эпоксидных групп, % 2 (14...34) | a

Температура вспышки, С_2 (100...414) |

Поверхностная плотность, г/м2 (0.6...1400) |

Потребление смолы, г/м2 (33...414) |

Угол нашивки, град (0 или 90) |

Шаг нашивки (0...15) |

Плотность нашивки (0...104) |

Рисунок 40 - Проверка входных параметров для прогнозирования модуля упругости при растяжении и прочности при растяжении

ВКР x + 127.0.0.1:5000/model_1_2/

Прогнозирование модуля упругости при растяжении и прочности при растяжении

Соотношение матрица-наполнитель (0...0)

Плотность, кг/м3 (1700...2300)

Модуль упругости, ГПа (2...2000)

Количество отвердителя, м.% (17...200)

Содержание эпоксидных групп, % 2 (14...34)

Температура вспышки, C_2 (100...414)

Поверхностная плотность, г/м2 (0.6...1400)

Потребление смолы, г/м2 (33...414)

Угол нашивки, град (0 или 90)

Шаг нашивки (0...15)

Плотность нашивки (0...104)

Входные переменные:

Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп, % 2	Температура вспышки, C_2	Поверхностная плотность, г/м2	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
4.029126	1880.0	622.0	111.86	22.267857	284.615385	470.0	220.0	90.0	4.0	60.0

Результат модели:

Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа
72.81891497929365	2523.9223070281537

Рисунок 41 - Результат работы модели для модуля упругости при растяжении и прочности при растяжении

ВКР x + 127.0.0.1:5000/model_3/

Прогнозирование соотношения матрица-наполнитель

Плотность, кг/м3 (1700...2300)

Модуль упругости, ГПа (2...2000)

Количество отвердителя, м.% (17...200)

Содержание эпоксидных групп, % 2 (14...34)

Температура вспышки, C_2 (100...414)

Поверхностная плотность, г/м2 (0.6...1400)

Модуль упругости при растяжении, ГПа (64...83)

Прочность при растяжении, МПа (1036...3849)

Потребление смолы, г/м2 (33...414)

Угол нашивки, град (0 или 90)

Шаг нашивки (0...15)

Плотность нашивки (0...104)

Рисунок 42 - Ввод входных параметров для прогнозирования соотношения матрица-наполнитель

ВКР x + 127.0.0.1:5000/model_3/

Прогнозирование соотношения матрица-наполнитель

Поверхностная плотность, г/м2 - некорректное значение "50066"
 Температура вспышки, С_2 - значение вне корректного диапазона
 Некоторые значения отсутствуют!

Плотность, кг/м3 (1700...2300) 2100

Модуль упругости, ГПа (2...2000) 1200

Количество отвердителя, м.% (17...200) 53

Содержание эпоксидных групп, %_2 (14...34) 28

Температура вспышки, С_2 (100...414) 0

Поверхностная плотность, г/м2 (0.6...1400) 50066

Модуль упругости при растяжении, ГПа (64...83)

Прочность при растяжении, МПа (1036...3849)

Потребление смолы, г/м2 (33...414)

Угол нашивки, град (0 или 90)

Шаг нашивки (0...15)

Плотность нашивки (0...104)

Отправить

Рисунок 43 - Проверка входных параметров для прогнозирования соотношения матрица-наполнитель

ВКР x + 127.0.0.1:5000/model_3/

Прогнозирование соотношения матрица-наполнитель

Плотность, кг/м3 (1700...2300) 1880.0

Модуль упругости, ГПа (2...2000) 622.0

Количество отвердителя, м.% (17...200) 111.86

Содержание эпоксидных групп, %_2 (14...34) 22.2678571428571

Температура вспышки, С_2 (100...414) 284.615384615384

Поверхностная плотность, г/м2 (0.6...1400) 470.0

Модуль упругости при растяжении, ГПа (64...83) 73.3333333333333

Прочность при растяжении, МПа (1036...3849) 2455.55555555555

Потребление смолы, г/м2 (33...414) 220.0

Угол нашивки, град (0 или 90) 90.0

Шаг нашивки (0...15) 4.0

Плотность нашивки (0...104) 60.0

Отправить

Входные переменные:

	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп, %_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	1880.0	622.0	111.86	22.267857	284.615385	470.0	73.333333	2455.555556	220.0	90.0	4.0	60.0

Результат модели:

Соотношение матрица-наполнитель
2.5154960585858928

Рисунок 44 - Результат работы модели для соотношения матрица-наполнитель

Так же было выполнено тестирование: модели, запущенные в тетрадке jupyter, где разрабатывалось аналитическое решение, и модели из приложения на одних и тех же данных возвращают одинаковый результат. Значит, загрузка моделей и подготовка параметров для моделей выполнены верно.