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 Четвериков Артем Васильевич

Москва, 2022

Содержание

[Введение 3](#__RefHeading___Toc6566_1566460903)

[1 Аналитическая часть 4](#__RefHeading___Toc6568_1566460903)

[1.1 Постановка задачи. 4](#__RefHeading___Toc6570_1566460903)

[1.2 Описание используемых методов 6](#__RefHeading___Toc6572_1566460903)

[1.3 Разведочный анализ данных 15](#__RefHeading___Toc6574_1566460903)

[2 Практическая часть 23](#__RefHeading___Toc6576_1566460903)

[2.1 Предобработка данных 23](#__RefHeading___Toc6578_1566460903)

[2.2 Разработка и обучение модели 35](#__RefHeading___Toc6580_1566460903)

[2.2.1 Предсказание прочности при растяжении 36](#__RefHeading___Toc19540_3616107706)

[2.2.2 Предсказание модуля упругости при растяжении 38](#__RefHeading___Toc19542_3616107706)

[2.3 Тестирование модели 41](#__RefHeading___Toc6582_1566460903)

[2.3.1 Предсказание прочности при растяжении 41](#__RefHeading___Toc19540_36161077061)

[2.3.2 Предсказание модуля упругости при растяжении 46](#__RefHeading___Toc19542_36161077061)

[2.4 Написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель 52](#__RefHeading___Toc3786_2159239638)

[2.5 Разработка приложения 56](#__RefHeading___Toc6584_1566460903)

[2.6 Создание удаленного репозитория и загрузка результатов работы 60](#__RefHeading___Toc6586_1566460903)

[Заключение 61](#__RefHeading___Toc6588_1566460903)

[Библиографический список 62](#__RefHeading___Toc6590_1566460903)

[Приложение 1 63](#__RefHeading___Toc6592_1566460903)

[Приложение 2 66](#__RefHeading___Toc6594_1566460903)

# Введение

# 1 Аналитическая часть

## 1.1 Постановка задачи.

Тема выпускной квалификационной работы − прогнозирование конечных свойств новых материалов (композиционных материалов).

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

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

Актуальность: созданные прогнозные модели помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов возможными новыми характеристиками материалов, и цифровыми двойниками новых композитов.

## 1.2 Описание используемых методов

Рассмотрим некоторые наиболее распространенные методы машинного обучения для решения задач регрессии.

Метод линейной регрессии (LinearRegression)

Линейная регрессия подгоняет линейную модель с коэффицентами w=(w1,...,wp ) к минимизации остаточной суммы квадрата между наблюдаемым целевоым признаком в наборе данных и предсказанным целевым признаком по линейной аппроксимации.

Преимущества:

- быстр и прост в реализации; легко интерпретируем;

- имеет меньшую сложность по сравнению с другими алгоритмами;

Недостатки:

- моделирует только прямые линейные зависимости;

- требует прямую связь между зависимыми и независимыми переменными;

- выбросы оказывают огромное влияние, а границы линейны.

Метод опорных векторов (SVR)

Метод опорных векторов (англ. support vector machine, SVM) — один из наиболее популярных методов обучения, который применяется для решения задач классификации и регрессии. Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора

Регрессия опорных векторов − это контролируемая модель обучения, которую можно использовать для выполнения как линейной, так и нелинейной регрессии. Цель применения линейной регрессии − минимизировать ошибку между прогнозом и данными. Однако цель применения регрессии опорных векторов к набору данных − убедиться, что ошибки не превышают пороговое значение. В SVR мы помещаем как можно больше экземпляров между строками, ограничивая при этом нарушение полей.

Сложность подбора времени более чем квадратична с количеством выборок, что затрудняет масштабирование до наборов данных, содержащих более пары 10000 выборок.

Преимущества SVM перед методом стохастического градиента и нейронными сетями:

- задача выпуклого квадратичного программирования хорошо изучена и имеет единственное решение;

- метод опорных векторов эквивалентен двухслойной нейронной сети, где число нейронов на скрытом слое определяется автоматически как число опорных векторов;

- принцип оптимальной разделяющей гиперплоскости приводит к максимизации ширины разделяющей полосы, а следовательно, к более уверенной классификации.

Недостатки классического SVM:

- неустойчивость к шуму: выбросы в исходных данных становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости;

- не описаны общие методы построения ядер и спрямляющих пространств, наиболее подходящих для конкретной задачи;

- нет отбора признаков;

- необходимо подбирать константу C при помощи кросс-валидации.

Метод К ближайших соседей (KneighborsRegressor)

Регрессия на основе соседей может использоваться в тех случаях, когда метки данных являются непрерывными, а не дискретными переменными. Метка, присвоенная точке запроса, вычисляется на основе среднего значения меток ее ближайших соседей. Базовая регрессия ближайших соседей использует одинаковые веса: то есть каждая точка в локальной окрестности вносит одинаковый вклад в классификацию точки запроса. При некоторых обстоятельствах может быть выгодно взвешивать точки таким образом, чтобы близлежащие точки вносили больший вклад в регрессию, чем удаленные точки.

Несмотря на свою простоту, метод К ближайших соседей успешно решает большое количество задач классификации и регрессии, включая рукописные цифры и сцены спутниковых изображений. Будучи непараметрическим методом, он часто оказывается успешным в ситуациях классификации, когда граница принятия решения очень нерегулярна.

Преимущества:

- алгоритм прост и легко реализуем;

- не чувствителен к выбросам;

- нет необходимости строить модель, настраивать несколько параметров или делать дополнительные допущения;

- алгоритм универсален. Его можно использовать для обоих типов задач: классификации и регрессии;

Недостатки:

- алгоритм работает значительно медленнее при увеличении объема выборки, предикторов или независимых переменных;

- из аргумента выше следуют большие вычислительные затраты во время выполнения;

- всегда нужно определять оптимальное значение k.

Метод решающих деревьев (DecisionTreeRegressor)

Деревья решений − это непараметрический метод обучения с контролем, используемый для классификации и регрессии. Цель состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной путем изучения простых правил принятия решений, выведенных из объектов данных. Дерево можно рассматривать как кусочно-постоянное приближение.

Некоторые преимущества метода деревьев решений:

- простой для понимания и интерпретации. Деревья можно визуализировать;

- требуется небольшая подготовка данных. Другие методы часто требуют нормализации данных, создания фиктивных переменных и удаления пустых значений. Однако, этот модуль не поддерживает пропущенные значения;

- стоимость использования дерева (т.е. прогнозирования данных) логарифмически зависит от количества точек данных, используемых для обучения дерева;

- способен обрабатывать как числовые, так и категориальные данные. Однако реализация scikit-learn пока не поддерживает категориальные переменные. Другие методы обычно специализируются на анализе наборов данных, которые имеют только один тип переменной;

- способен обрабатывать проблемы с несколькими выводами;

- использует модель белого ящика. Если данная ситуация наблюдаема в модели, объяснение условия легко объясняется булевой логикой. В отличие от этого, в модели черного ящика (например, в искусственной нейронной сети) результаты могут быть более сложными для интерпретации;

- возможно проверить модель с помощью статистических тестов. Это позволяет учитывать надежность модели;

- работает хорошо, даже если его предположения несколько нарушаются истинной моделью, из которой были сгенерированы данные.

К недостаткам деревьев решений относятся:

- могут создаваться слишком сложные деревья, которые плохо обобщают данные. Это называется перенасыщением. Чтобы избежать этой проблемы, необходимы такие механизмы, как обрезка, установка минимального количества выборок, требуемых для конечного узла, или установка максимальной глубины дерева;

- деревья решений могут быть нестабильными, поскольку небольшие отклонения в данных могут привести к созданию совершенно другого дерева. Эта проблема устраняется путем использования деревьев решений в ансамбле;

- предсказания деревьев решений не являются ни гладкими, ни непрерывными, а кусочно-постоянными приближениями. Поэтому они не очень хороши для экстраполяции;

- известно, что проблема обучения оптимального дерева решений является NP-полной при нескольких аспектах оптимальности и даже для простых концепций. Следовательно, практические алгоритмы обучения дерева решений основаны на эвристических алгоритмах, таких как жадный алгоритм, где локально оптимальные решения принимаются на каждом узле. Такие алгоритмы не могут гарантировать возврат глобально оптимального дерева решений. Это может быть смягчено путем обучения нескольких деревьев в обучаемом ансамбле, где функции и образцы выбираются случайным образом с заменой;

- есть концепции, которые трудно изучить, потому что деревья решений не выражают их легко, например, XOR, проблемы с четностью или мультиплексором;

- могут создаваться предвзятые деревья, если некоторые классы доминируют. Поэтому рекомендуется сбалансировать набор данных перед подгонкой к дереву решений.

Метод стахостического градиентного спуска (SGDRegressor)

Стохастический градиентный спуск (SGD) это простой, но очень эффективный подход к подгонке линейных классификаторов и регрессоров под выпуклые функции потерь. Несмотря на то, что SGD существует в сообществе машинного обучения уже давно, в последнее время ему уделяется значительное внимание в контексте крупномасштабного обучения.

Стохастический градиентный спуск представляет собой итерационный метод оптимизации целевой функции с подходящими свойствами гладкости (например, дифференцируемыми или поддифференцируемыми). Его можно рассматривать как стохастическое приближение оптимизации градиентного спуска, поскольку он заменяет фактический градиент (рассчитанный на основе всего набора данных) его оценкой (рассчитанной на основе случайно выбранного подмножества данных). Особенно в задачах многомерной оптимизации это снижает очень высокую вычислительную нагрузку, обеспечивая более быстрые итерации в обмен на более низкую скорость сходимости.

SGD успешно применяется для решения крупномасштабных и разреженных задач машинного обучения, часто встречающихся при классификации текста и обработке естественного языка. Учитывая, что данных мало, классификаторы в этом модуле легко масштабируются для решения задач с более чем 10 ^ 5 обучающими примерами и более чем 10 ^ 5 функциями.

Строго говоря, SGD - это просто метод оптимизации и не соответствует определенному семейству моделей машинного обучения. Это всего лишь способ для обучения модели.

Многослойный перцептрон (MLPRegressor)

Это искусственная нейронная сеть, имеющая 3 или более слоёв персептронов. Эти слои - один входной слой, 1 или более скрытых слоёв и один выходной слой персептронов.

Эта модель оптимизирует квадрат ошибки с использованием LBFGS или стохастического градиентного спуска

MLPRegressor обучается итеративно, поскольку на каждом временном шаге вычисляются частные производные функции потерь по параметрам модели для обновления параметров.

К функции потерь также может быть добавлен термин регуляризации, который уменьшает параметры модели для предотвращения переобучения.

Эта реализация работает с данными, представленными в виде плотных и разреженных числовых массивов значений с плавающей запятой.

Преимущества многослойного перцептрона:

- возможность изучать нелинейные модели;

- возможность изучения моделей в режиме реального времени (онлайн-обучение) с использованием partial\_fit.

К недостаткам многослойного персептрона (MLP) можно отнести:

- MLP со скрытыми слоями имеют невыпуклую функцию потерь, когда существует более одного локального минимума. Поэтому разные инициализации случайных весов могут привести к разной точности проверки;

- MLP требует настройки ряда гиперпараметров, таких как количество скрытых нейронов, слоев и итераций;

- MLP чувствителен к масштабированию функций.

Метод Лассо регрессии (Lasso)

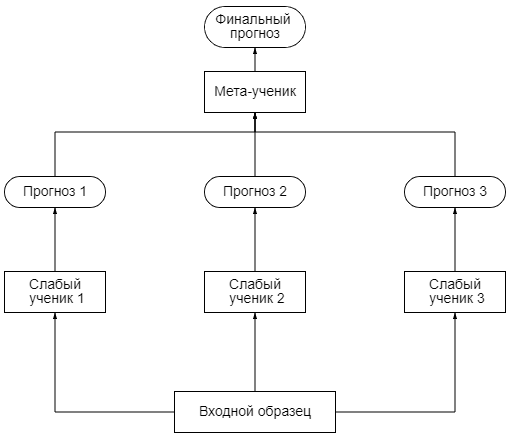
Лассо — это линейная модель, которая оценивает разреженные коэффициенты. Это полезно в некоторых контекстах из-за своей тенденции отдавать предпочтение решениям с меньшим количеством ненулевых коэффициентов, эффективно уменьшая количество функций, от которых зависит данное решение. По этой причине лассо и его варианты являются фундаментальными для области сжатого зондирования. При определенных условиях он может восстановить точный набор ненулевых коэффициентов

Ансамблевые методы

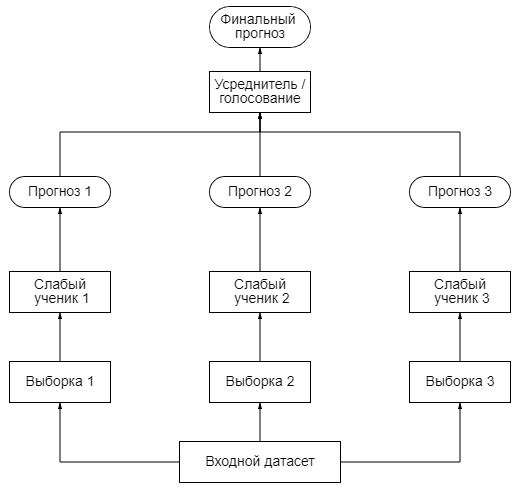
Для повышения точности предсказания применяются ансамблевые методы метод случайного леса (Random Forest Regressor), метод градиентного бустинга (GradientBoostingRegressor), стекинг (StackingRegressor) и др.

Наиболее популярными ансамблевыми методами являются: стекинг, бэггинг, бустинг.

Стекинг: используется несколько разнородных слабых учеников. Их обучают и объединяют для построения прогноза, основанного на результатах различных слабых моделей (рисунок 1).

Рисунок 1 − Схема обучения моделей методом стекинга

Бэггинг: в этом случае однородные модели обучают на разных наборах данных и объединяют (рисунок 2). Получают прогноз путём усреднения. Если использовать в качестве слабого ученика деревья решений, то получится случайный лес .

Рисунок 2 − Схема обучения моделей методом бэггинга

Бустинг: при использовании данного метода несколько однородных моделей последовательно обучаются, исправляя ошибки друг друга.

Метод бустинга в чём то схож с методом бэггинга: берётся множество одинаковых моделей и объединяется, чтобы получить сильного ученика. Но разница заключается в том, что модели приспосабливаются к данным последовательно, то есть каждая модель будет исправлять ошибки предыдущей.

Базовые модели для бустинга - это модели с низким разбросом и высоким смещением. Например неглубокие деревья решений. Одна из причин такого выбора моделей - они требуют меньше вычислительных затрат. Ещё бустинг (в отличии от бэггинга) нельзя распараллелить.

Существует два наиболее распространённых алгоритма бустинга - адаптивный бустинг и градиентный бустинг.

## 1.3 Разведочный анализ данных

Разведочный анализ данных − один из первых и определяющих шагов в обработке данных.Разведочный анализ данных означает изучение данных для получения из них практической информации. Он включает в себя анализ и обобщение массивных наборов данных, часто в форме диаграмм и графиков.

Цель разведочного анализа − получение первоначальных представлений о характерах распределений переменных исходного набора данных, формирование оценки качества исходных данных (наличие пропусков, выбросов), выявление характера взаимосвязи между переменными с целью последующего выдвижения гипотез о наиболее подходящих для решения задачи моделях машинного обучения.

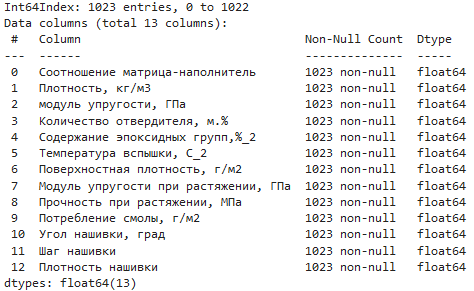
В качестве инструментов анализа и визуализации данных используются оценка статистических характеристик датасета; гистограммы распределения каждого признака; диаграммы ящика с усами; попарные графики рассеяния точек; квантиль-квантильный график; тепловая карта взаимной корелляции признаков; проверка наличия пропусков и дубликатов.

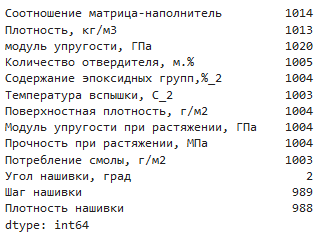
Проведем разведочный анализ данных.

На входе имеется два файла с наборами данных X\_bp.xlsx, X\_nup.xlsx, число строк в них различное, 1023 и 1040 соответственно. По условию задачи объединим данные файлы в один датасет по индексу с типом объединения INNER, при этом остаются строки, присутствующие только в обоих датасетах.

Информация о датасете представлена на рисунке 3. Видно, что все переменные содержат значения float64, качественные характеристики отсутствуют; пропусков не имеется; ни одна из записей не является NaN, очистка не требуется. Объединенный датасет имеет всего 1023 строки.

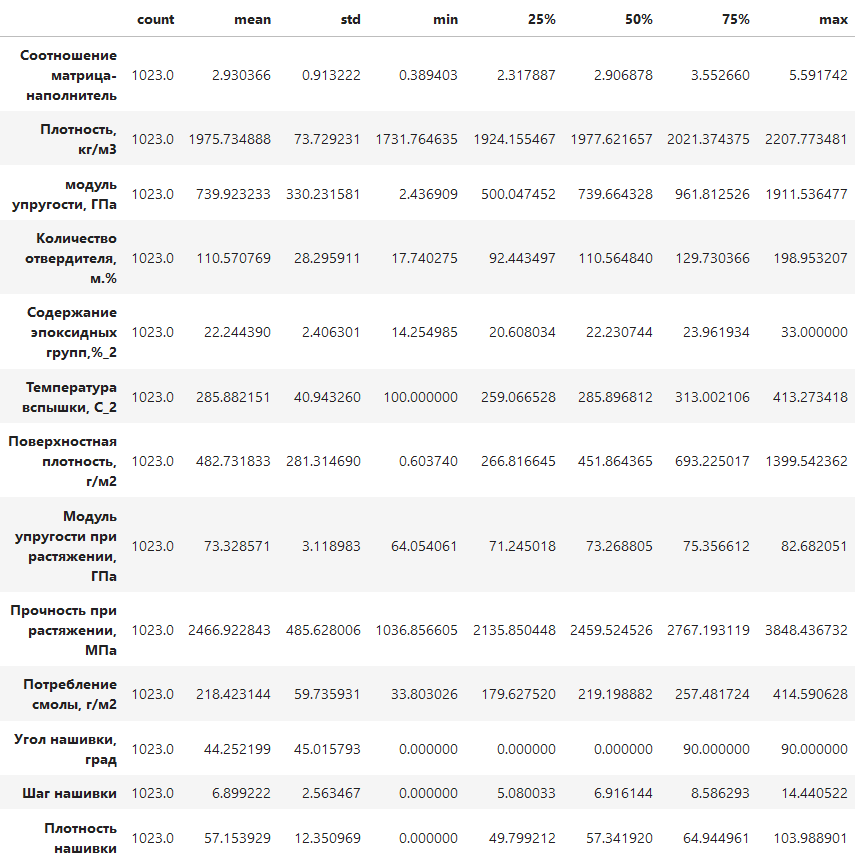
На рисунке 4 представлено содержание уникальных значений в датасете. В основном в каждом столбце содержатся только уникальные значения, но в столбце "Угол нашивки, град" всего 2 значения.

Рисунок 3 − Информация о датасете

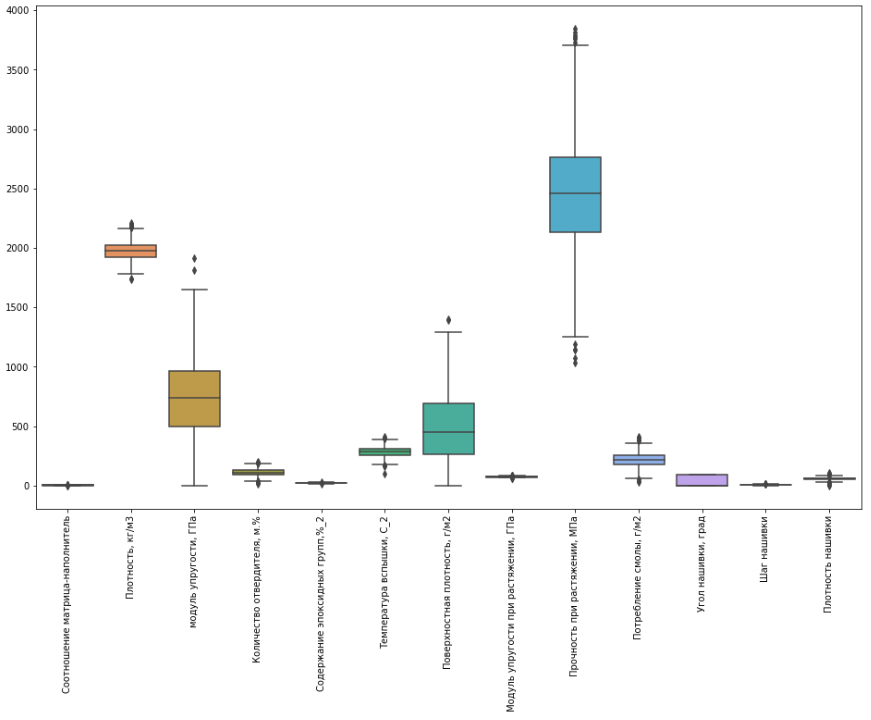
Рисунок 4 − Содержание уникальных значений в столбцах датасета

Дополнительный анализ датасета показывает, что пропущенные данные отсутствуют. Также в датасете отсутствуют дубликаты в записях.

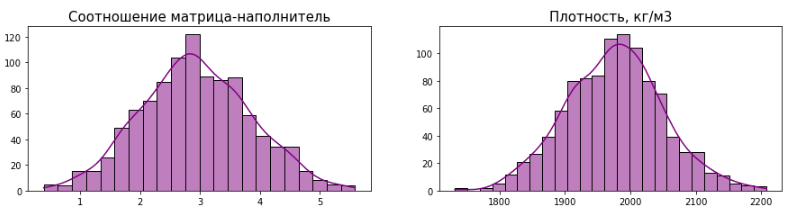
Из описательной статистики на рисунке 5 видно, что медианное и среднее значение каждой переменной довольно близки друг к другу, кроме угла нашивки, что говорит о распределении данных переменных близком к нормальному.

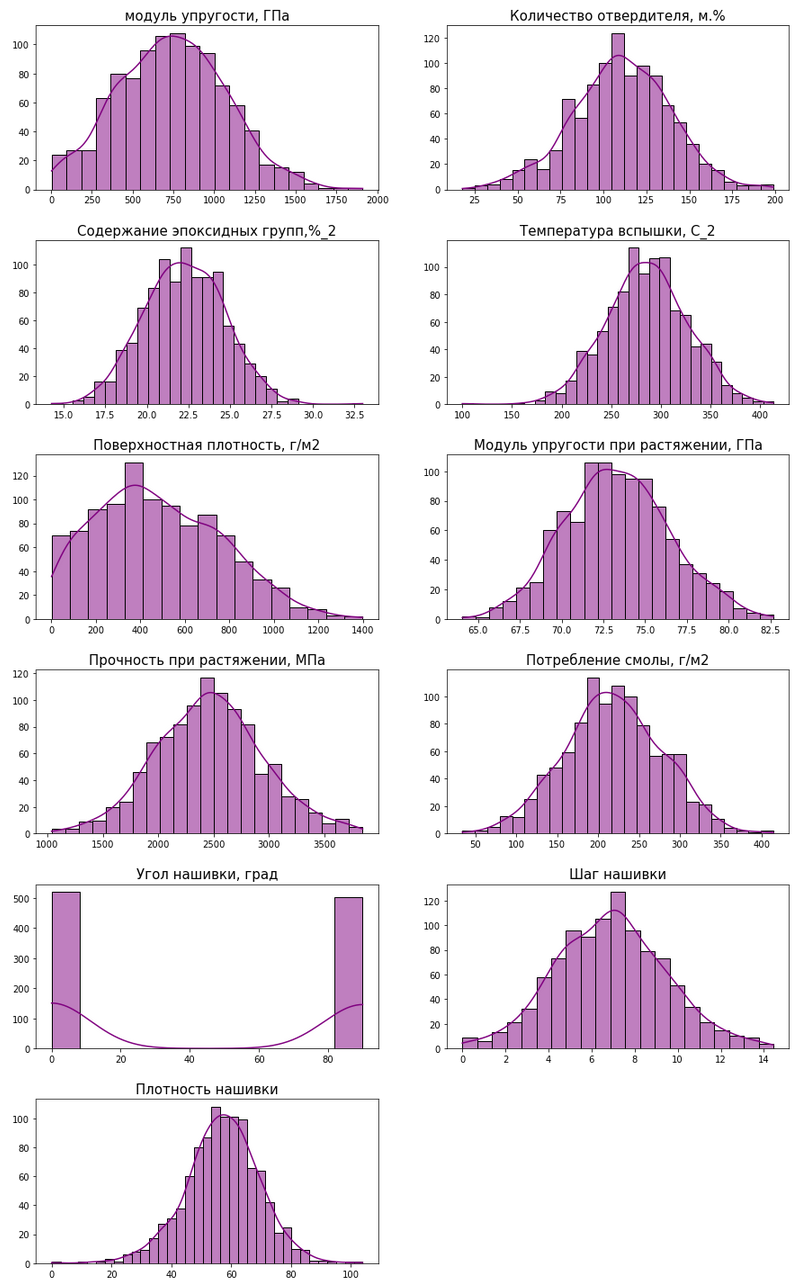
Рисунок 5 − Описательная статистика датасета

Построив график типа боксплот («ящик с усами») для исходного датасета, из рисунка 6 видно, что порядок значений переменных сильно различается − практически в 1000 раз.

Рисунок 6 − Описательная статистика датасета

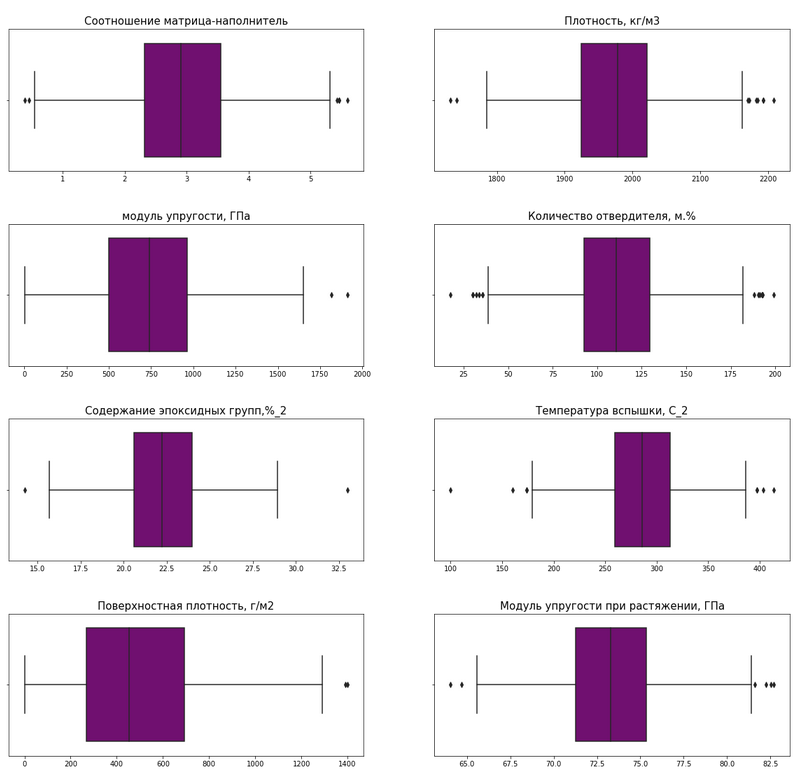
Для более информативного отображения характера распределения переменных в датасете можно построить гистаграммы распределения (рисунки 7 и 8).

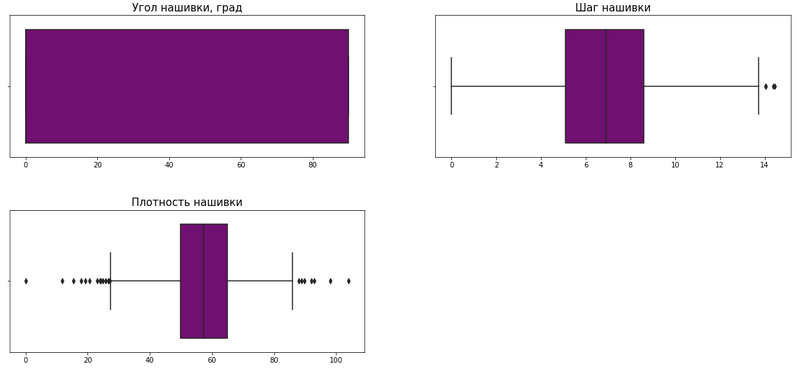
Рисунок 7 − Гисторгаммы распределения для переменных датасета

Рисунок 8 − Гисторгаммы распределения для переменных датасета

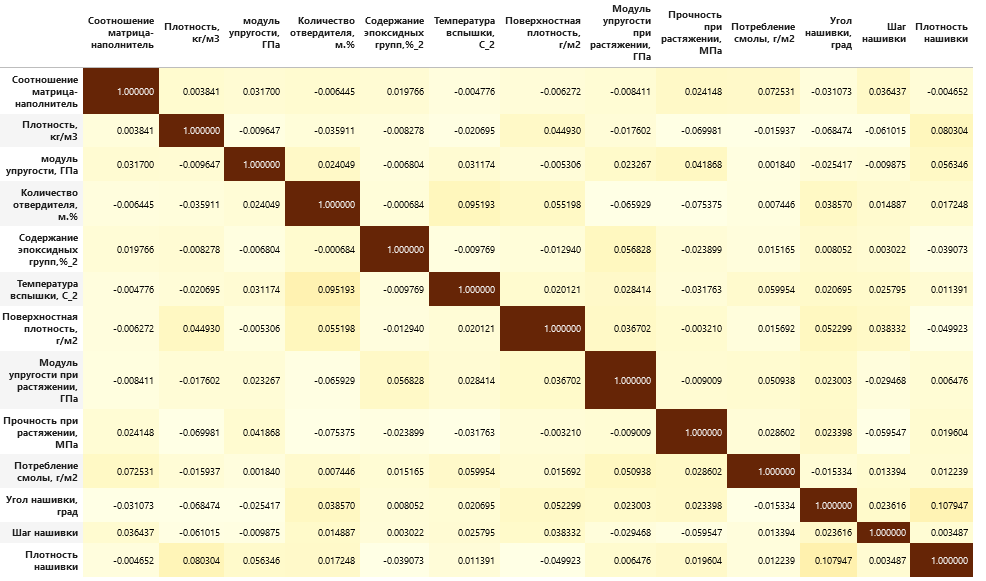
По гистограммам распределения видно, что распределение величин близко к нормальному для большей части параметров, за исключением поверхностной плотности – большое смещением влево и угла нашивки – дискретная величина, график оказался не показателен.

Построив диаграммы «ящик с усами» для переменных датасета (рисунки 9 и 10), видно что практически у всех переменных имеются выбросы, кроме угла нашивки, так как данный параметр принимает дискретные значения и диаграмма «ящик с усами» для него не показательна.

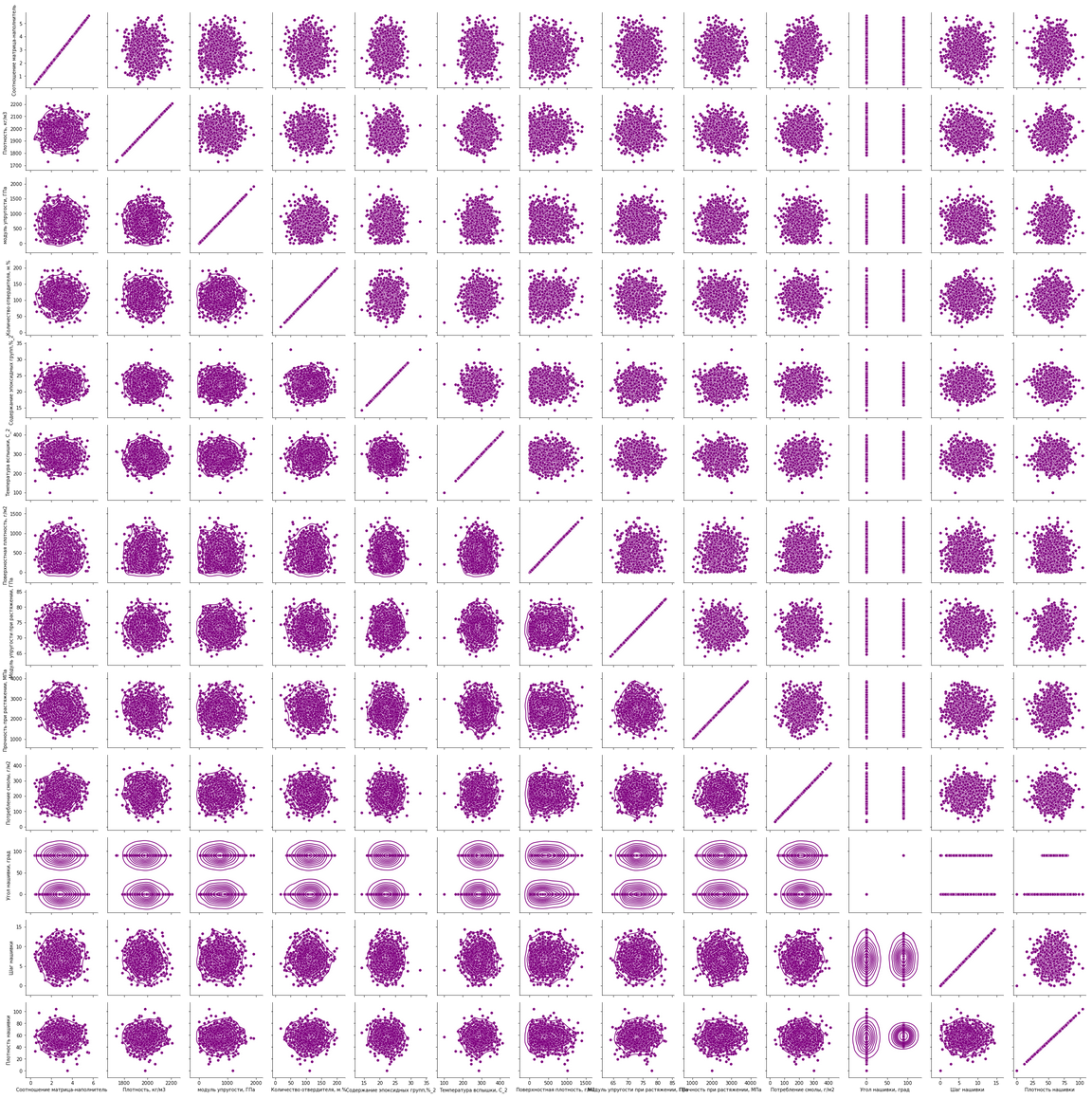
Рисунок 9 − Диаграммы «ящик с усами» переменных в датасете

Рисунок 10 − Диаграммы «ящик с усами» переменных в датасете

В целях выявления зависимостей между переменными можно построить тепловую карта коэффициентов корреляции (рисунок 11). Коэффициенты корелляции предварительно показывают, что явная зависимость между переменными датасета отсутствует.

Рисунок 11 − Тепловая карта коэффициентов корреляции

Также можно построить попарные графики рассеивания (рисунок 12), на которых также не просматривается какой-либо зависимости между переменными.

Рисунок 12 − Попарные графики рассеивания

# 2 Практическая часть

## 2.1 Предобработка данных

После предварительного анализа датасета нужно привести данные к виду, при котором модели машинного обучения могут наиболее эффективно обучаться.

Для начала необходимо удалить выбросы. Для удаления выбросов существует 2 основных метода - метод 3-х сигм и межквартильных расстояний. Так как у нас датасет небольшой по объему, удалим только экстремальные выбросы методом 3-х сигм. Результаты на рисунке 13.

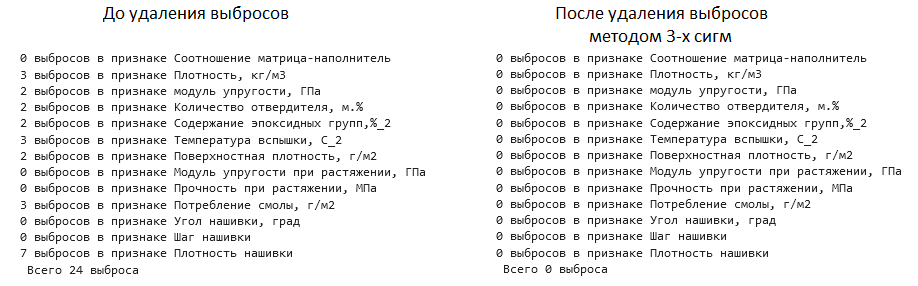
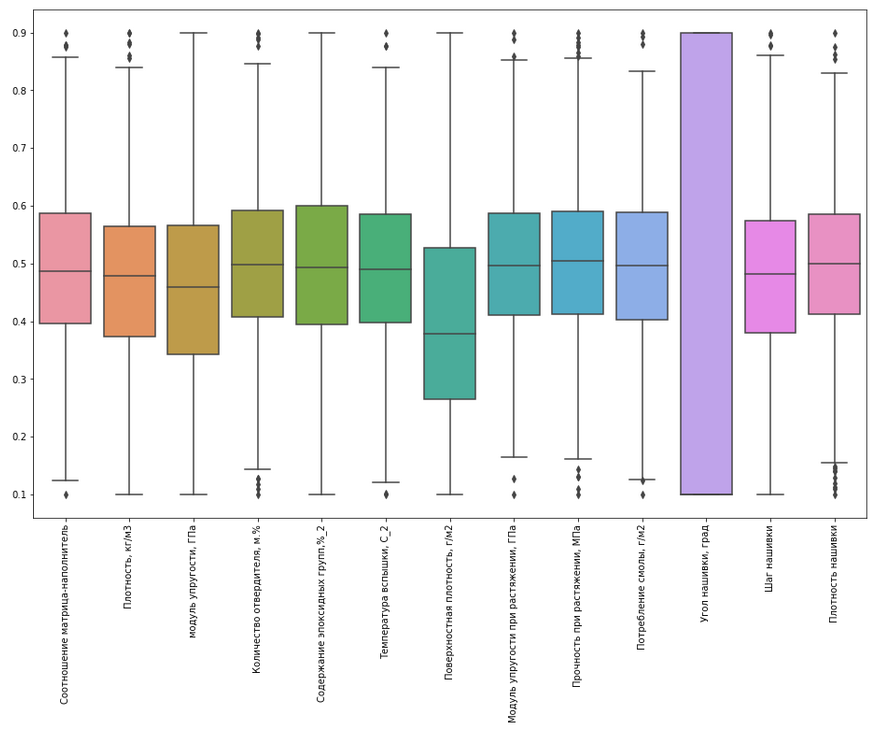
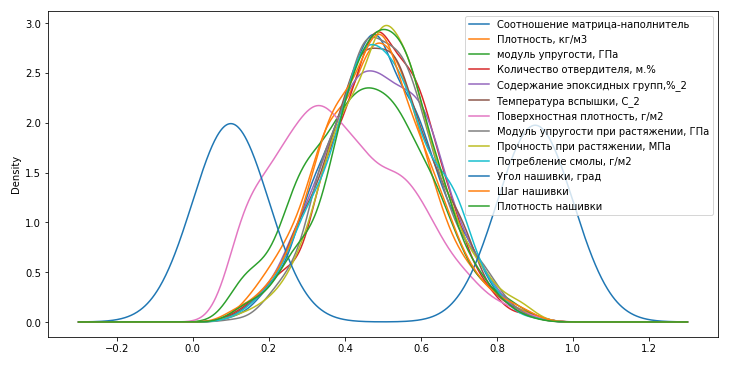


Рисунок 13 − Количество выбросов до и после удаления методом 3-х сигм

Дальнейшая подготовка сводится к нормализации и стандартизации данных. Наиболее распространенные методы для нормализации данных − это MinMaxScaler(), Normalizer(), для стандартизации данных − StandardScaler(). Рассмотрим особенности применения каждого из методов.

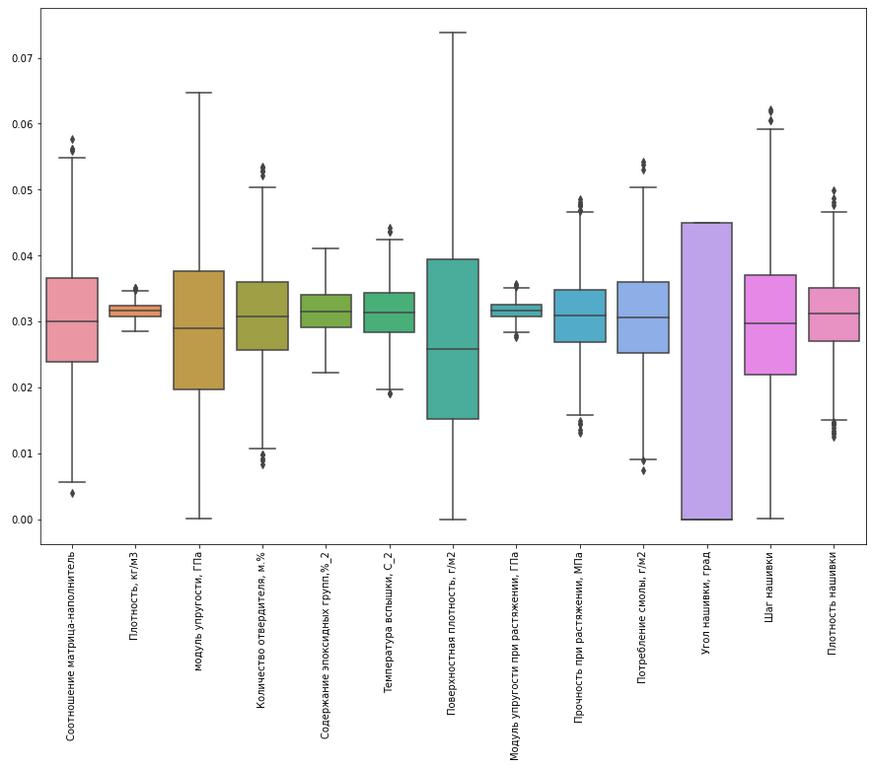
Нормализация при промощи метода MinMaxScaler() выполняется к заданному диапазону, по умолчанию от 0 до 1. Применим метод MinMaxScaler() к нашему датасету, указав диапазон от 0,1 до 0,9. На рисунках 14 и 15 отображены результаты преобразования.

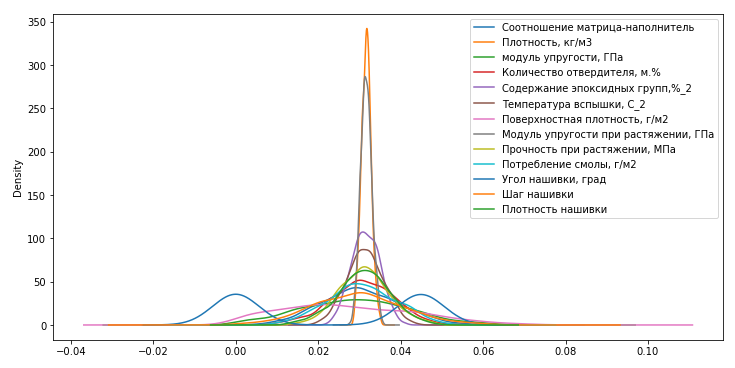
Рисунок 14 − Боксплоты переменных датасета после MinMaxScaler()

Рисунок 15 − График оценки плотности ядра датасета после MinMaxScaler()

Нормализация при промощи метода Normalizer() − это процесс масштабирования отдельных образцов до единичной нормы . Этот процесс может быть полезен, если планируется использовать квадратичную форму, такую как скалярное произведение или любое другое ядро, для количественной оценки подобия любой пары образцов. Это предположение является основой модели векторного пространства, часто используемой в контекстах классификации и кластеризации текста. Применим метод Normalizer() к нашему датасету.

На рисунках 16 и 17 отображены результаты преобразования.

Рисунок 16 − Боксплоты переменных датасета после Normalizer()

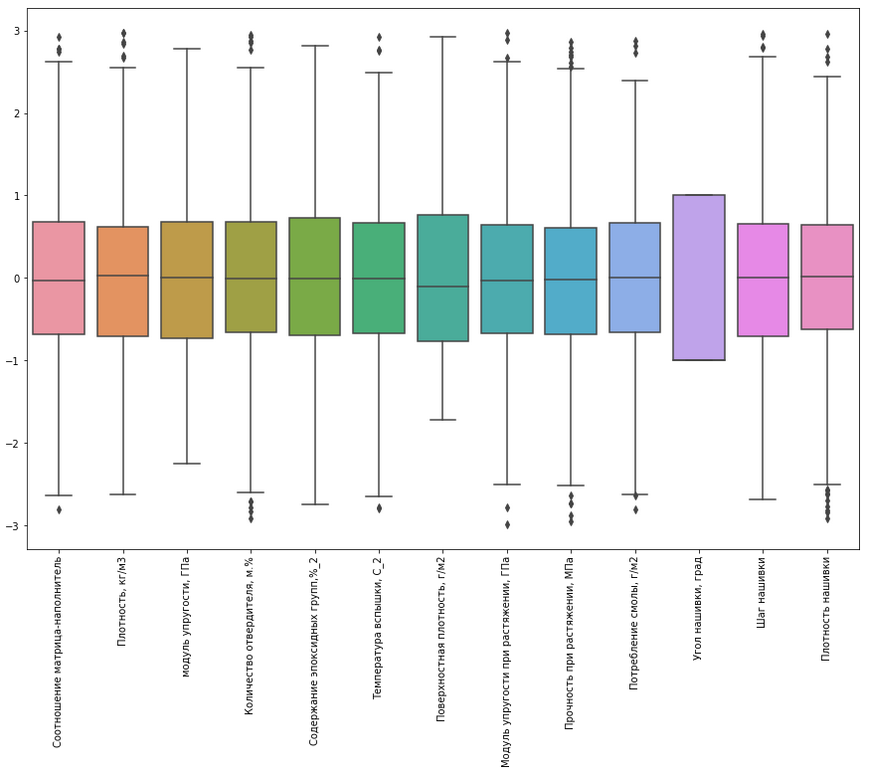
Рисунок 17 − График оценки плотности ядра датасета после Normalizer()

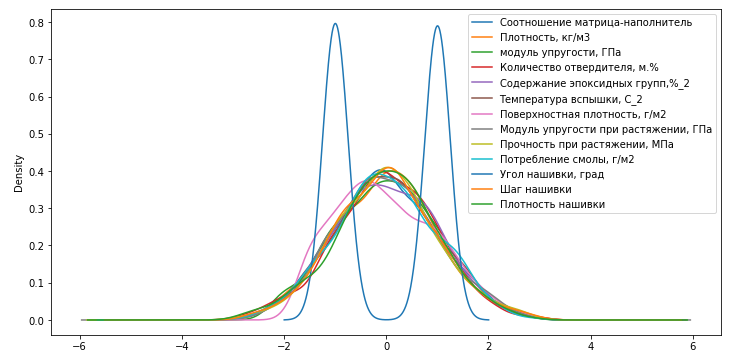
Стандартизация наборов данных является общим требованием для многих оценщиков машинного обучения, реализованных в scikit-learn; они могут вести себя плохо, если отдельные функции не более или менее выглядят как стандартные нормально распределенные данные: гауссовские с нулевым средним и единичной дисперсией. На практике часто игнорируется форма распределения и просто преобразуем данные для их центрирования, удаляя среднее значение каждой функции, а затем масштабируется, деля непостоянные характеристики на их стандартное отклонение.

Применим метод StandardScaler() к нашему датасету.

На рисунках 18 и 19 отображены результаты преобразования.

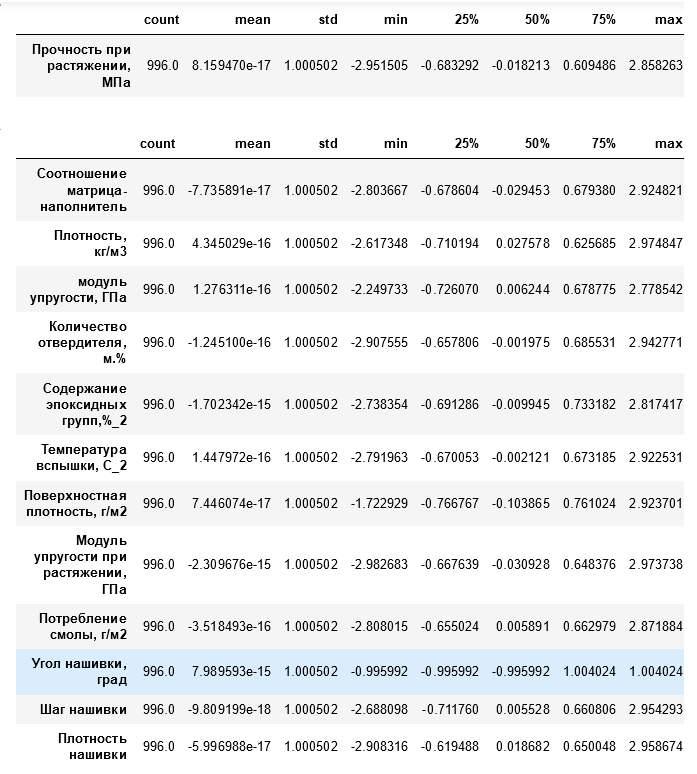
Масштабированные данные имеют нулевое среднее значение и единичную дисперсию

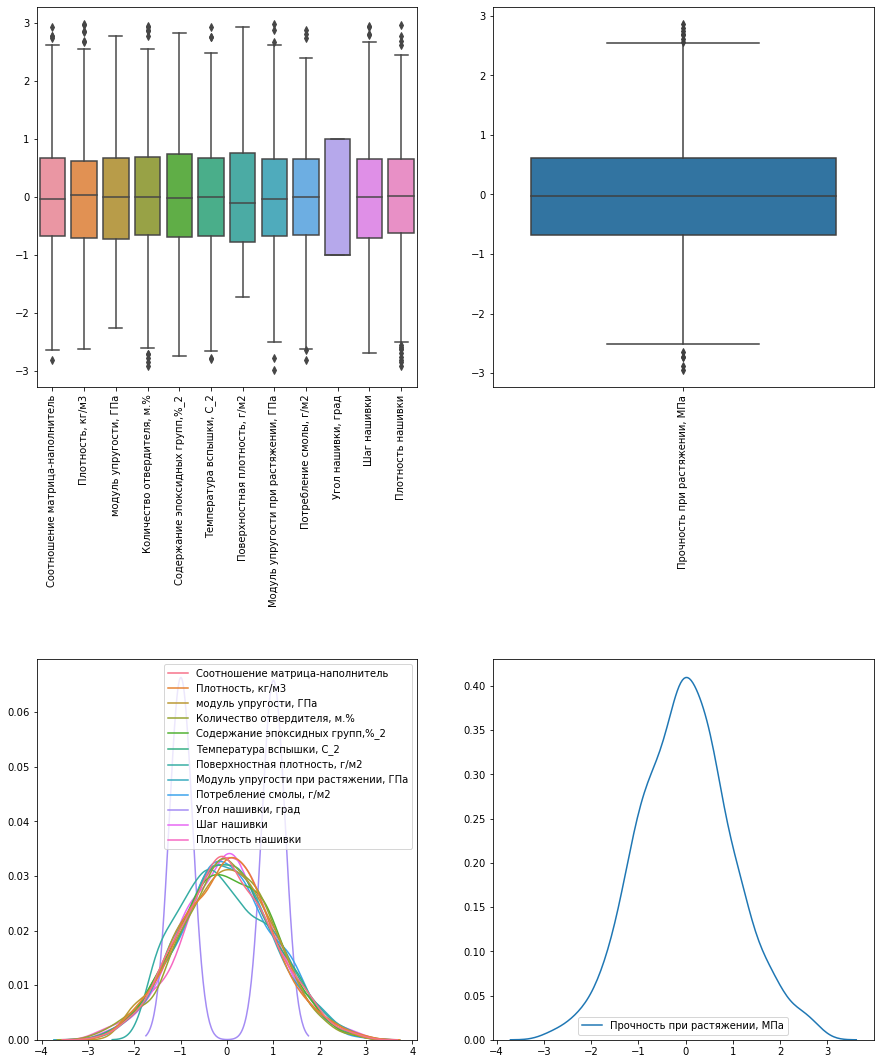
Рисунок 18 − Боксплоты переменных датасета после StandardScaler()

Рисунок 19 − График оценки плотности ядра датасета после StandardScaler()

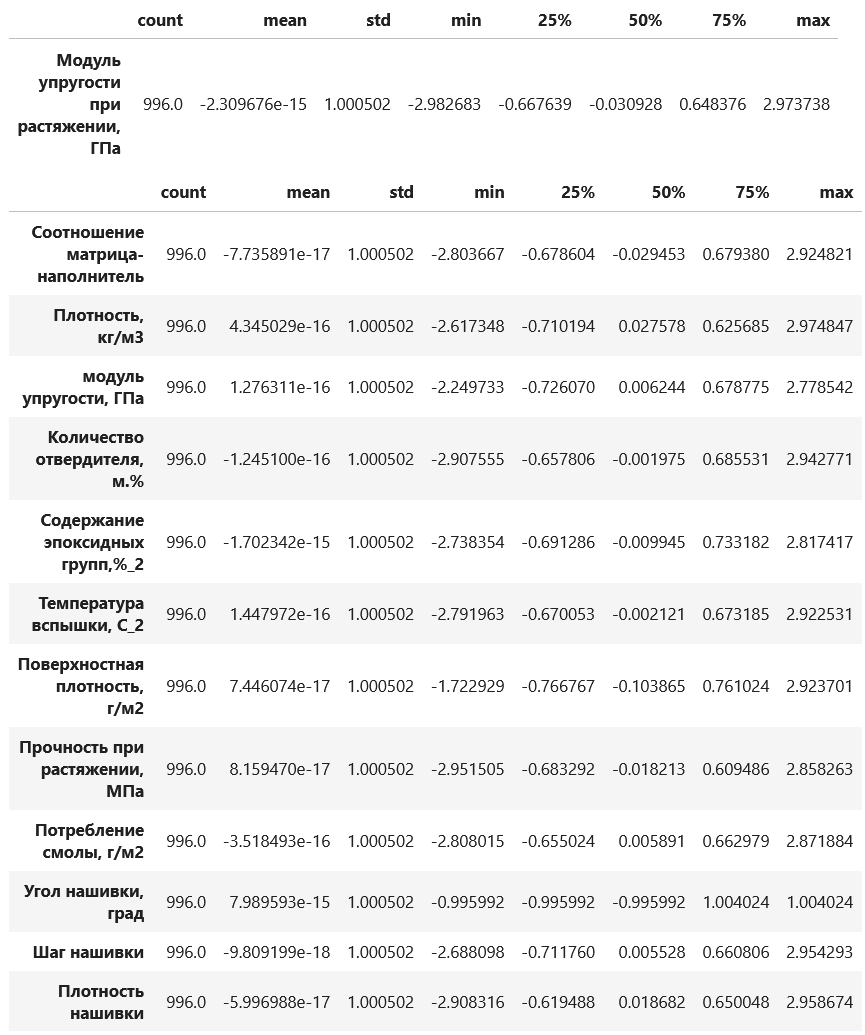
Для дальнейшей работы разделим наш датасет на два — целевая переменная и признаки и применим к данным датасетам метод StandardScaler()

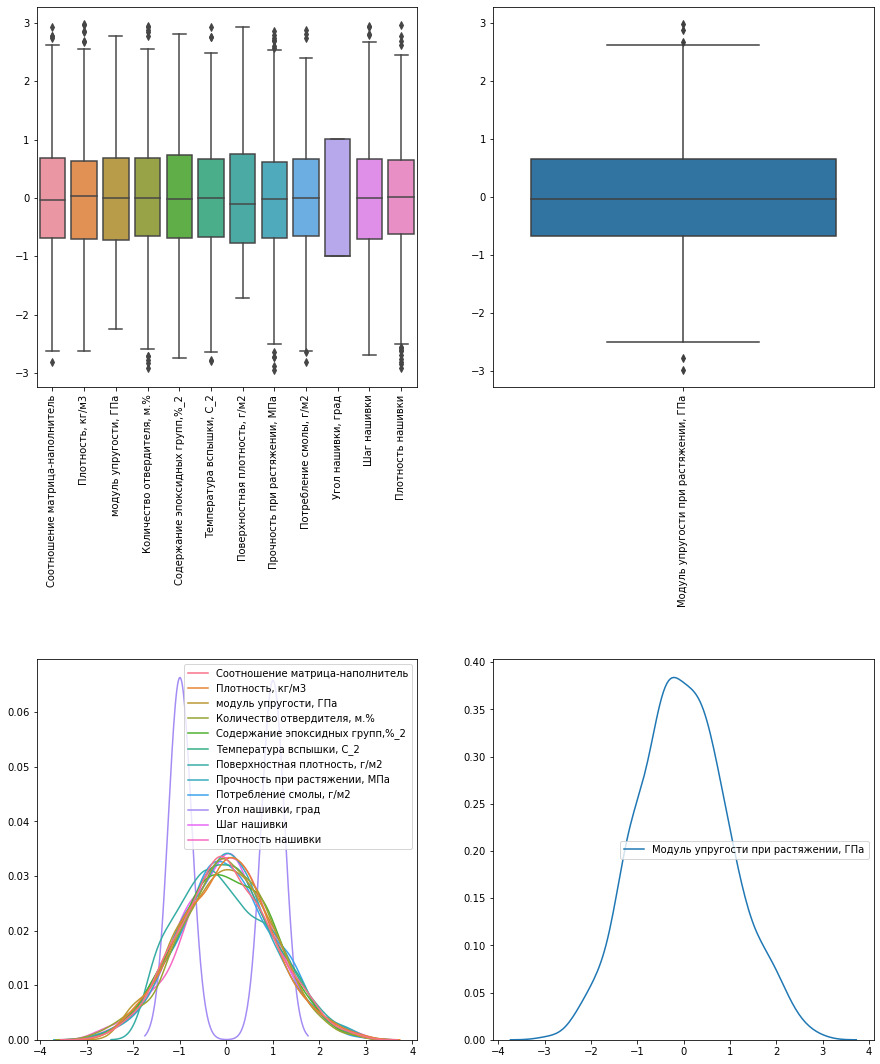
Для случая прогнозирования прочности при растяжении после применения StandardScaler() описательная статистика и визуализация переменных датасета выглядит следующим образом (рисунок 20 и 21).

Рисунок 20 − Описательная статистика датасета после стандартизации методом StandardScaler() для случая прогнозирования прочности при растяжении

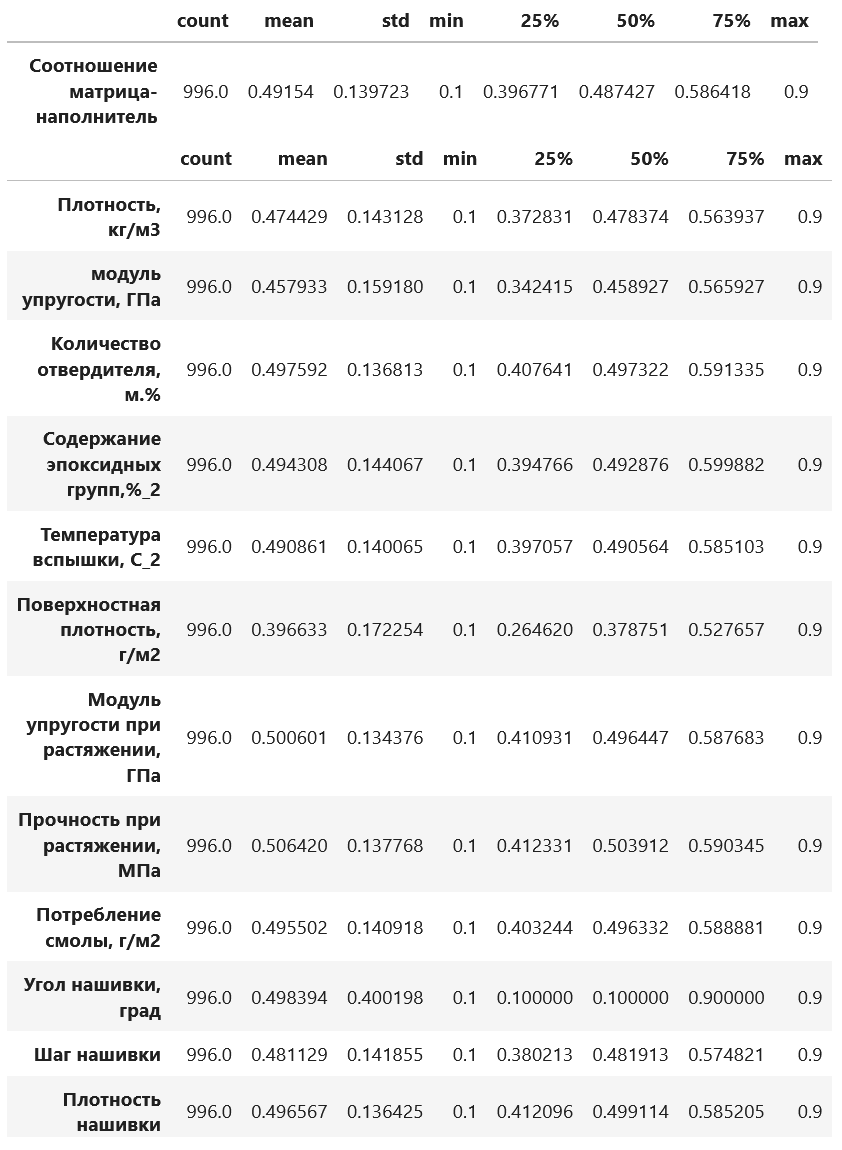
Рисунок 21 − Визуализация переменных датасета после стандартизации методом StandardScaler() для случая прогнозирования прочности при растяжении

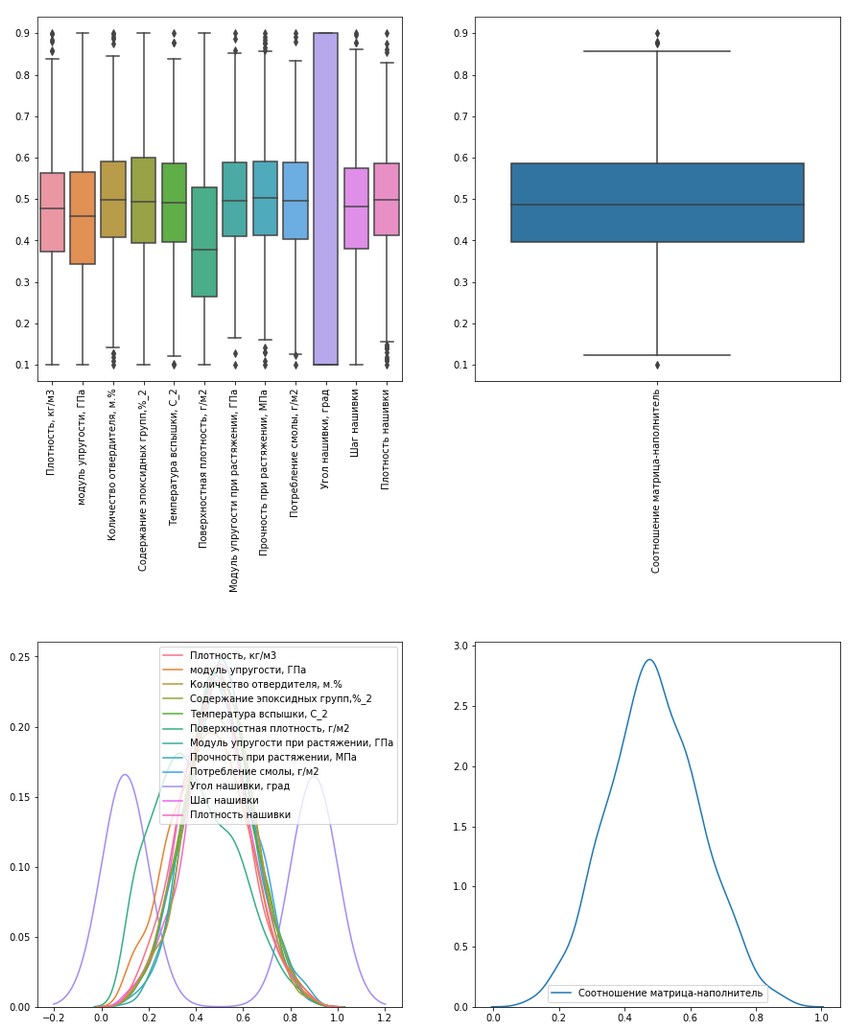
Для случая прогнозирования модуля упругости при растяжении после применения StandardScaler() описательная статистика и визуализация переменных датасета выглядит следующим образом (рисунок 22 и 23).

Рисунок 22 − Описательная статистика датасета после стандартизации методом StandardScaler() для случая прогнозирования модуля упругости при растяжении

Рисунок 23 − Визуализация переменных датасета после стандартизации методом StandardScaler() для случая прогнозирования модуля упругости при растяжении

Для случая предсказания нейросетью соотношения матрица-наполнитель применим MinMaxScaler(), описательная статистика и визуализация переменных датасета выглядит следующим образом (рисунок 24 и 25).

Рисунок 24 − Описательная статистика датасета после стандартизации методом MinMaxScaler() для случая предсказания соотношения матрица

Рисунок 25 − Визуализация переменных датасета после стандартизации методом MinMaxScaler() для случая предсказания соотношения матрица

Для дальнейшего использования нормализаторы были сохранены в отдельные файлы при помощи библиотеки Joblib.

Перед обучением моделей датасеты были разделены на обучающую и тестовую выборки, в соответствии с условием задачи 70% на обучение и 30% на тестирование.

## 2.2 Разработка и обучение модели

В ранной работе использовались методы и модели машинного обучения из широкоизвестной библиотеки scikit-learn. При разработке и обучении моделей был проведен поиск оптимальных гиперпараметров моделей с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10, для чего был применен метод GridSearchCV() с параметрами: количество перекрестных проверок cv = 10, сравнение качества моделей по коэффициенту детерминации scoring = 'r2'.

В дальнейшем для оценки качества результатов обучения использованы такие показатели, как средняя абсолютная ошибка (MAE), средняя квадратичная ошибка (MSE), коэффициент детерминации(R²)

MAE - измеряет среднюю абсолютную ошибку прогнозов. Для каждой точки вычисляется разница между прогнозами и целью, а затем усредняются эти значения. Чем дальше это значение от нуля, тем хуже модель. Показатель принимает так же нулевое значение, что может интерпретироваться как обучение идеальной модели, но и как, то, что сумма положительных и отрицательных значений прогнозов в итоге дала ноль.

MAPE - средняя абсолютная ошибка в процентах

MSE - измеряет среднюю квадратичную ошибку прогнозов. Для каждой точки вычисляется квадратная разница между прогнозами и целью, а затем усредняются эти значения. Чем выше это значение, тем хуже модель. Он никогда не бывает отрицательным, поскольку мы возводим в квадрат отдельные ошибки прогнозирования, прежде чем их суммировать, но для идеальной модели это будет ноль.

RMSE – квадратный корень из среднеквадратичной ошибки

R² - rоэффициент детерминации, или R-квадрат, является еще одним показателем, который мы можем использовать для оценки модели, и он тесно связан с MSE, но имеет преимущество в том, что не имеет значения, являются ли выходные значения очень большими или очень маленькими, R² всегда будет между -∞ и 1. Коэффициент детерминации для модели с константой принимает значения от 0 до 1. Чем ближе значение коэффициента к 1, тем сильнее зависимость. При оценке регрессионных моделей это интерпретируется как соответствие модели данным. Для приемлемых моделей предполагается, что коэффициент детерминации должен быть хотя бы не меньше 50 %. Модели с коэффициентом детерминации выше 80 % можно признать достаточно хорошими. Значение коэффициента детерминации 1 означает функциональную зависимость между переменными. Когда R² отрицательно, это означает, что модель хуже, чем предсказание среднего значения.

### 2.2.1 Предсказание прочности при растяжении

Метод линейной регрессии (LinearRegression): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода LinearRegression(fit\_intercept = False, positive = False). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.03.

Метод опорных векторов (SVR): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода SVR(gamma= 'scale', kernel= 'linear', shrinking= False). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.02.

Метод К ближайших соседей (KneighborsRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода KNeighborsRegressor(algorithm= 'brute', n\_neighbors= 10, weights= 'distance'). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 1.00.

Метод решающих деревьев (DecisionTreeRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода DecisionTreeRegressor(criterion= 'friedman\_mse', random\_state= 42, splitter= 'random'). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 1.00.

Метод стахостического градиентного спуска (SGDRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода SGDRegressor(fit\_intercept= False, learning\_rate= 'invscaling', loss= 'huber', penalty= None, random\_state= 42, shuffle= True). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.02.

Многослойный перцептрон (MLPRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода MLPRegressor(activation= 'logistic', hidden\_layer\_sizes= (100,), learning\_rate= 'constant', nesterovs\_momentum= True, random\_state= 42, shuffle= True, solver= 'sgd'). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.02.

Метод Лассо регрессии (Lasso): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода Lasso(fit\_intercept= False, positive= True, precompute= True, random\_state= 42, selection= 'cyclic'). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: -0.00.

Ансамблевые методы.

Метод случайного леса (Random Forest Regressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода RandomForestRegressor (bootstrap= True, criterion= 'absolute\_error', max\_features= 'sqrt', max\_samples= None, n\_estimators= 50, oob\_score= True, random\_state= 42). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.86.

Метод градиентного бустинга (GradientBoostingRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода GradientBoostingRegressor(criterion= 'friedman\_mse', loss= 'squared\_error', max\_depth= 1, random\_state= 42). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.13.

Стекинг (StackingRegressor): ансабль был составлен из методов LinearRegression(fit\_intercept= False, positive= False) и DecisionTreeRegressor(criterion= 'friedman\_mse', random\_state= 42, splitter= 'random'). Объединяющим результаты ансамбля методом служил метод LinearRegression(). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: -0.14.

### 2.2.2 Предсказание модуля упругости при растяжении

Метод линейной регрессии (LinearRegression): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода LinearRegression(fit\_intercept= False, positive= True). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.01.

Метод опорных векторов (SVR): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода SVR(gamma= 'scale', kernel= 'linear', shrinking= False). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.01.

Метод К ближайших соседей (KneighborsRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода KNeighborsRegressor(algorithm= 'auto', n\_neighbors= 10, weights= 'uniform'). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.08.

Метод решающих деревьев (DecisionTreeRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода DecisionTreeRegressor(criterion= 'absolute\_error', random\_state= 42, splitter= 'best'). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 1.00.

Метод стахостического градиентного спуска (SGDRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода SGDRegressor(fit\_intercept= False, learning\_rate= 'invscaling', loss= 'huber', penalty= 'l1', random\_state= 42, shuffle= False). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.01.

Многослойный перцептрон (MLPRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода MLPRegressor(activation= 'logistic', hidden\_layer\_sizes= (128, 2, 64), learning\_rate= 'constant', nesterovs\_momentum= False, random\_state= 42, shuffle= True, solver= 'sgd'). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: -0.00.

Метод Лассо регрессии (Lasso): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода Lasso(fit\_intercept= False, positive= True, precompute= True, random\_state= 42, selection= 'cyclic'). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: -0.00.

Ансамблевые методы

Метод случайного леса (RandomForestRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода RandomForestRegressor(bootstrap= True, criterion= 'squared\_error', max\_features= None, max\_samples= 1, n\_estimators= 50, oob\_score= True, random\_state= 42). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: -0.01.

Метод градиентного бустинга (GradientBoostingRegressor): по результатам поиска оптимальных гиперпараметров выявлены основные настройки для получения наилучшего результата при применении метода GradientBoostingRegressor(criterion= 'friedman\_mse', loss= 'squared\_error', max\_depth= 1, random\_state= 42). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: 0.10.

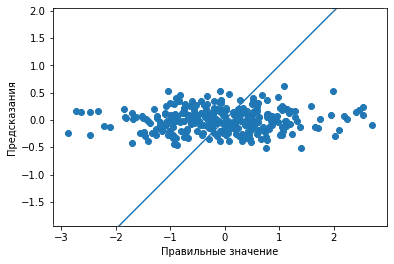
Стекинг (StackingRegressor): ансабль был составлен из методов LinearRegression(fit\_intercept= False, positive= False) и DecisionTreeRegressor(criterion= 'friedman\_mse', random\_state= 42, splitter= 'random'). Объединяющим результаты ансамбля методом служил метод LinearRegression(). С данными настройками модель была обучена на тренировочной выборке. Результат на тренировочной выборке: R²: -0.04.

## 2.3 Тестирование модели

### 2.3.1 Предсказание прочности при растяжении

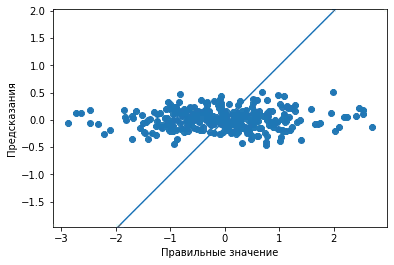
Метод линейной регрессии (LinearRegression).

Результат на тестовой выборке: R²: -0.03, MAE: 0.78, MAPE: 1.97, MSE: 0.97, RMSE: 0.98. Графическое представление точности предсказания на рисунке 26.

Рисунок 26 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода LinearRegression()

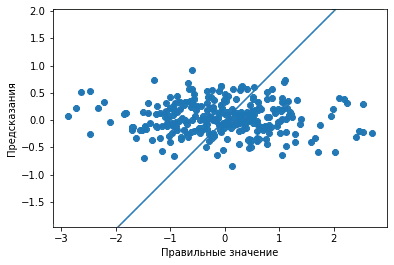
Метод опорных векторов (SVR).

Результат на тестовой выборке: R²: -0.02, MAE: 0.78, MAPE: 2.77, MSE: 0.96, RMSE: 0.98. Графическое представление точности предсказания на рисунке 27.

Рисунок 27 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода SVR()

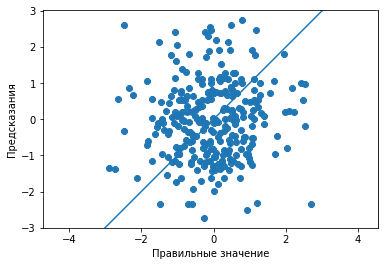
Метод К ближайших соседей (KneighborsRegressor).

Результат на тестовой выборке: R²: -0.14, MAE: 0.81, MAPE: 6.23, MSE: 1.07, RMSE: 1.03. Графическое представление точности предсказания на рисунке 28.

Рисунок 28 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода KneighborsRegressor()

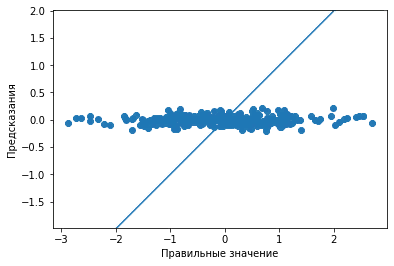
Метод решающих деревьев (DecisionTreeRegressor).

Результат на тестовой выборке: R²: -1.04, MAE: 1.11, MAPE: 7.13, MSE: 1.92, RMSE: 1.39. Графическое представление точности предсказания на рисунке 29.

Рисунок 29 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода DecisionTreeRegressor()

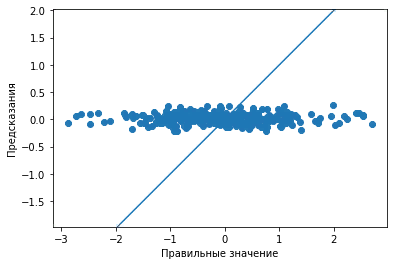
Метод стахостического градиентного спуска (SGDRegressor).

Результат на тестовой выборке: R²: -0.00, MAE: 0.76, MAPE: 1.42, MSE: 0.94, RMSE: 0.97. Графическое представление точности предсказания на рисунке 30.

Рисунок 30 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода SGDRegressor()

Многослойный перцептрон (MLPRegressor).

Результат на тестовой выборке: R²: -0.01, MAE: 0.77, MAPE: 1.54, MSE: 0.95, RMSE: 0.98. Графическое представление точности предсказания на рисунке 31.

Рисунок 31 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода MLPRegressor()

Метод Лассо регрессии (Lasso).

Результат на тестовой выборке: R²: -0.00, MAE: 0.76, MAPE: 1.00, MSE: 0.94, RMSE: 0.97. Графическое представление точности предсказания на рисунке 32.

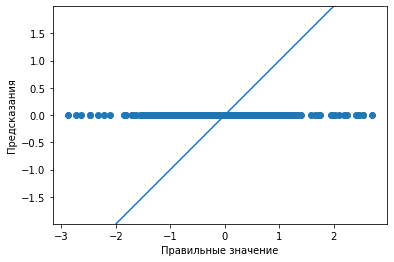
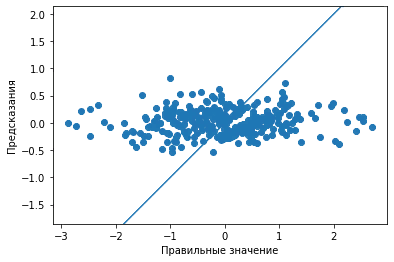


Рисунок 32 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода Lasso()

Ансамблевые методы.

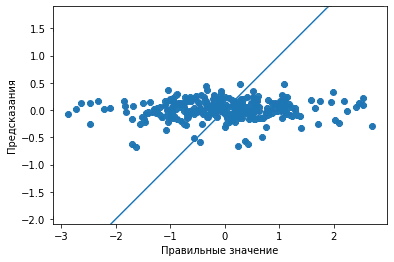
Метод случайного леса (RandomForestRegressor).

Результат на тестовой выборке: R²: -0.02, MAE: 0.78, MAPE: 2.62, MSE: 0.96, RMSE: 0.98. Графическое представление точности предсказания на рисунке 33.

Рисунок 33 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода RandomForestRegressor()

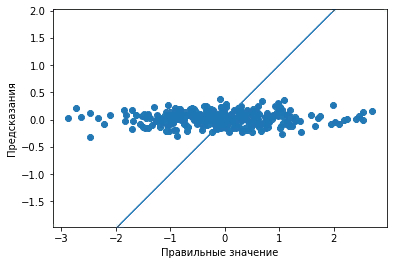
Метод градиентного бустинга (GradientBoostingRegressor)

Результат на тестовой выборке: R²: -0.02, MAE: 0.77, MAPE: 2.26, MSE: 0.96, RMSE: 0.98. Графическое представление точности предсказания на рисунке 34.

Рисунок 34 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода GradientBoostingRegressor()

Стекинг (StackingRegressor)

Результат на тестовой выборке: R²: -0.03, MAE: 0.77, MAPE: 1.56, MSE: 0.96, RMSE: 0.98. Графическое представление точности предсказания на рисунке 35.

Рисунок 35 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода StackingRegressor()

### 2.3.2 Предсказание модуля упругости при растяжении

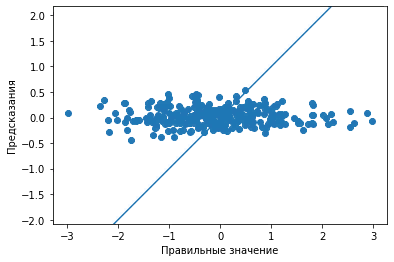
Метод линейной регрессии (LinearRegression)

Результат на тестовой выборке: R²: 0.00, MAE: 0.80, MAPE: 1.17, MSE: 1.01, RMSE: 1.01. Графическое представление точности предсказания на рисунке 36.

Рисунок 36 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода LinearRegression()

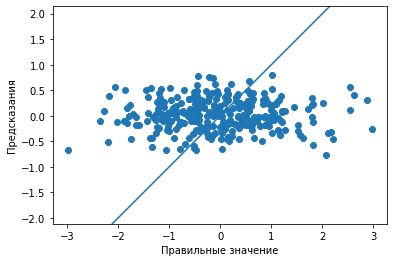
Метод опорных векторов (SVR)

Результат на тестовой выборке: R²: -0.04, MAE: 0.81, MAPE: 1.34, MSE: 1.06, RMSE: 1.03. Графическое представление точности предсказания на рисунке 37.

Рисунок 37 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода SVR()

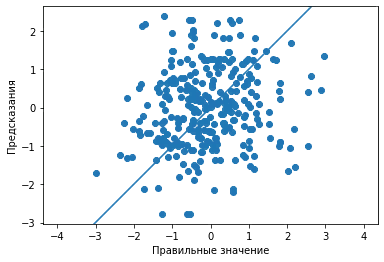
Метод К ближайших соседей (KneighborsRegressor)

Результат на тестовой выборке: R²: -0.09, MAE: 0.84, MAPE: 1.74, MSE: 1.10, RMSE: 1.05. Графическое представление точности предсказания на рисунке 38.

Рисунок 38 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода KneighborsRegressor()

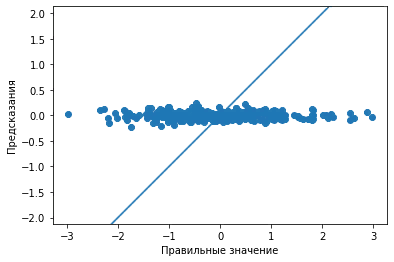
Метод решающих деревьев (DecisionTreeRegressor)

Результат на тестовой выборке: R²: -0.81, MAE: 1.08, MAPE: 3.30, MSE: 1.84, RMSE: 1.36. Графическое представление точности предсказания на рисунке 39.

Рисунок 39 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода DecisionTreeRegressor()

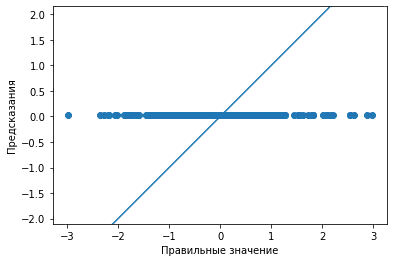
Метод стахостического градиентного спуска (SGDRegressor)

Результат на тестовой выборке: R²: -0.01, MAE: 0.08, MAPE: 1.10, MSE: 1.03, RMSE: 1.01. Графическое представление точности предсказания на рисунке 40.

Рисунок 40 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода SGDRegressor()

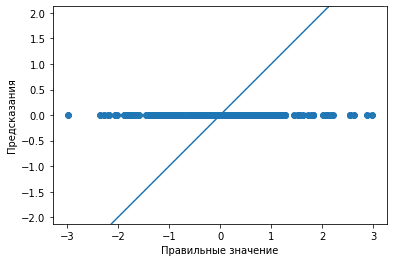
Многослойный перцептрон (MLPRegressor)

Результат на тестовой выборке: R²: -0.01, MAE: 0.80, MAPE: 1.02, MSE: 1.03, RMSE: 1.01. Графическое представление точности предсказания на рисунке 41.

Рисунок 41 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода MLPRegressor()

Метод Лассо регрессии (Lasso)

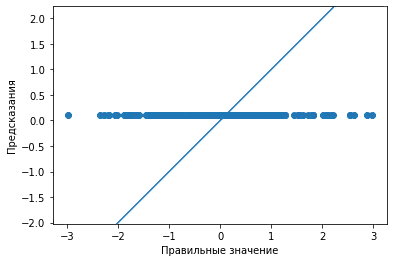
Результат на тестовой выборке: R²: -0.01, MAE: 0.80, MAPE: 1.02, MSE: 1.02, RMSE: 1.01. Графическое представление точности предсказания на рисунке 42.

Рисунок 42 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода Lasso()

Ансамблевые методы

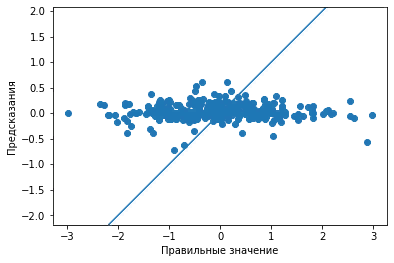
Метод случайного леса (RandomForestRegressor)

Результат на тестовой выборке: R²: -0.03, MAE: 0.82, MAPE: 1.23, MSE: 1.05, RMSE: 1.02. Графическое представление точности предсказания на рисунке 43.

Рисунок 43 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода RandomForestRegressor()

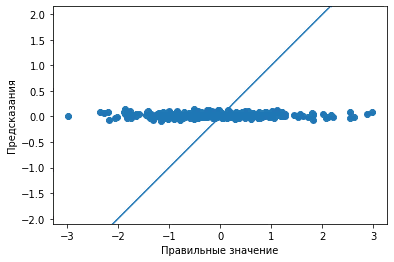
Метод градиентного бустинга (GradientBoostingRegressor)

Результат на тестовой выборке: R²: -0.04, MAE: 0.82, MAPE: 1.38, MSE: 1.05, RMSE: 1.03. Графическое представление точности предсказания на рисунке 44.

Рисунок 44 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода GradientBoostingRegressor()

Стекинг (StackingRegressor)

Результат на тестовой выборке: R²: -0.04, MAE: 0.81, MAPE: 1.10, MSE: 1.03, RMSE: 1.01. Графическое представление точности предсказания на рисунке 45.

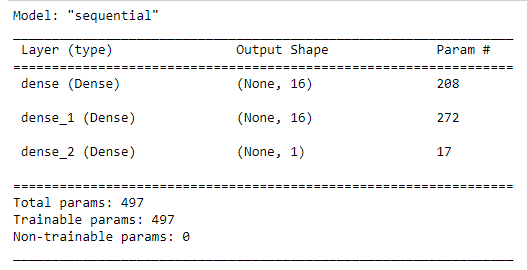
Рисунок 45 − График сравнения прогнозируемых значений с наблюдаемыми значениями для метода StackingRegressor()

## 2.4 Написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель

Для рекомендации соотношения матрица-наполнитель обучим несколько искусственных нейронных сетей с разной архитектурой.

Данные в нейросеть подаются номализованные в диапазоне от 0,1 до 0,9.

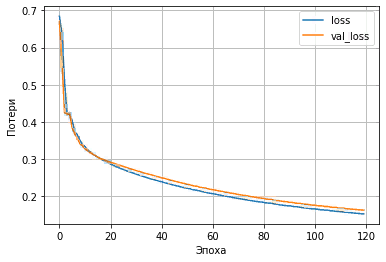
Первая нейросеть имеет два скрытых слоя по 16 нейронов, активационная функция на внутренних слоях "tanh", на выходном слое "relu". На рисунке 46 сводные данные первой нейросети.

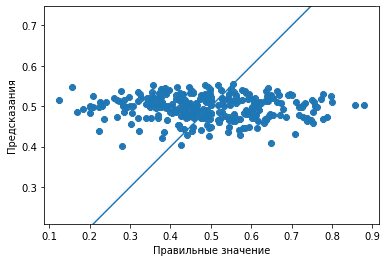
Рисунок 46 − Сводные данные первой нейросети

Модель скомпилирована с параметрами loss="mae", optimizer="adam", metrics = 'accuracy', дропаут-слои не применялись.

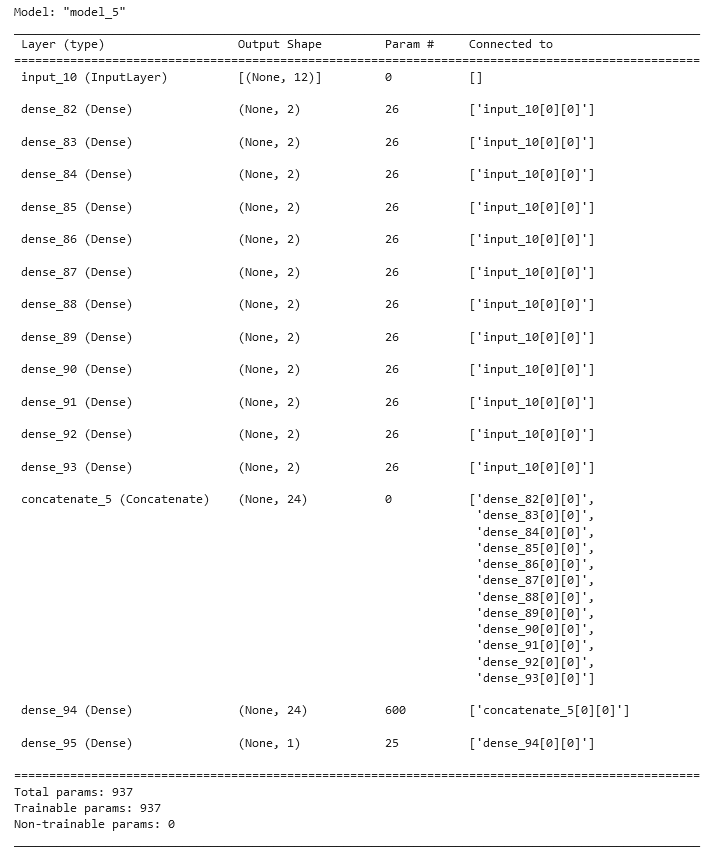
Обучение проводилось пакетами данных по 100 записей в течение 120 эпох. Процесс обучения показан на рисунке 47.

Результат на тестовой выборке: R²: -23.16, MAE: 0.12, MSE: 0.02. Графическое представление точности предсказания на рисунке 48.

Рисунок 47 − Обучение первой нейросети

Рисунок 48 − График сравнения прогнозируемых значений с наблюдаемыми значениями для первой нейросети

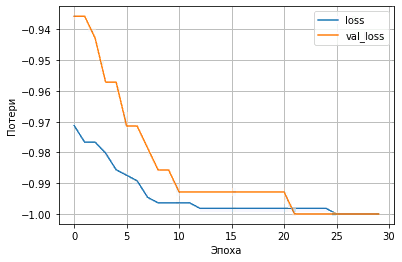
Вторая нейросеть имеет разветвленную структуру: после входного слоя разделяется по числу входных признаков на 12 потоков по 1 слою из 2 нейронов, далее потоки объединяются в один слой из 24 нейронов, активационная функция на внутренних слоях "tanh", на выходном слое "relu". На рисунке 49 сводные данные второй нейросети.

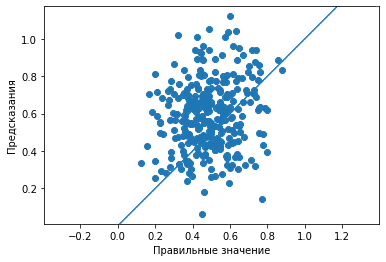
Рисунок 49 − Сводные данные второй нейросети

Модель скомпилирована с параметрами loss= "log\_cosh", optimizer= "adam", дропаут-слои не применялись.

Обучение проводилось пакетами данных по 150 записей в течение 30 эпох. Процесс обучения показан на рисунке 50.

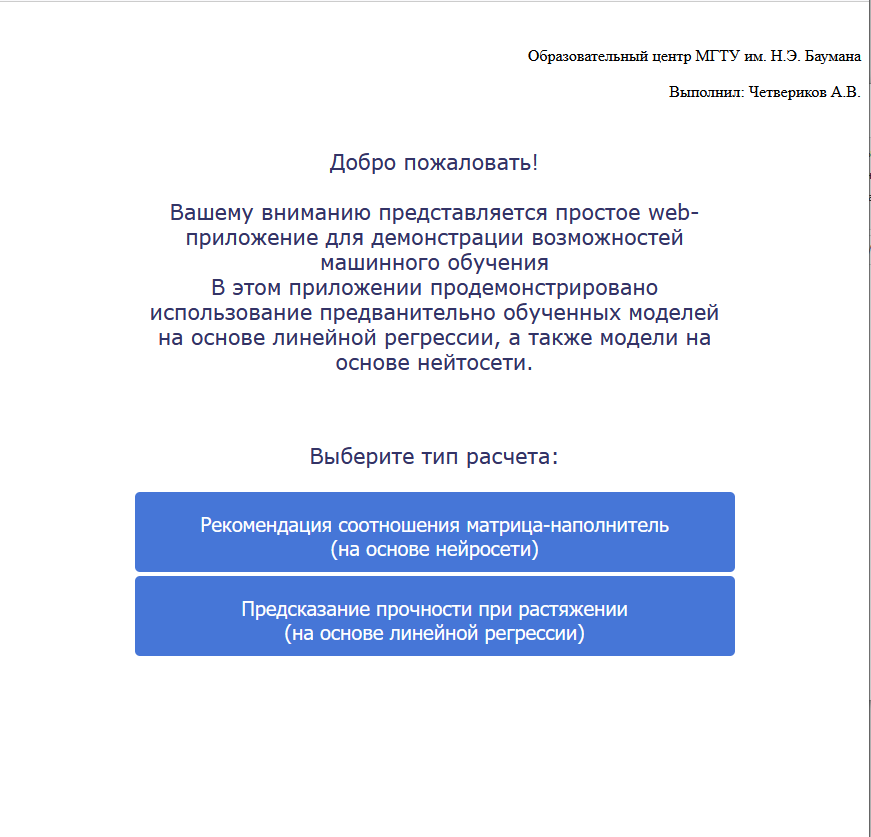
Результат на тестовой выборке: R²: -0.63, MAE: 0.19, MSE: 0.06. Графическое представление точности предсказания на рисунке 51.

Рисунок 50 − Обучение второй нейросети

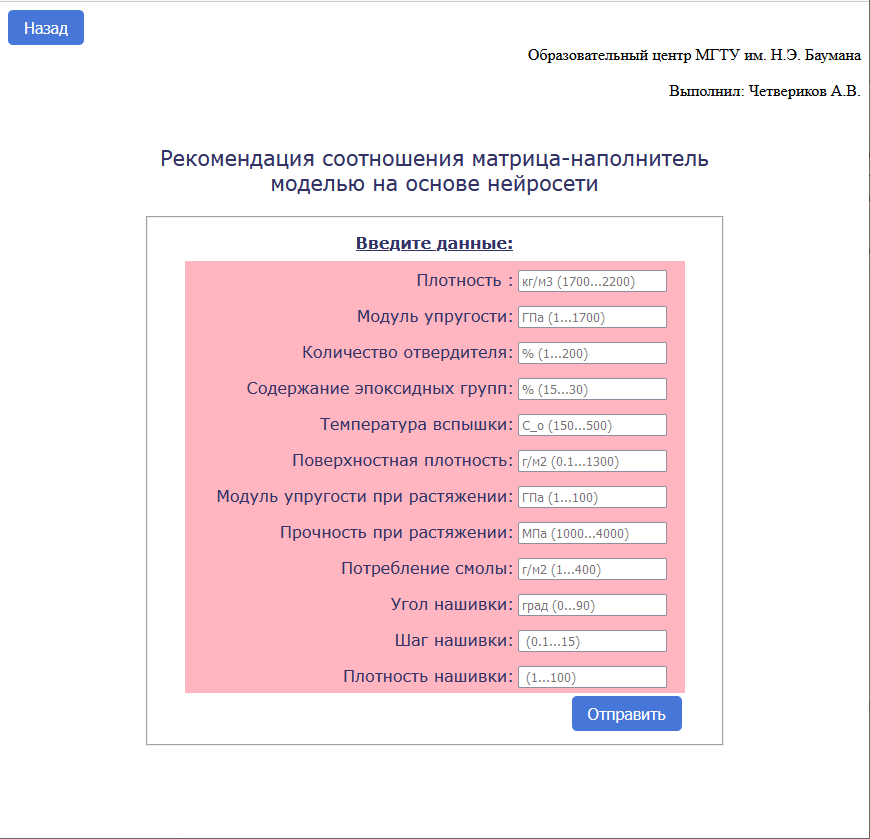
Рисунок 51 − График сравнения прогнозируемых значений с наблюдаемыми значениями для второй нейросети

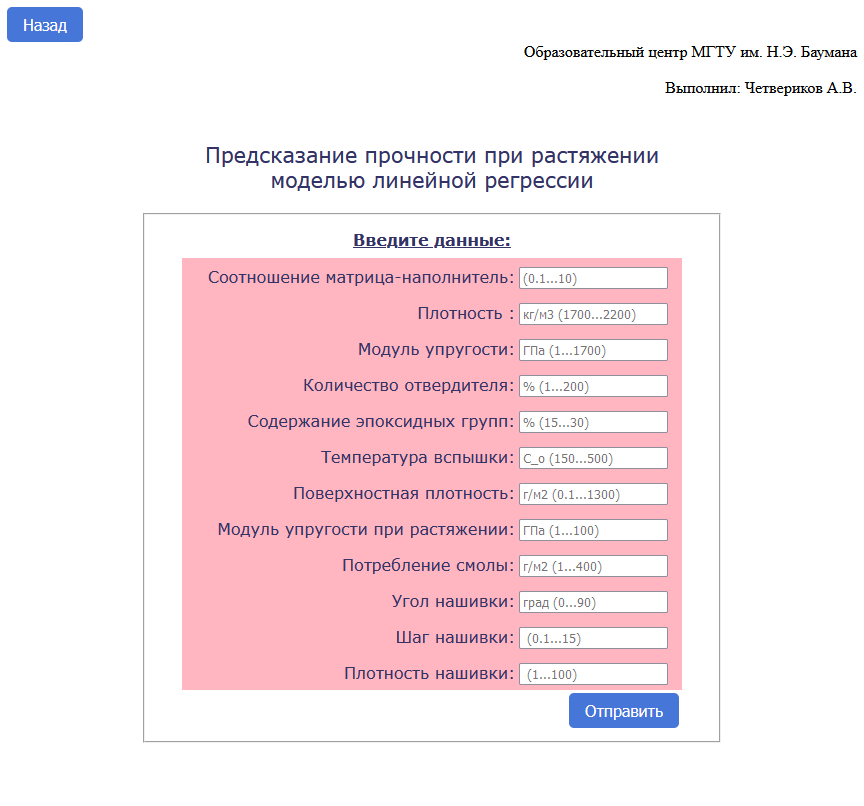
## 2.5 Разработка приложения

Разработанные и обученные модели теперь необходимо как-то применять на практике. Для этих целей было разработано удобное web-приложение с использованием библиотеки Flask. Листинг представлен в прилжении 1. Web-приложение ваполнено по традиционной многостраничное схеме. На стартовой странице (рисунок 52) можно выбрать необходимый тип расчета, в приложении представлены расчеты на основе двух моделей машинного обучения: расчет для рекомендации соотношения матрица-наполнитель выполняется нейросетью, а расчет для предсказания прочности при растяжении выполняется моделью на основе линейной регрессии. Приложение можно запустить локально из среды Python 3.9. Также приложение доступно по адресу: http://avchetverikov.pythonanywhere.com/.

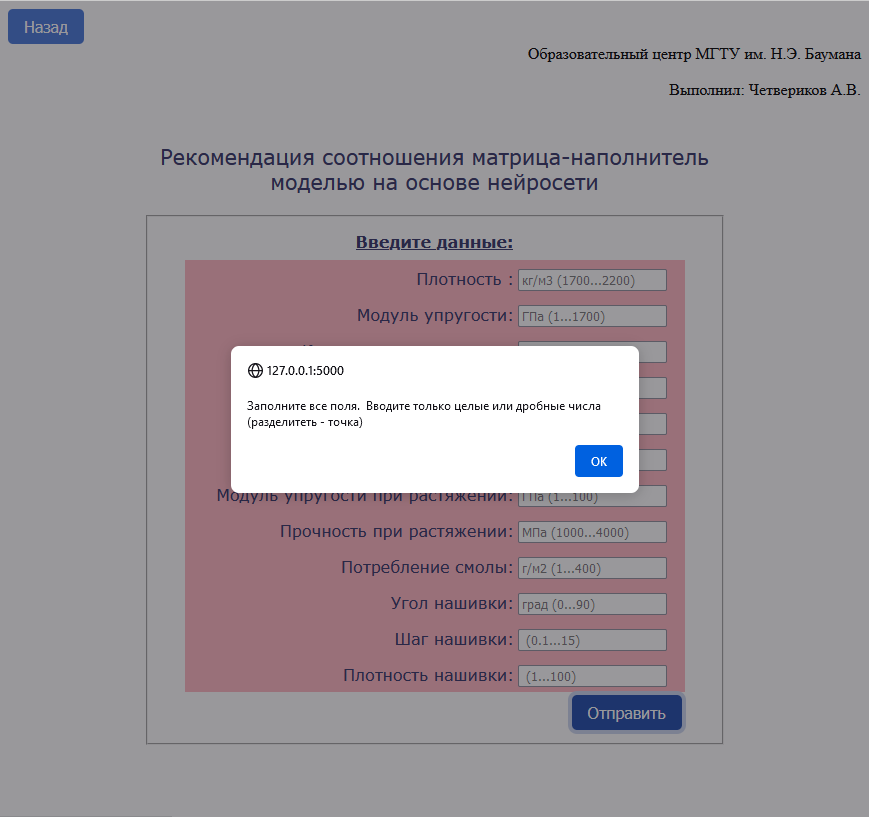
Рисунок 52 – Стартовая страница web-приложения

При выборе типа расчета открывается соответствующая страница (рисунки 53 и 54)

Рисунок 53 – Страница расчета соотношения матрица-наполнитель

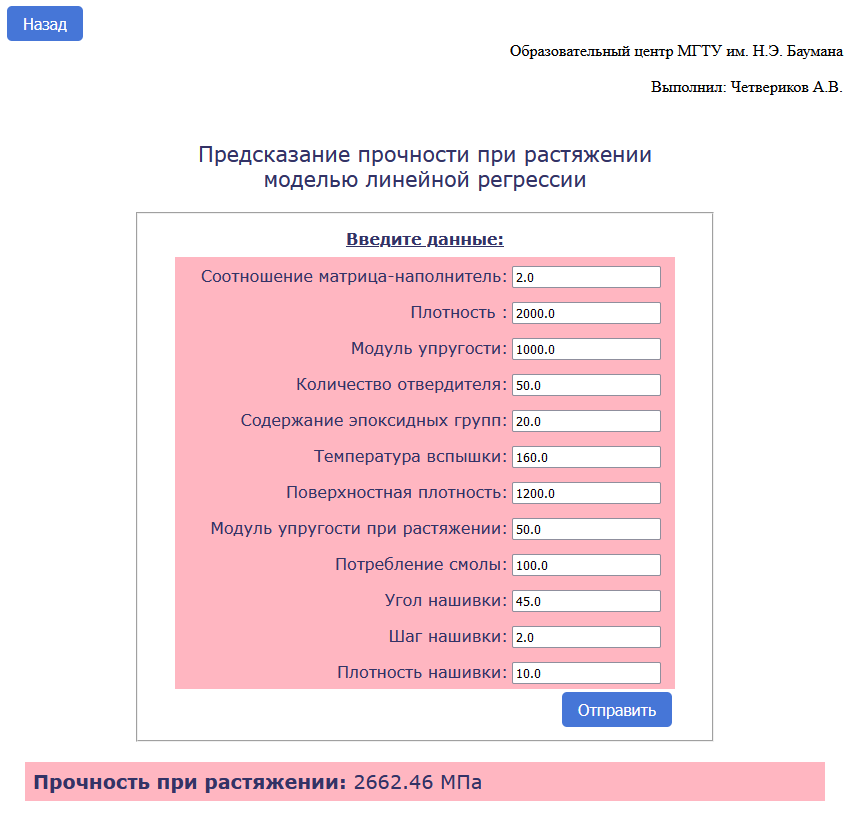
Рисунок 54 – Страница расчета прочности при растяжении

При неправильном вводе исходных данных (допускаются только цифры и разделительный знак – точка) предусмотрена остановка программы расчета и вывод предупреждающего сообщения (рисунок 55)

Рисунок 55 – Предупреждающее сообщение при неправильном вводе

После расчета моделями результата, данные отображаются на экране (рисунок 56 и 57)

Рисунок 56 – Расчитанное значение соотношения матрица-наполнитель

Рисунок 57 – Расчитанное значение прочности при растяжении

## 2.6 Создание удаленного репозитория и загрузка результатов работы

Результаты работы, исходный код приложения доступны в удаленном репозитории: <https://github.com/Artem-Chetverikov/VKR_MGTU_2022>.

# Заключение

# Библиографический список

1. https://habr.com/ru/post/571296/

2. https://scikit-learn.ru/

# Приложение 1

Листинг web-приложения:

# импортируем необходимые библиотеки для нашего приложения

**import** pathlib #для задания относительного пути к файлам

**from** pathlib **import** Path #для задания относительного пути к файлам

**import** pandas **as** pd

**import** numpy **as** np

**import** joblib # для сохранения/загрузки моделей машинного обучения

**from** sklearn.linear\_model **import** LinearRegression

**from** tensorflow **import** keras

**from** sklearn.preprocessing **import** Normalizer, MinMaxScaler, StandardScaler

**from** flask **import** Flask, request, render\_template # для работы приложения

# Определим пути к предварительно подготовленной модели и нормализаторам

dir\_path = pathlib.Path.cwd()

paht\_model\_ann = Path(dir\_path.parents[0], 'domain', 'models\_saved', 'ann', 'recommendation', 'my\_model\_1.h5')

paht\_norm\_y\_ann = Path(dir\_path.parents[0], 'domain', 'models\_saved', 'ann', 'recommendation', 'norm\_minmax\_y.joblib')

paht\_norm\_x\_ann = Path(dir\_path.parents[0], 'domain', 'models\_saved', 'ann', 'recommendation', 'norm\_minmax\_x.joblib')

paht\_model\_lr = Path(dir\_path.parents[0], 'domain', 'models\_saved', 'prediction\_of\_properties', 'lr\_joblib\_model.joblib')

paht\_norm\_y\_lr = Path(dir\_path.parents[0], 'domain', 'models\_saved', 'prediction\_of\_properties', 'stand\_y.joblib')

paht\_norm\_x\_lr = Path(dir\_path.parents[0], 'domain', 'models\_saved', 'prediction\_of\_properties', 'stand\_x.joblib')

# загрузим обученные модели

model\_lr = LinearRegression()

model\_lr = joblib.load(paht\_model\_lr)

model\_ann = keras.models.load\_model(paht\_model\_ann)

# загрузим нормализаторы

norm\_y\_lr = StandardScaler()

norm\_x\_lr = StandardScaler()

norm\_y\_lr = joblib.load(paht\_norm\_y\_lr)

norm\_x\_lr = joblib.load(paht\_norm\_x\_lr)

norm\_y\_ann = MinMaxScaler()

norm\_x\_ann = MinMaxScaler()

norm\_y\_ann = joblib.load(paht\_norm\_y\_ann)

norm\_x\_ann = joblib.load(paht\_norm\_x\_ann)

# наше приложение

app = Flask(\_\_name\_\_, static\_folder="/")

**def** calculate\_recomendation(param):

param = np.array(param)

param = pd.DataFrame(param).T

param = norm\_x\_ann.transform(param) # нормализация входных параметров

prediction = model\_ann.predict(param) # предсказание моделью

prediction = norm\_y\_ann.inverse\_transform(prediction) # денормализация предсказанного значения

**return** prediction

**def** calculate\_prediction(param):

param = np.array(param)

param = pd.DataFrame(param).T

param = norm\_x\_lr.transform(param) # нормализация входных параметров

prediction = model\_lr.predict(param) # предсказание моделью

prediction = pd.DataFrame(prediction)

prediction = norm\_y\_lr.inverse\_transform(prediction) # денормализация предсказанного значения

**return** prediction

@app.route('/', methods=['get', 'post'])

**def** index():

**return** render\_template('index.html')

# страница рекомендации соотношения матрица-наполнитель

@app.route('/calculation\_1', methods=['post', 'get'])

**def** calculation\_1():

param\_lst = []

message = ''

alert = False

**if** request.method == 'POST':

# получим данные из наших форм и кладем их в список, который затем передадим функции calculate\_recomendation

**for** i **in** range(1, 13, 1):

param = request.form.get(f'param{i}')

param\_lst.append(param)

**for** i **in** range(0, 12, 1):

**if** param\_lst[i] == '':

alert = True

**else**:

**try**:

param\_lst[i] = float(param\_lst[i])

**except** ValueError:

alert = True

**break**

**if** alert == False:

message = calculate\_recomendation(param\_lst)

message = round(message[0][0], 4)

**else**:

message = ''

**return** render\_template('calculation\_1.html', message = message, value = param\_lst, alert = alert)

# страница предсказания прочности при растяжении

@app.route('/calculation\_2', methods=['post', 'get'])

**def** calculation\_2():

param\_lst = []

message = ''

alert = False

**if** request.method == 'POST':

# получим данные из наших форм и кладем их в список, который затем передадим функции calculate\_prediction

**for** i **in** range(1, 13, 1):

param = request.form.get(f'param{i}')

param\_lst.append(param)

**for** i **in** range(0, 12, 1):

**if** param\_lst[i] == '':

alert = True

**else**:

**try**:

param\_lst[i] = float(param\_lst[i])

**except** ValueError:

alert = True

**break**

**if** alert == False:

message = calculate\_prediction(param\_lst)

message = round(message[0][0], 4)

**else**:

message = ''

**return** render\_template('calculation\_2.html', message = message, value = param\_lst, alert = alert)

**if** \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

# Приложение 2

прп ККК