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 Царева Наталья Викторовна

Москва, 2023

Содержание

Введение

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).

1. Аналитическая часть

1.1. Постановка задачи

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

Кейс основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

В процессе работы стоят следующие задачи:

1) Необходимо провести разведочный анализ полученных на входе данных с помощью изученных в курсе методов.
2) Данные необходимо предварительно обработать, продемонстрировать этапы обработки и выбранные методы.
3) На выходе необходимо обучить модели машинного обучения и спрогнозировать значения Модуль упругости при растяжении и Прочность при растяжении, а также написать нейронную сеть, которая будет рекомендовать Соотношение матрица-наполнитель. Полученные модели необходимо оценить.
4) Также нужно разработать приложение с графическим интерфейсом или интерфейсом командной строки, которое будет выдавать прогноз на основе обученных моделей.

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

Решаемая задача имеет следующей вид: имеется множество объектов X – элементов выборки, и множество ответов Y – переменных параметров элементов выборки.

В данном случае объектами являются конкретные образцы композиционного материала, а переменными параметрами – их физические свойства, зафиксированные по результатам измерительных испытаний.

Предполагается, что существует функциональная зависимость $f: X \to Y$ между объектами и ответами, но она неизвестна. Известна лишь совокупность пар вида (объект, ответ), называемая выборкой данных. Требуется найти приближенный вид этой f путём построения аппроксимирующей функции. Или другими словами, по известным значениям yi некоторой неизвестной функции в заданных точках xi ($i = 1, \ldots, l$) предсказать значения этой функции в других точках.

Такое прогнозирование конечных свойств новых материалов на основании имеющихся данных испытаний относится к задаче регрессии.

Оценка качества модели машинного обучения, решающей задачу регрессии, заключается в определении величины ошибки ответа, который предсказывает модель, от реального значения показателя. Чем меньше такое отклонение, тем лучше модель.

Существует несколько метрик для определения ошибки модели предсказания. Для оценки качества моделей регрессии в работе будут использованы следующие метрики:

Коэффициент детерминации (R^2) – квадрат коэффициента корреляции, который показывает насколько хорошо регрессионная модель описывает данные. R^2 равный 1, означает что функция идеально ложится на все точки – данные идеально описаны моделью. Метрика помогает понять, какую долю данных модель смогла объяснить.

Средняя абсолютная ошибка (MAE) – сумма модулей разницы между прогнозом и реальным значением для всех объектов деленная на число всех объектов. Преимуществом данной метрики является удобство интерпретации, так как погрешность измеряется в тех же единицах, что и значения целевых переменных. Подходит для данной задачи, так как требуется сравнить разные модели, предсказывающие одно и то же по одинаковым признакам.

1.2. Описание используемых методов

Работа была выполнена на языке Python 3.8 в среде Jupiter Notebook с использованием следующих библиотек:

1)numpy для работы с массивами;

2)pandas для обработки датафреймов;

3)matplotlib для визуализации;

4)seaborn для визуализации;

5)xgboost для построение особой регрессионой модели – деревья решений с градиентным усилением;

6)sklearn для разработки моделей машинного обучения;

7)pickle для сохранения полученных моделей;

8)tensorflow.keras для разработки нейронной сети.

Для решения поставленной задачи – построение моделей машинного обучения для целевых параметров: было применено обучение с учителем с разбиением датасета на тестовую и тренировочную части.

Для переменных Модуль упругости при растяжении и Прочность при растяжении поставленная задача, по сути, сводится к задаче регрессии с предсказанием параметра, поэтому для прогнозирования целевых параметров было решено собрать и протестировать следующие регрессионные модели:

а) 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.

Преимущества:

1)алгоритм прост и легко реализуем;

2)не чувствителен к выбросам;

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

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

Недостатки:

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

2)из аргумента выше следуют большие вычислительные затраты во время выполнения;

3)всегда нужно определять оптимальное значение k.

б) GradientBoostingRegressor

Ансамблевый метод с применением градиентного бустинга.

В случайных лесах каждое дерево в ансамбле строится из выборки, взятой с заменой (то есть выборкой начальной загрузки) из обучающего набора.

Кроме того, при разбиении каждого узла во время построения дерева наилучшее разбиение находится либо по всем входным характеристикам, либо по случайному подмножеству размера max_features.

Назначение этих двух источников случайности — уменьшить дисперсию оценки леса. В самом деле, отдельные деревья решений обычно демонстрируют высокую дисперсию и имеют тенденцию переоснащаться. Внедренная случайность в лесах дает деревья решений с несколько несвязанными ошибками прогнозирования. Если взять среднее значение этих прогнозов, некоторые ошибки могут быть устранены. Случайные леса уменьшают дисперсию за счет комбинирования разных деревьев, иногда за счет небольшого увеличения смещения. На практике уменьшение дисперсии часто бывает значительным, что дает в целом лучшую модель.

в) ExtraTreesRegressor

В данном классе реализован метаоценщик, который подходит к ряду рандомизированных деревьев решений (так же известных как экстра-деревья)на различных подвыборках набора данных и использует усреднение для повышения точности прогнозирования и контроля над подгонкой.

г) RandomForestRegressor

Случайный лес. Суть алгоритма случайного леса состоит в использовании ансамбля решающих деревьев. Из-за большого количества решающих деревьев, результат модели значительно улучшается по сравнению с единичным решающим деревом.

Алгоритм построения случайного леса, состоящего из N деревьев, выглядит следующим образом. Для каждого n = 1, …, N сгенерировать выборку X_n с помощью статистического бутстрэпа, построить решающее дерево b_n по выборке X_n:

1)по заданному критерию выбирается лучший признак, по которому выполняется разбиение в дереве, и так до исчерпания выборки;

2)дерево строится, пока в каждом листе не более n_{min} объектов или пока не будет достигнута определённая высота дерева;

3)при каждом разбиении сначала выбирается m случайных признаков из n исходных, и оптимальное разделение выборки ищется только среди них.

Итоговый классификатор для задачи регрессии выбирает решение по среднему значению.

Преимущества алгоритма:

1)имеет высокую точность предсказания. Не требует тщательной настройки параметров, хорошо работает со значениями по умолчанию.

2)редко переобучается. На практике добавление деревьев только улучшает композицию. В случае наличия проблемы переобучения, она преодолевается путем усреднения или объединения результатов различных деревьев решений.

3)способен эффективно обрабатывать данные с большим числом признаков и классов.

4)одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки

Недостатки алгоритма:

1) для реализации алгоритма случайного дерева требуется значительный объем вычислительных ресурсов. Построение случайного леса отнимает больше времени, чем деревья решений или линейные алгоритмы.

д) 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:

1)простой для понимания и интерпретации. Деревья можно визуализировать;

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

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

4)способен обрабатывать проблемы с несколькими выводами;

5)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;

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

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

К недостаткам деревьев решений относятся:

1)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;

2)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;

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

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

e) LinearRegression

Наиболее распространенный метод.

Линейная регрессия подгоняет линейную модель с коэффицентами $w=(w_1,...,w_p)$ к минимизации остаточной суммы квадрата между наблюдаемым целевоым признаком в наборе данных и предсказанным целевым признаком по линейной аппроксимации.

Преимущества:

1)быстр и прост в реализации; легко интерпретируем;

2)имеет меньшую сложность по сравнению с другими алгоритмами;

Недостатки:

1)моделирует только прямые линейные зависимости;

2)требует прямую связь между зависимыми и независимыми переменными;

- выбросы оказывают огромное влияние, а границы линейны.

ж) 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.

з) Ridge

Преимущество гребенчатой регрессии по сравнению с регрессией наименьших квадратов заключается в компромиссе дисперсии и смещения.

Самым большим преимуществом гребневой регрессии является ее способность давать более низкую среднеквадратичную ошибку теста (MSE) по сравнению с регрессией наименьших квадратов, когда присутствует мультиколлинеарность.

Однако самым большим недостатком гребневой регрессии является ее неспособность выполнять выбор переменных, поскольку она включает все переменные-предикторы в окончательную модель. Поскольку некоторые предикторы будут сжаты очень близко к нулю, это может затруднить интерпретацию результатов модели.

На практике гребенчатая регрессия может создать модель, которая может давать более точные прогнозы по сравнению с моделью наименьших квадратов, но часто бывает сложнее интерпретировать результаты модели.

и) XGBRegressor

Градиентный бустинг – это продвинутый алгоритм машинного обучения для решения задач классификации и регрессии. Он строит предсказание в виде ансамбля слабых предсказывающих моделей, которыми в основном являются деревья решений. Из нескольких слабых моделей в итоге мы собираем одну, но уже эффективную. Общая идея алгоритма – последовательное применение

предиктора (предсказателя) таким образом, что каждая последующая модель сводит ошибку предыдущей к минимуму.

XGBoost – более регуляризованная форма градиентного бустинга. Основным преимуществом данной библиотеки является производительность и эффективная оптимизация вычислений (лучший результат с меньшей затратой ресурсов).

к) 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:

1)возможность изучать нелинейные модели;

2)возможность изучения моделей в режиме реального времени (онлайн-обучение) с использованием partial_fit.

К недостаткам многослойного персептрона (MLP) можно отнести:

1)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;

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

3)MLP чувствителен к масштабированию функций.

В процессе работы планируется пройти по всем моделям, подобрать наиболее подходящий по полученным метрикам, далее применив метод GridSearch, подобрать лучшие параметры и обучить модели.

Для второй поставленной задачи - построение нейронной сети для значения Соотношение матрица-наполнитель использовались стандартные методы сборки модели по слоям, во многом основанные на интуитивном подборе слоев и количества нейронов.

Гибкость нейронных сетей является так же одним из главных недостатков: существует много гиперпараметров для подстройки. Мало того, что можно выбрать любую вообразимую топологию сети, так же можно провести отбор функции активации, логику инициализации слоев и так далее. Для облегчения задачи можно воспользоваться рандомизированным поиском для подбора гиперпараметров, но в данной работе попробуем ручной подбор параметров.

Несомненно, одним из важнейших гиперпараметров является функция активации. Функция активации a(x) определяет выходное значение нейрона в зависимости от результата взвешенной суммы входов и порогового значения.

В большинстве случаев, в скрытых слоях для задачи регрессии применяется функция активации ReLU (или один из ее вариантов), так как она намного быстрее в вычислениях и не насыщается как другие функции. В случае задачи регрессии на выходном слое можно вообще не применять функцию активации, что будет проверено в этой работе.

Rectified Linear Unit или ReLU (рисунок 1) возвращает 0, если принимает отрицательный аргумент, в случае же положительного аргумента, функция возвращает само число. То есть она может быть записана как указано в формуле 1.

$$f(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Формула 1 – Функция ReLU

На первый взгляд может показаться, что она линейна и имеет те же проблемы что и линейная функция, но это не так и ее можно использовать в нейронных сетях с множеством слоев. Функция ReLU обладает несколькими преимущества перед сигмоидой и гиперболическим тангенсом:

1)Очень быстро и просто считается производная. Для отрицательных значений — 0, для положительных — 1.

2)Разреженность активации. В сетях с очень большим количеством нейронов использование сигмоидной функции или гиперболического тангенса в качестве активационный функции влечет активацию почти всех нейронов, что может сказаться на производительности обучения модели. Если же использовать ReLU, то количество включаемых нейронов станет меньше, в силу характеристик функции, и сама сеть станет легче.

У данной функции есть один недостаток, называющийся проблемой умирающего ReLU·. Так как часть производной функции равна нулю, то и градиент для нее будет нулевым, а то это значит, что веса не будут изменяться во время спуска и нейронная сеть перестанет обучаться.

Функцию активации ReLU следует использовать, если нет особых требований для выходного значения нейрона, вроде неограниченной области определения. Но если после обучения модели результаты получились не оптимальные, то стоит перейти к другим функциям, которые могут дать лучший результат.

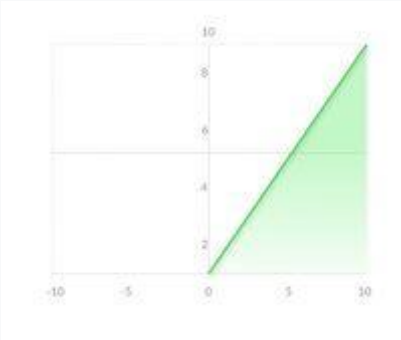


Рисунок 1 - Функция ReLU

Одной из проблем стандартного ReLU является затухающий, а именно нулевой, градиент при отрицательных значениях. При использовании обычного ReLU некоторые нейроны умирают, а отследить умирание нейронов не просто. Чтобы решить эту проблему иногда используется подход ReLU с «утечкой» (график функции активации на отрицательных значениях образует не горизонтальную прямую, а наклонную, с маленьким угловым коэффициентом (порядка 0,01) (Рисунок 2).

Однако, функция Leaky ReLU имеет некоторые недостатки:

1) Сложнее считать производную, по сравнению со стандартным подходом (так как значения уже не равны нулю), что замедляет работу каждой эпохи.
2) Угловой коэффициент прямой также является гиперпараметром, который надо настраивать.
3) На практике, результат не всегда сильно улучшается относительно ReLU.

На практике LReLU используется не так часто. Практический результат использования LReLU вместо ReLU отличается не слишком сильно.

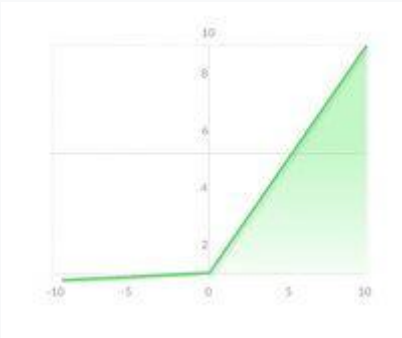


Рисунок 2 - Функция Leaky ReLU

Так же, одной из проблем глубокого обучения нейронных сетей является переобучение. И метод дропаут — популярный способ решения этой проблемы, благодаря простому алгоритму и хорошим практическим результатам.

В обычной нейронной сети явление переобучения появляется из-за так называемой совместной адаптации, то есть при обновлении весов нейрона, во

время обучения методом обратного распространения ошибки, учитывается деятельность остальных нейронов с целью минимизировать функцию потерь (ошибки). Поэтому веса нейронов могут меняться, исправляя при этом ошибки других нейронов. Метод дропаута как раз предотвращает эту адаптацию.

В глубоком обучении оценка моделей нейронных сетей происходит по функции ошибки или потерь. Цель обучения – минимизировать ошибку при обучении, ошибка – это отклонение предсказанного значения от реального. В целях минимизации ошибки нейронная сеть учиться подбирать веса. В задаче регрессии при прогнозировании непрерывной переменной цель минимизации ошибки заключается в снижении разрыва между спрогнозированным и реальным значением. В данной работе для оценки моделей используется средняя абсолютная ошибка MAE.

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; тепловая карта взаимной корелляции признаков; проверка наличия пропусков и дубликатов.

В данном разделе были использованы методы, реализованные в библиотеке pandas, которые позволяют провести разведочный анализ данных, а именно:

1).info() общая информация по датасету;

2).shape структура датасета;

3).describe() описательная статистика датасета;

4).isnull() выявление пропусков;

5).dublicated() выявление дублей в строках;

6).corr() проверка корреляции.

Библиотеки seaborn и matplotlib - для визуализации данных, построения парных графиков, гистограмм, графиков корреляции и тд.

Так же с помощью метода SelectKBest из библиотеки Sklearn была проведена попытка ранжирования признаков по важности для каждой целевой переменной. В дальнейшем это можно применить для создания более сложных моделей машинного обучения, что не было реализовано в данной работе.

Опишем основные этапы работы.

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.

Объединённая электронная таблица (исходный набор данных для дальнейшей работы) содержит информацию о 13 параметрах для 1023 элементов. Пропусков в исходном наборе данных не выявлено. Исходный набор данных определён в коде как «df».

Первичный анализа датасета дает информацию по типу данных, их количеству, количеству столбцов. (Рисунок 3)

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1023 entries, 0 to 1022
Data columns (total 13 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   Соотношение матрица-наполнитель        1023 non-null   float64
 1   Плотность, кг/м3                       1023 non-null   float64
 2   модуль упругости, ГПа                  1023 non-null   float64
 3   Количество отвердителя, м.%            1023 non-null   float64
 4   Содержание эпоксидных групп,%_2        1023 non-null   float64
 5   Температура вспышки, С_2               1023 non-null   float64
 6   Поверхностная плотность, г/м2          1023 non-null   float64
 7   Модуль упругости при растяжении, ГПа   1023 non-null   float64
 8   Прочность при растяжении, МПа          1023 non-null   float64
 9   Потребление смолы, г/м2                1023 non-null   float64
 10  Угол нашивки, град                     1023 non-null   int64  
 11  Шаг нашивки                            1023 non-null   float64
 12  Плотность нашивки                      1023 non-null   float64
dtypes: float64(12), int64(1)
```

Рисунок 3 – Применение метода df.info()

Применение метода .nunique() позволяет определить количество уникальных значений для каждой переменной, а так же выявить

категориальную переменную с двумя вариантами значений «Угол нашивки, град», с которой в дальнейшем надо провести работу (Рисунок 4).

```
df['Угол нашивки, град'].unique() #выявлена категориальная переменная
```

```
array([ 0, 90], dtype=int64)
```

Рисунок 4 – Категориальная переменная Угол нашивки, град.

Далее проведем описательную статистику, определяем среднее значение, минимуму и максимумы, для понимания распределения в данных (Рисунок 5).

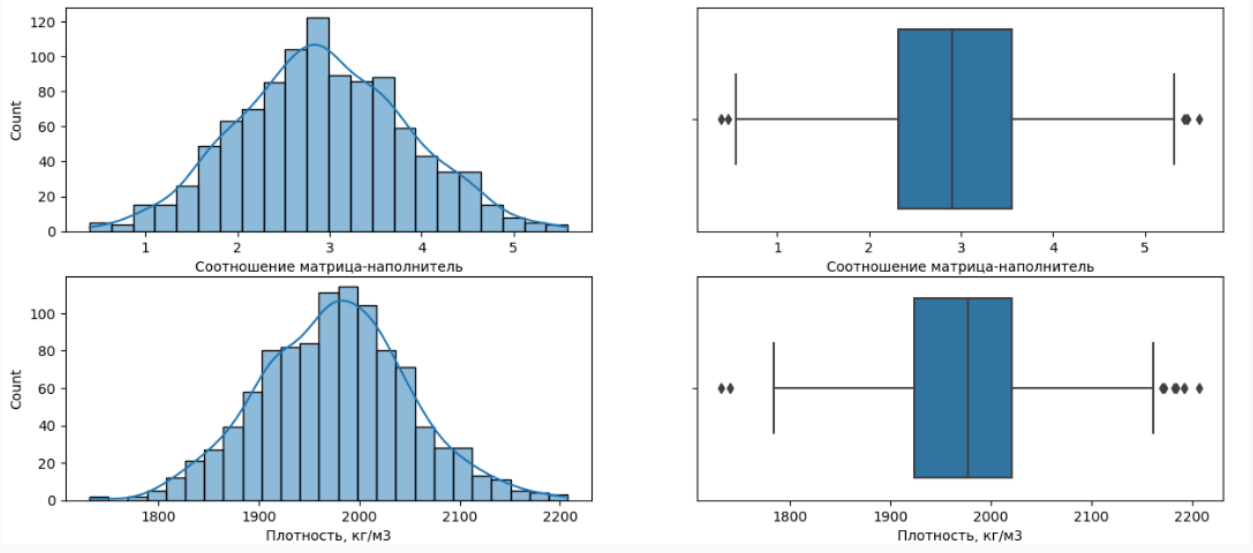
```
df.describe().T#общая описательная статистика
```

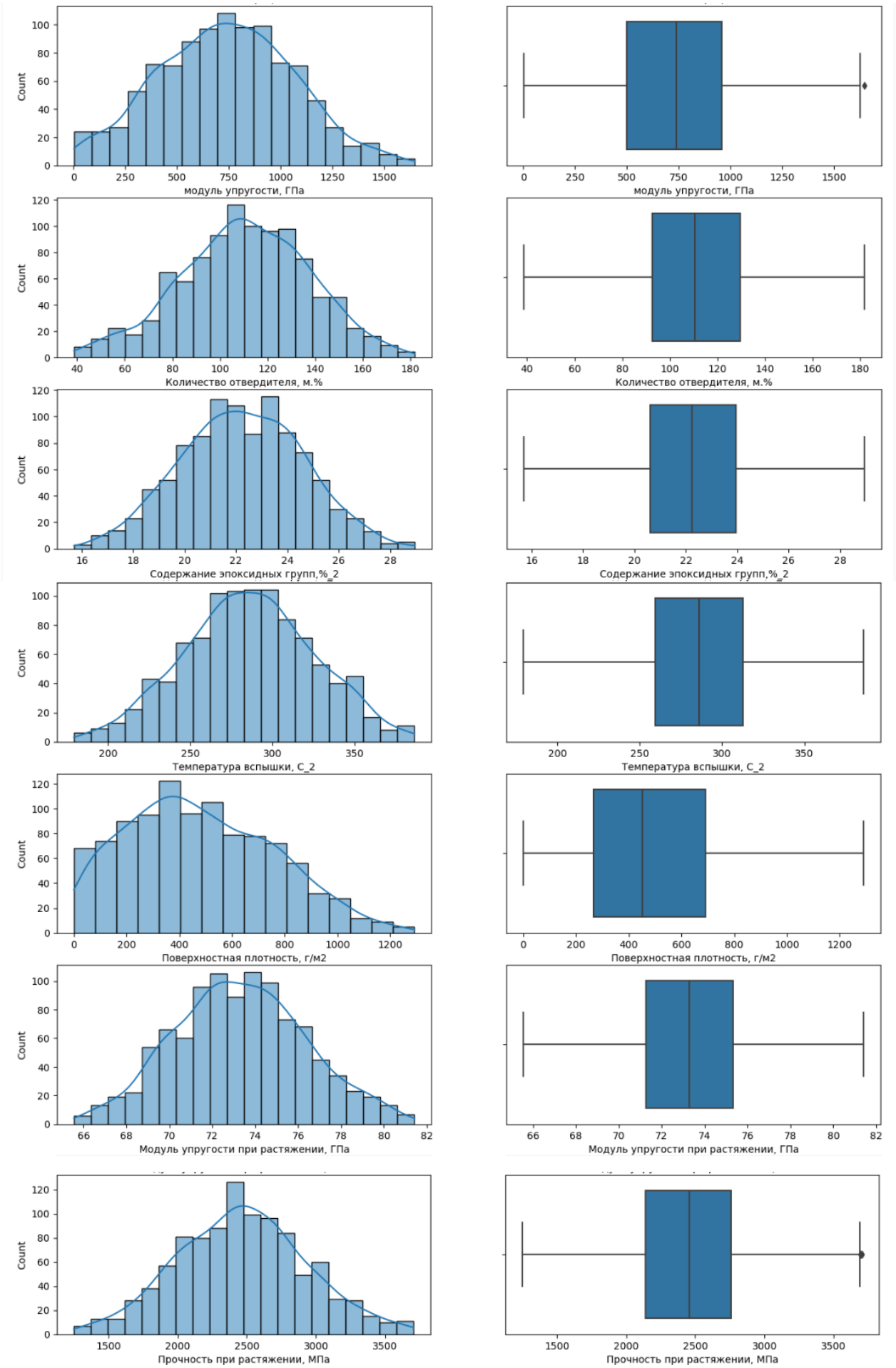
	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Рисунок 5 – Описательная статистика метод .describe()

Предварительно можно оценить, что данные имеют нормальное распределение и, скорее всего, были предварительно обработаны.

Далее строим графики для визуальной статической оценки (Рисунок 6).




Count
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

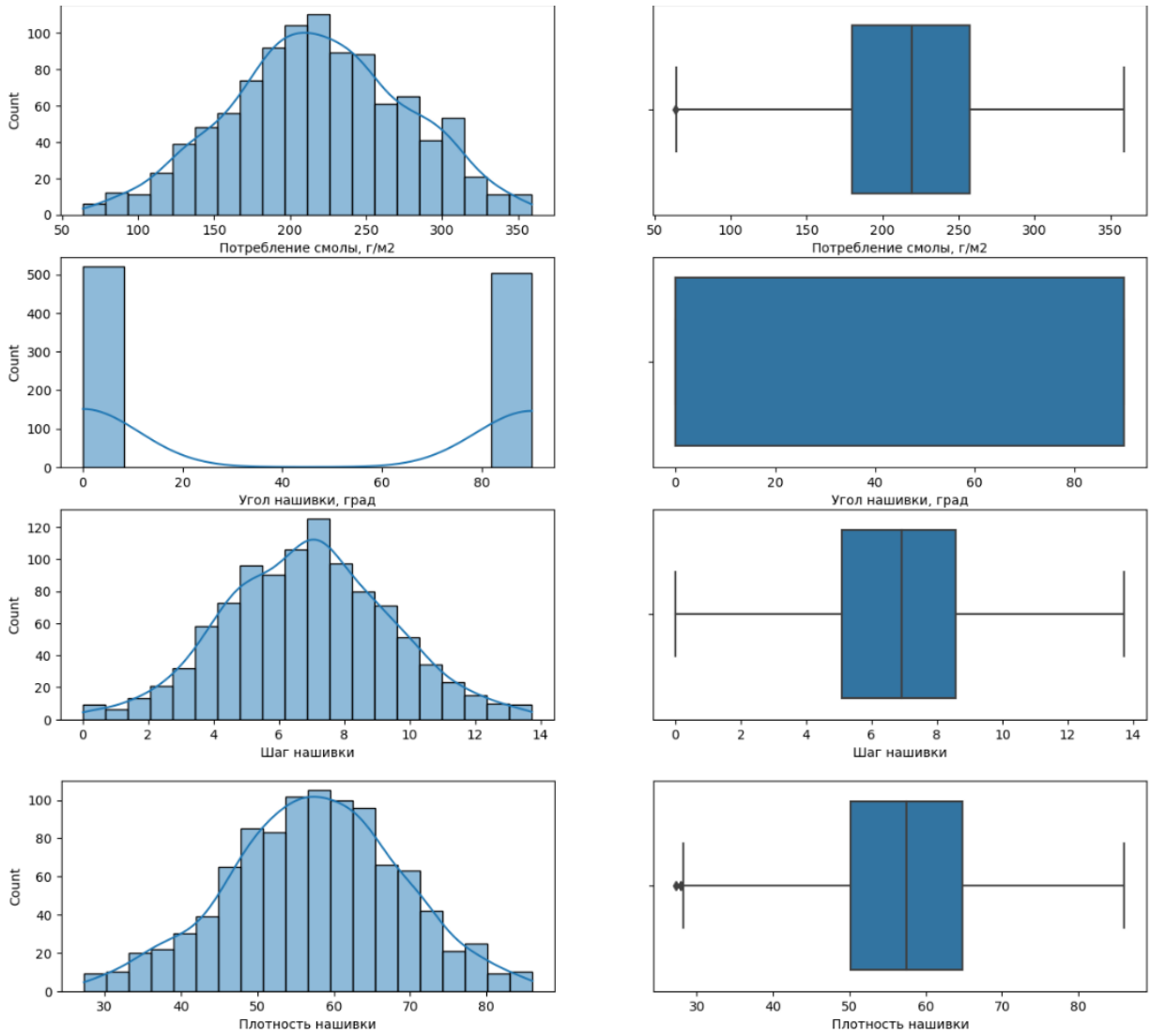


Рисунок 6 – Гистограммы распределения для каждого параметра.

Построение гистограмм показывает, что данные имеют нормальное распределение, а так же имеют выбросы, которые могут негативность сказаться на работе моделей машинного обучения, поэтому выбросы необходимо будет удалить.

Далее была проведена оценка корреляции между параметрами с помощью стандартного метода .corr() библиотеки pandas и визуализации из библиотеки seaborn (рисунок 7).

В процессе применения данных методов значительной корреляции между переменными, а так же с целевыми переменными выявлено не было. В данном случае для корректной отработки моделей машинного обучения лучше провести обогащение данных, проработав вопрос с экспертами, либо более

сложными методами попробовать отработать значение каждого параметра для целевых переменных.

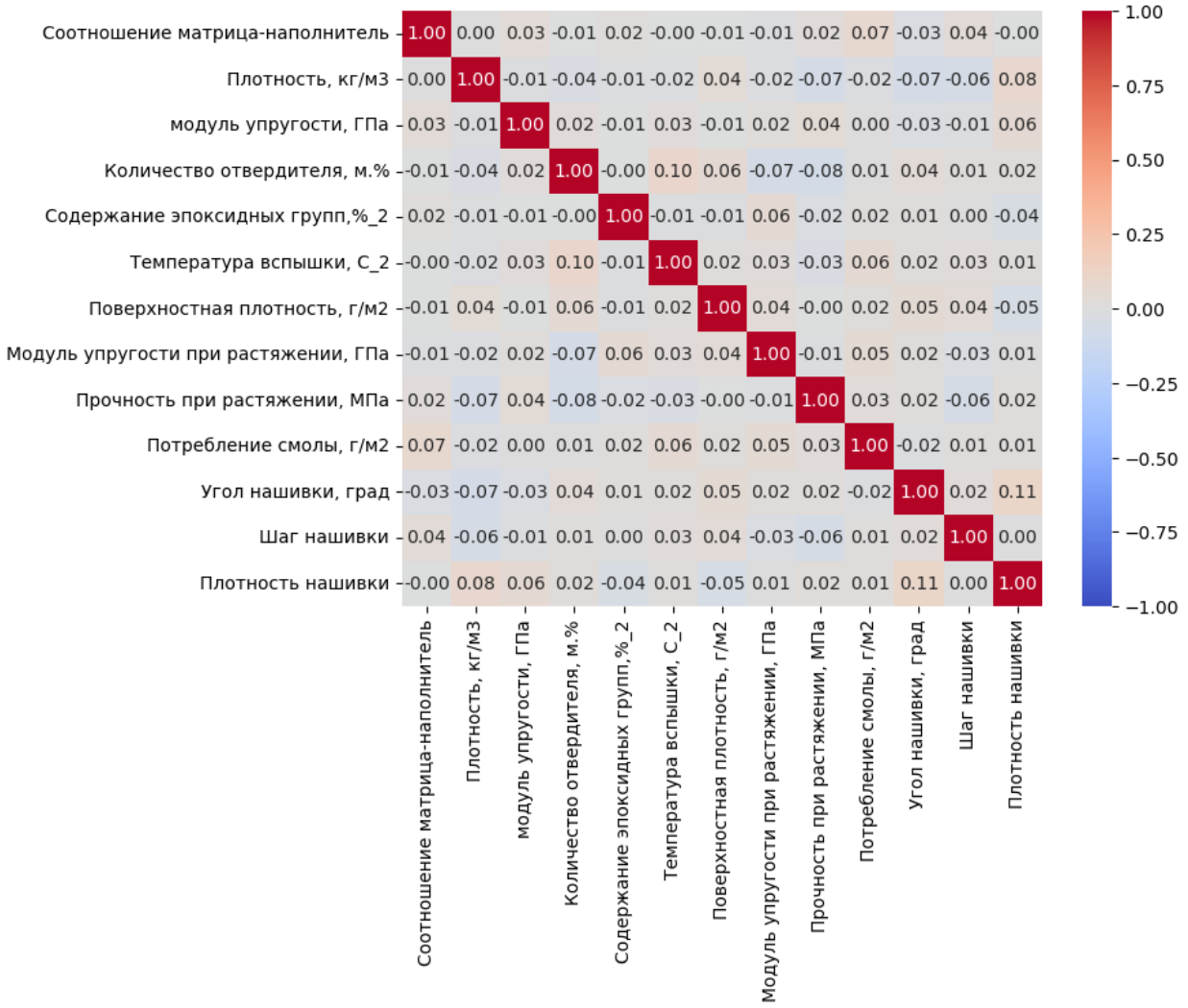


Рисунок 7 – Оценка корреляции параметров датасета.

Попарный график рассеивания так же не показал значительной корреляции между параметрами.

В попытке выявить скрытую взаимосвязь был применен метод скоринга SelectKBest для регрессионных задач из библиотеки sklearn.feature_selection (рисунок 8). Были получены следующие результаты. Для дальнейшей работы с ними необходимо экспертное мнение, возможно провести перегруппировку параметров или провести групповую подачу данных в модели.

```
Важность признаков для Модуль упругости при растяжении, ГПа:
                               Specs     Score
3       Количество отвердителя, м.%  4.457238
4   Содержание эпоксидных групп,%_2  3.307961
8           Потребление смолы, г/м2  2.656047
6     Поверхностная плотность, г/м2  1.377153
10                     Шаг нашивки  0.887374
5          Температура вспышки, С_2  0.824993
2             модуль упругости, ГПа  0.553027
9                Угол нашивки, град  0.540559
1                  Плотность, кг/м3  0.316430
7     Прочность при растяжении, МПа  0.082870
0   Соотношение матрица-наполнитель  0.072228
11                Плотность нашивки  0.042818

Важность признаков для Прочность при растяжении, МПа, ГПа:
                                 Specs     Score
3            Количество отвердителя, м.%  5.833834
1                      Плотность, кг/м3  5.024804
10                          Шаг нашивки  3.633195
2                 модуль упругости, ГПа  1.792851
5              Температура вспышки, С_2  1.031125
8               Потребление смолы, г/м2  0.835932
0        Соотношение матрица-наполнитель  0.595726
4        Содержание эпоксидных групп,%_2  0.583473
9                    Угол нашивки, град  0.559256
11                    Плотность нашивки  0.392536
7   Модуль упругости при растяжении, ГПа  0.082870
6          Поверхностная плотность, г/м2  0.010519

Важность признаков для Соотношение матрица-наполнитель, МПа, ГПа:
                                 Specs     Score
8               Потребление смолы, г/м2  5.399606
10                          Шаг нашивки  1.357330
1                 модуль упругости, ГПа  1.026992
9                    Угол нашивки, град  0.986751
7          Прочность при растяжении, МПа  0.595726
3        Содержание эпоксидных групп,%_2  0.399046
6   Модуль упругости при растяжении, ГПа  0.072228
2            Количество отвердителя, м.%  0.042409
5          Поверхностная плотность, г/м2  0.040161
4               Температура вспышки, С_2  0.023293
11                    Плотность нашивки  0.022094
0                      Плотность, кг/м3  0.015066
```

Рисунок 8 – Скоринг параметров методом SelectKBest

В результате разведочного анализа данных было определено наличие выбросов, предположение, что данные уже были нормализованы, не удалось выявить сколько бы значимой зависимости с целевыми переменными.

2. Практическая часть

2.1. Предобработка данных

В первую очередь, выполним очистку данных от выбросов.

Выбросы – это данные, которые существенно отличаются от других наблюдений. Они могут соответствовать реальным отклонениям, но могут быть и просто ошибками. Но большая часть моделей машинного обучения плохо обучается при наличии выбросов.

Первично выбросы были удалены с помощью метода трех сигм, основанном на расчете среднего значения параметра и среднего отклонения. Метод был применен дважды, однако не принес удовлетворительного результата, что отражено в коде.

Повторное удаление выбросов выполнено с помощью межквартильных диапазонов

IQR = Quartile3 – Quartile1

Для выявления выбросов для каждого параметра определяются значения, находящиеся выше и ниже нормального диапазона наборов данных, т.е. за пределами верхней и нижней границы:

- верхняя граница = Q3+1.5*IQR;

- нижняя граница = Q1-1.5*IQR.

Данный метод дал удовлетворительный результат.

Повторное построение графиков бокс-плотов показывает отсутствие выбросов (рисунок 9).

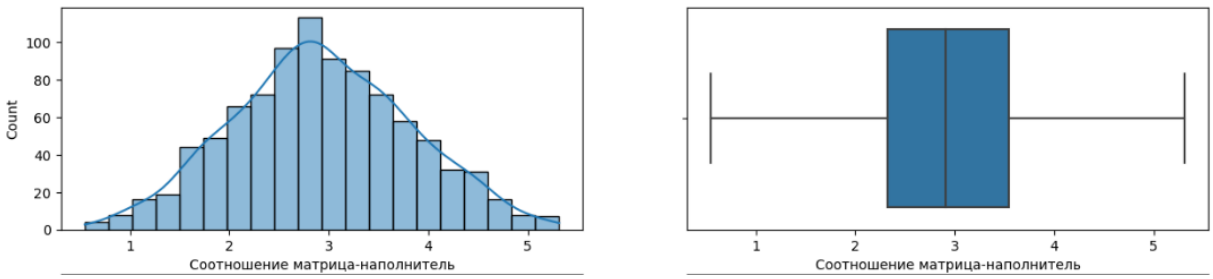


Рисунок 9 – Выбросы после проведения очистки (пример на одном параметре).

Очищенный от выбросов дата сет берем в дальнейшую работу под названием «df_pred1». Количество строк в новом датасете 936 (рисунок 10).

```
#примененим очиещнный методом межквартильного интервала датасет для дальнейшей работы,
#так как метод показал бОльшую эффективность в решении вопроса удаления выбросов
#переобозначим переменную
df_pred1=df_cleared3
df_pred1.shape
```

```
(936, 13)
```

Рисунок 10 – Информация по очищенному от выбросов датасету

Так же на этапе предобработки данных проводим кодировку категориальной переменной «Угол нашивки, град»(рисунок 11).

```
encoder=LabelEncoder()
df_pred1['Угол нашивки, град'] =encoder.fit_transform(df_pred1['Угол нашивки, град'])

df_pred1['Угол нашивки, град']
```

Рисунок 11 – Кодировка категориальной переменной

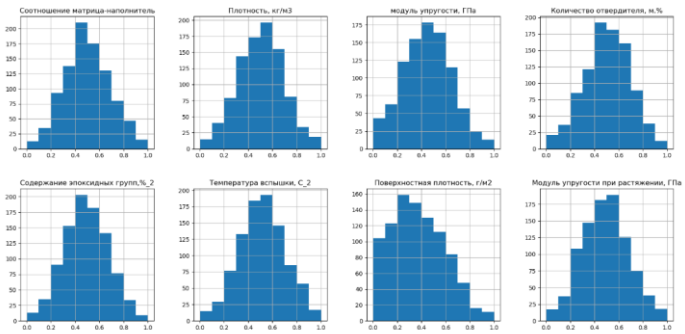
Хотя предположительно данные были нормализованы, подготовим два датасета – нормализованный с помощью метода MinMaxScaler для подачи в модели машинного обучения. Данные будут сохранены в наборе «df_norm».

Из каждого значения признака, MinMaxScaler вычитает минимальное значение признака и делит на диапазон. Диапазон это разница между минимумом и максимумом этого признака.

MinMaxScaler сохраняет форму исходного распределения. Он не меняет содержательно информацию, хранящуюся в исходных данных.

Диапазон по умолчанию для признака после MinMaxScaler находится между 0 и 1.

В результате нормализации получены следующие графики распределения данных (рисунок 12)



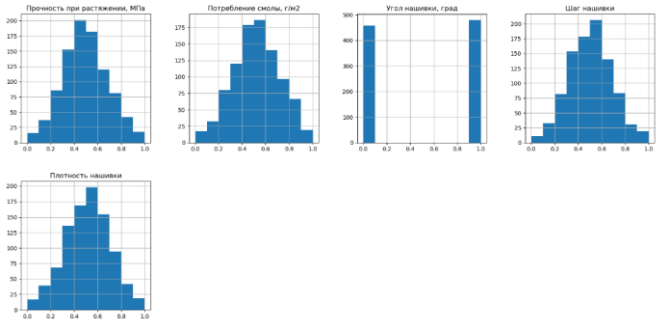


Рисунок 12 – Визуализация прошедшего нормализацию набора данных

Описательная статистика полученного в результате обработки датасета приведена ниже на рисунке 13.

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	936.0	0.498933	0.187489	0.0	0.372274	0.494538	0.629204	1.0
Плотность, кг/м3	936.0	0.502695	0.187779	0.0	0.368517	0.511229	0.624999	1.0
модуль упругости, ГПа	936.0	0.446764	0.199583	0.0	0.301243	0.447061	0.580446	1.0
Количество отвердителя, м.%	936.0	0.504664	0.188865	0.0	0.376190	0.506040	0.637978	1.0
Содержание эпоксидных групп,%_2	936.0	0.491216	0.180620	0.0	0.367716	0.489382	0.623410	1.0
Температура вспышки, С_2	936.0	0.516059	0.190624	0.0	0.386128	0.515980	0.646450	1.0
Поверхностная плотность, г/м2	936.0	0.373733	0.217078	0.0	0.205619	0.354161	0.538683	1.0
Модуль упругости при растяжении, ГПа	936.0	0.488647	0.191466	0.0	0.359024	0.485754	0.615077	1.0
Прочность при растяжении, МПа	936.0	0.495706	0.188915	0.0	0.365149	0.491825	0.612874	1.0
Потребление смолы, г/м2	936.0	0.521141	0.195781	0.0	0.392067	0.523766	0.652447	1.0
Угол нашивки, град	936.0	0.511752	0.500129	0.0	0.000000	1.000000	1.000000	1.0
Шаг нашивки	936.0	0.502232	0.183258	0.0	0.372211	0.504258	0.624604	1.0
Плотность нашивки	936.0	0.513776	0.191342	0.0	0.390482	0.516029	0.638842	1.0

Рисунок 13 – Описательная статистика нормализованного набора данных.

Так же сделаем еще один вариант датасета, прошедшего стандартизацию (рисунок 14, 15), и попробуем подать в модели оба варианта, чтобы проверить на каких данных лучше сходятся модели. Данный датасет будет работать под названием «df_std».

Стандартизацию проведем методом StandartScaler. StandardScaler стандартизирует признак вычитая среднее и затем масштабируя к единчной дисперсии. Единичная дисперсия значит деление все значений на стандартное отклонение.

В результате применения StandardScaler мы получаем распределение со стандартным отклонением равным 1. Дисперсия так же равна 1, так как дисперсия = квадрату стандартного отклонения. А 1 в квадрате = 1.

StandardScaler делает среднее распределения равным 0.

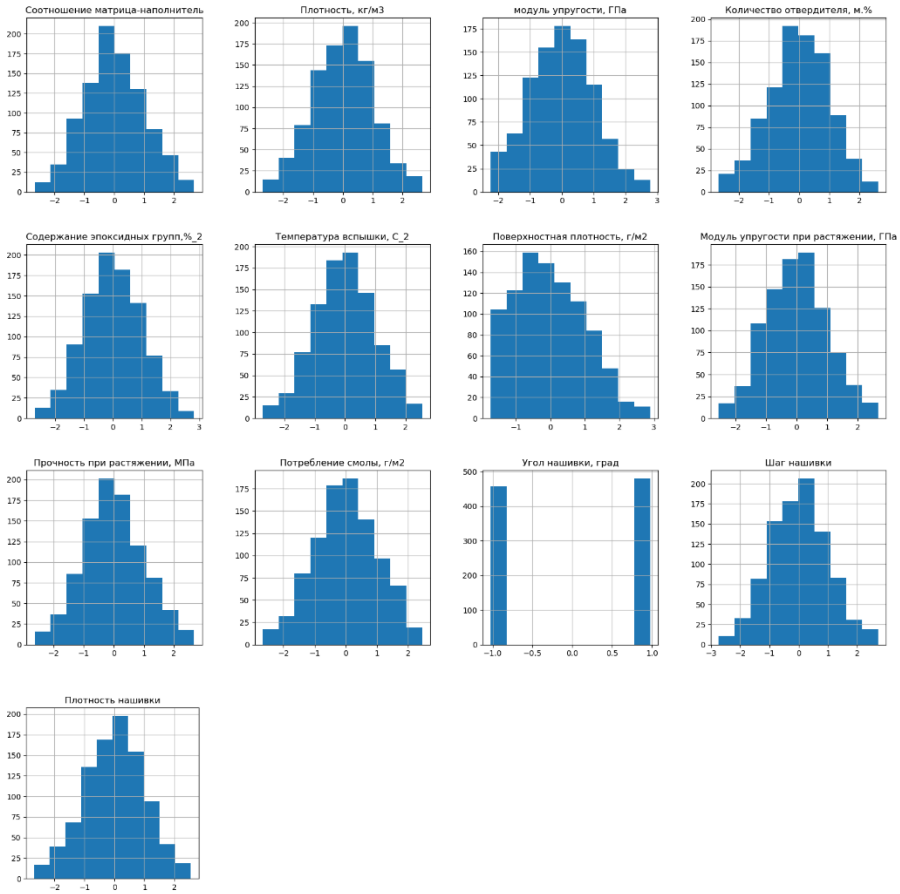


Рисунок 14 - Визуализация стандартизированного набора данных

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	936.0	-3.985416e-16	1.000535	-2.662564	-0.675917	-0.023457	0.695189	2.673947
Плотность, кг/м3	936.0	-1.040004e-15	1.000535	-2.678494	-0.714937	0.045471	0.651668	2.649773
модуль упругости, ГПа	936.0	1.480297e-16	1.000535	-2.239686	-0.729517	0.001489	0.670165	2.773444
Количество отвердителя, м.%	936.0	-4.934325e-17	1.000535	-2.673520	-0.680609	0.007288	0.706247	2.624101
Содержание эпоксидных групп,%_2	936.0	-8.264994e-16	1.000535	-2.721073	-0.684125	-0.010160	0.732284	2.818386
Температура вспышки, С_2	936.0	2.827748e-16	1.000535	-2.708660	-0.681975	-0.000414	0.684390	2.540085
Поверхностная плотность, г/м2	936.0	1.537232e-16	1.000535	-1.722572	-0.774851	-0.090207	0.760272	2.886532
Модуль упругости при растяжении, ГПа	936.0	-4.531987e-15	1.000535	-2.553495	-0.677361	-0.015119	0.660679	2.672150
Прочность при растяжении, МПа	936.0	7.591269e-16	1.000535	-2.625366	-0.691456	-0.020555	0.620548	2.670853
Потребление смолы, г/м2	936.0	-1.622634e-16	1.000535	-2.663276	-0.659631	0.013415	0.671035	2.447193
Угол нашивки, град	936.0	1.214603e-16	1.000535	-1.023787	-1.023787	0.976766	0.976766	0.976766
Шаг нашивки	936.0	-2.068621e-16	1.000535	-2.742040	-0.709873	0.011064	0.668120	2.717671
Плотность нашивки	936.0	-1.821904e-16	1.000535	-2.686557	-0.644710	0.011780	0.653975	2.542482

Рисунок 15 – Описательная статистика стандартизированного датасета

Полученные обработанные датасеты под названиями df_norm и df_std будут поданы для работы в модели машинного обучения для предсказания целевых параметров.

2.2. Разработка и обучение модели для параметра «Модуль упругости при растяжении, ГПа»

Для обучения и проверки качества моделей, каждый обработанный набор данных будет разделён на обучающую и тестовую выборки объёмом 70% и 30% соответственно с помощью метода train_test_split библиотеки sklearn.

В целях оптимизации работы был собран список регрессионных моделей, объединенных в список под названием regressors. Далее по циклу подаем данные в каждую модель и по метрике R2 выбираем оптимальные 4 модели для каждого целевого показателя для дальнейшей работы (рисунок 16).

```
regressors = [
    KNeighborsRegressor(),
    GradientBoostingRegressor(),
    KNeighborsRegressor(),
    ExtraTreesRegressor(),
    RandomForestRegressor(),
    DecisionTreeRegressor(),
    LinearRegression(),
    Lasso(),
    Ridge(),
    XGBRegressor(),
    MLPRegressor()
]
cv = KFold(10, shuffle=True)


for model in regressors:
    model.fit(X1_train, y1_train)
    y1_pred = model.predict(X1_test)
    print(model)
    print("\tExplained variance:", explained_variance_score(y1_test, y1_pred))
    print("\tMean absolute error:", mean_absolute_error(y1_test, y1_pred))
    print("\tR2 score:", r2_score(y1_test, y1_pred))
    print()
```

Рисунок 16 – Код обхода регрессионных моделей для параметра «Модуль упругости при растяжении, гПа»

В процессе обхода так же подаем в модели данные как стандартизированного, так и нормализованного датасетов. По результатам видно, что df_norm показывает лучшие результаты, в дальнейшем работа будет идти только с этим набором данных.

Модель	R2 score, df_norm	R2 score, df_std
KNeighborsRegressor()	-0.2473177674515108	-0.3052044617089842
GradientBoostingRegressor()	-0.07473180049718531	-0.07572035751956108
ExtraTreesRegressor()	-0.05811323098584009	-0.03813786621800386
RandomForestRegressor()	-0.03142425874735899	-0.021003032359695917
DecisionTreeRegressor()	-1.0838006517070262	-1.1107293072841764
LinearRegression()	0.005942777416883649	0.005942777416883427
Lasso()	-0.001437885728528876	-0.001437885728529098
Ridge()	0.006040279246332103	0.005943985423210396
XGBRegressor()	-0.3306237599604134	-0.22799027930954696
MLPRegressor()	-0.016353210015360053	-0.17124796812711263

Таблица 1. Регрессиионые модели и их результат оценки для параметра Модуль упругости при растяжении

Для параметра Модуль упругости при растяжении, гПа прошли отбор следующие модели: LinearRegression, Ridge, Lasso, RandomForestRegressor (таблица 1)

По этим моделям проводим подбор гиперпараметров с помощью метода GridSearch. Результаты и визуализация результатов для каждой модели по данному параметру представлена в файле с кодом Python.

2.3. Разработка и обучение модели для параметра «Прочность при растяжении, мПа»

Аналогичным способом, как и в п. 2.2. проведена разбивка датасета df-norm на обучающую и тестовую выборки в соотношении 70 на 30.

В целях оптимизации работы был собран список регрессионных моделей, объединенных в список под названием regressors. Далее по циклу подаем данные в каждую модель и по метрике R2 выбираем оптимальные 4 модели для каждого целевого показателя для дальнейшей работы (рисунок 18).

```
#список возможных регрессоров
regressors = [
    KNeighborsRegressor(),
    GradientBoostingRegressor(),
    ExtraTreesRegressor(),
    RandomForestRegressor(),
    DecisionTreeRegressor(),
    LinearRegression(),
    Lasso(),
    Ridge(),
    XGBRegressor(),
    MLPRegressor()
]
cv = KFold(10, shuffle=True)


for model in regressors:
    model.fit(X2_train, y2_train)
    y2_pred = model.predict(X2_test)
    print(model)
    print("\tExplained variance:", explained_variance_score(y2_test, y2_pred))
    print("\tMean absolute error:", mean_absolute_error(y2_test, y2_pred))
    print("\tR2 score:", r2_score(y2_test, y2_pred))
    print()
```

Рисунок 18 – Код обхода регрессионных моделей для параметра «Прочность при растяжении, мПа»

Модель	R2 score, df_norm
KNeighborsRegressor()	-0.18942316371677514
GradientBoostingRegressor()	-0.08738726979232947
ExtraTreesRegressor()	-0.04833097135269093
RandomForestRegressor()	-0.007120120066017588

DecisionTreeRegressor()	-0.9375791854598725
LinearRegression()	0.0014984945769408453
Lasso()	-0.005565272873978788
Ridge()	0.0018620444709597228
XGBRegressor()	-0.20826219034627447
MLPRegressor()	-0.19310357373801068

Таблица 2 – Регрессионные модели и их результат оценки для параметра Прочность при растяжении.

Для параметра Модуль упругости при растяжении, гПа прошли отбор следующие модели: LinearRegression, Ridge, Lasso, RandomForestRegressor, MLPRegressor (таблица 2).

По этим моделям проводим подбор гиперпараметров с помощью метода GridSearch. Результаты и визуализация результатов для каждой модели по данному параметру представлена в файле с кодом Python.

2.4. Тестирование моделей для параметра «Модуль упругости при растяжении, ГПа»

В результате тестирования полученных моделей получаем следующие результаты, объединённые в таблицу (рисунок 17):

	Model	Explained variance	MAE	R2 score
Модуль упругости при растяжении,гПа	LinearRegression	0.007377	0.146740	0.006
Модуль упругости при растяжении,гПа	RidgeRegression	0.005644	0.146948	0.004
Модуль упругости при растяжении,гПа	LassoRegression	0.000000	0.147274	-0.001
Модуль упругости при растяжении,гПа	RandomForestRegressor	-0.008845	0.147400	-0.010

Рисунок 17 – Результаты работы регрессионных моделей с подбором гиперпараметров по сетке для параметра Модуль упругости при растяжении, Гпа.

Лучшие результаты (из сравниваемых) получила модель Линейной регрессии со следующими параметрами: 'fit_intercept': 'True', который является булевым параметром, определяющим вычислять или нет пересечение (если не вычислять – значение False).

Удовлетворительных результатов по прогнозированию на текущем этапе не удалось достичь.

2.5. Тестирование моделей для параметра «Прочность при растяжении, мПа»

В результате тестирования отобранных моделей с настроенным гиперпараметрами, получаем следующие результаты (рисунок 18):

	Model	Explained variance	MAE	R2 score
Прочность при растяжении, МПа	RidgeRegression	0.007792	0.152404	0.002
Прочность при растяжении, МПа	LinearRegression	0.007064	0.152871	0.001
Прочность при растяжении, МПа	MLPRegressor	0.000178	0.152418	-0.004
Прочность при растяжении, МПа	LassoRegression	0.000000	0.152556	-0.006
Прочность при растяжении, МПа	RandomForestRegressor	-0.030630	0.154431	-0.035

Рисунок 18 – Результаты работы регрессионных моделей с подбором гиперпараметров по сетке для параметра Прочность при растяжении, мПа.

Лучшие результаты (из сравниваемых) получила модель Гребневой регрессии со следующими параметрами: {'alpha': 1, 'fit_intercept': True, 'normalize': True, 'solver': 'saga'}, где

1)“alpha” – константа, управляющая регуляризацией, не должна равняться 0;

2)“fit_intercept” – вычисление отрезка пересечения, в позиции False не вычисляет пересечение;

3)“normalize” – булевый параметр, отвечающий за нормализацию входных данных;

4)“solver” – решатель для использования в вычислительных процедурах, в данном случае “saga” использует стохастические средний градиентный спуск.

Удовлетворительных результатов при текущем построении моделей по прогнозированию параметра Прочность при растяжении, Мпа достичь не удалось.

2.6. Разработка нейронной сети для параметра «Соотношение матрица-наполнитель»

До начала работы с разработкой нейронной сети для прогнозирования параметра «Соотношение матрица-наполнитель» проводим так же разбивку нормализованного датасета df_norm на обучающую и тестовую выборку в со отношении 70 процентов на 30 процентов.

Во всех моделей входящий слой имеет 12 входов, так как 12 входных параметров, на выходе 1 нейрон, требуемый к прогнозированию параметр.

Далее пробуем разную структуру сети.

Первую модель строим в формате 12-8-8-1 с функцией активации ReLU на всех слоях (рисунок 19):

```
model_X3.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 104

 dense_1 (Dense)             (None, 8)                 72

 dense_2 (Dense)             (None, 1)                 9

=================================================================
Total params: 185
Trainable params: 185
Non-trainable params: 0
_________________________________________________________________
```

```
#собираем модель
model_X3.compile(loss="mae", optimizer="adam")
```

```
#обучаем модель
history = model_X3.fit(X3_train, y3_train, batch_size=100, epochs=60, validation_split=0.3)
```

Рисунок 19 Модель X3 нейронной сети для параметра Соотношение матрица-наполнитель

Оценка модели 1: 0.15512359142303467

Далее представлен график потерь и график обучения модели (рисунок 20, 21)

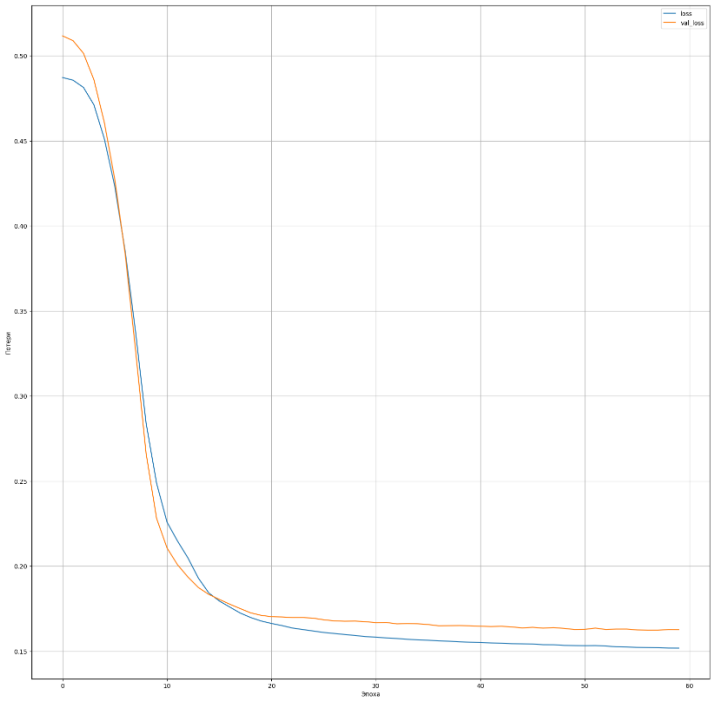


Рисунок 20 – График потерь на каждой эпохе для модели X3.

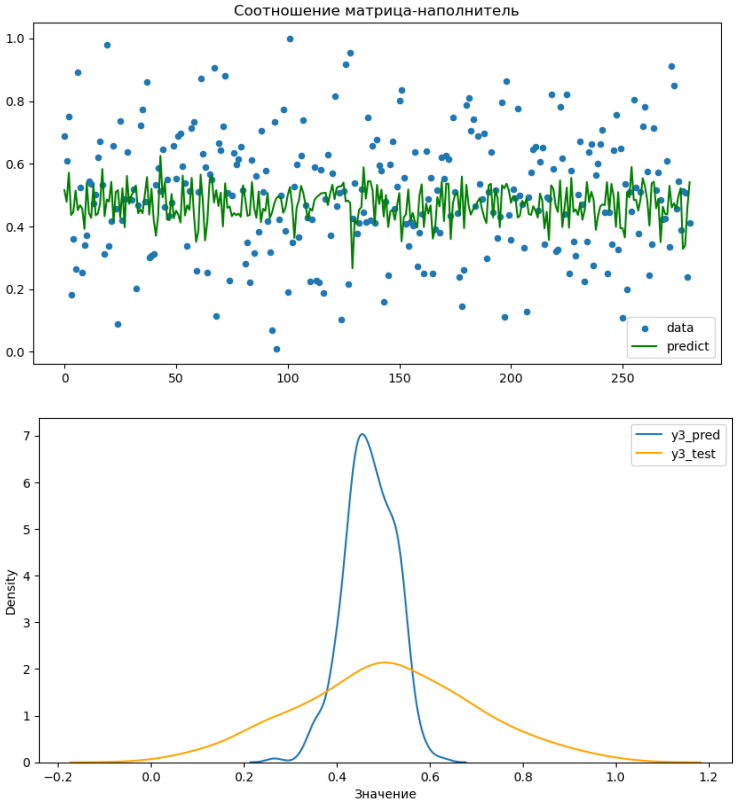


Рисунок 21 – График прогнозирования целевой переменной для модели X3.

Пробуем создать еще одну тестовую модель X3_test1, убрав активационную функцию на последнем слое, что теоретически допустимо для регрессионных нейронных сетей.

Для модели X3_test1 получаем следующую оценку: 0.1580493003129959, что дает немного лучший результат в сходимости модели, но не оказывает значительного влияния на результат.

Для модели X3_test2 пробуем увеличить количество слоев и применить метод прореживания Dropout для регуляризации обучения (рисунок 22)

```
model_X3_test2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 12)                156

 dropout (Dropout)           (None, 12)                0

 dense_7 (Dense)             (None, 8)                 104

 dropout_1 (Dropout)         (None, 8)                 0

 dense_8 (Dense)             (None, 4)                 36

 dropout_2 (Dropout)         (None, 4)                 0

 dense_9 (Dense)             (None, 1)                 5

=================================================================
Total params: 301
Trainable params: 301
Non-trainable params: 0
_________________________________________________________________
```

```
model_X3_test2.compile(loss="mae", optimizer="adam")
history_test2 = model_X3_test2.fit(X3_train, y3_train, batch_size=30, epochs=60, validation_split=0.3)
```

Рисунок 22 – Модель X3_test2 для параметра Соотношение матрица-наполнитель

Оценка данной модели составила 0.1562628448009491,что даже хуже чем в предыдущей тестовой модели.

В результате, значительных улучшений при перестройке исходной модели нейронной сети не удалось добиться.

2.7. Разработка приложения

В ходе выполнения работы разработано приложение для предсказания параметров прочности при растяжении и соотношение матрица-наполнитель. Приложение разработано в среде Visual Studio Code с помощью библиотеки FLASK.

Для разработки приложения были выбраны две модели:

а) модель линейной регрессии для параметра «Модель упругости при растяжении» - сохранена с помощью библиотеки pickle (рисунок 23).

```
#сохраняем модель Линейной Регрессии для Модуля упругости при растяжении
#Модуль упругости при растяжении,гПа    LinearRegression    0.007377    0.146740    0.006

with open("lr1_model.pkl", "wb") as f:
    pickle.dump(lr1, f)
```

Рисунок 23 – Сохранение модели машинного обучения в файл pkl

б) модель нейронной сети вариант Х3 для параметра «Соотношение матрица-наполнитель» - сохранена методом библиотеки tensorflow.save (рисунок 24).

```
#сохраняем модель - выбрана нейронная сеть для Соотношения матрица-наполнительб первый вариант


model_X3.save('vkr_nn_model')
```

```
WARNING:absl:Found untraced functions such as _update_step_xla while saving (showing 1 of 1). These functions will not be d
tly callable after loading.

INFO:tensorflow:Assets written to: vkr_nn_model\assets
```

Рисунок 24 – Сохранение модели нейронной сети

После сохранения и до запуска в производство приложения модели еще раз выгружены из файлов и проверены на корректность проведенной операции.

В структуре приложения заложены файлы:

1)my_app.py - приложение, язык Python,

2)main.html, upr.html, mn.html – код для веб страниц,

3)vkr_nn_model – сохраненная модель нейронной сети,

4)lr1_model.pkl – сохраненная модель линейной регрессии,

5)requirement.txt – файл с указанием требуемых библиотек,

6)ProcFile.

Ниже приведен поэтапный запуск приложения (рисунок 25-31).

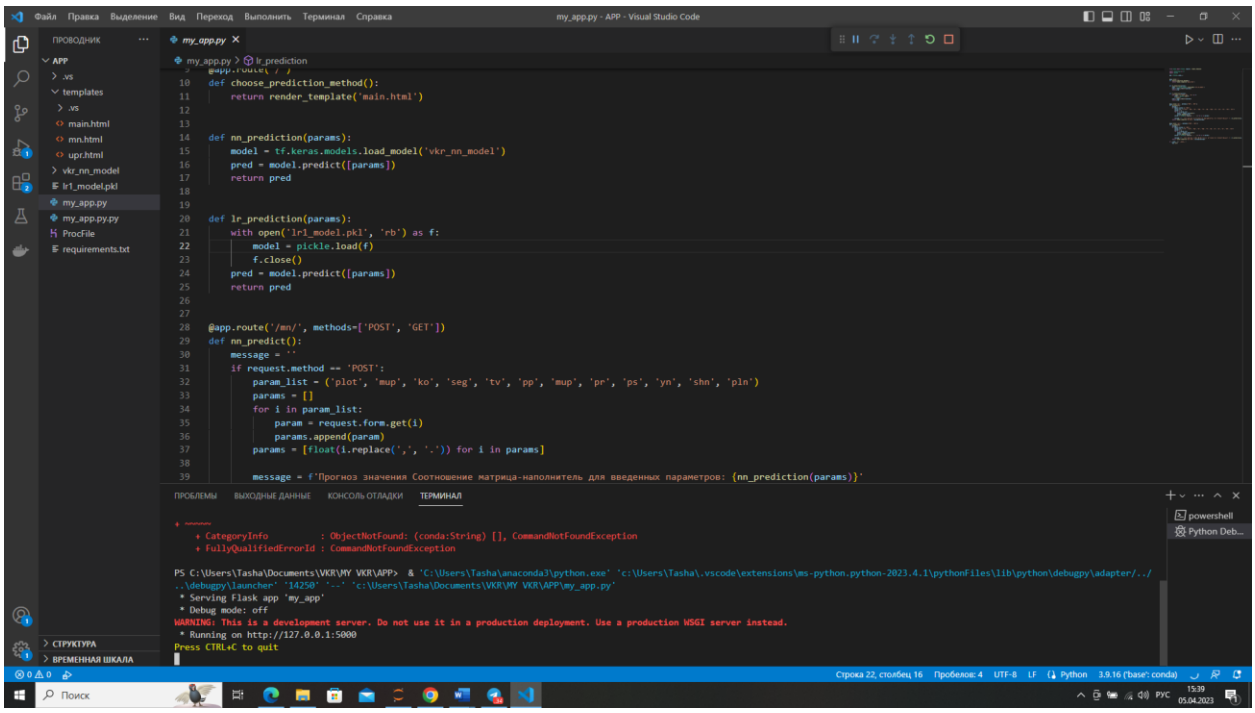


Рисунок 25 – Запуск приложения

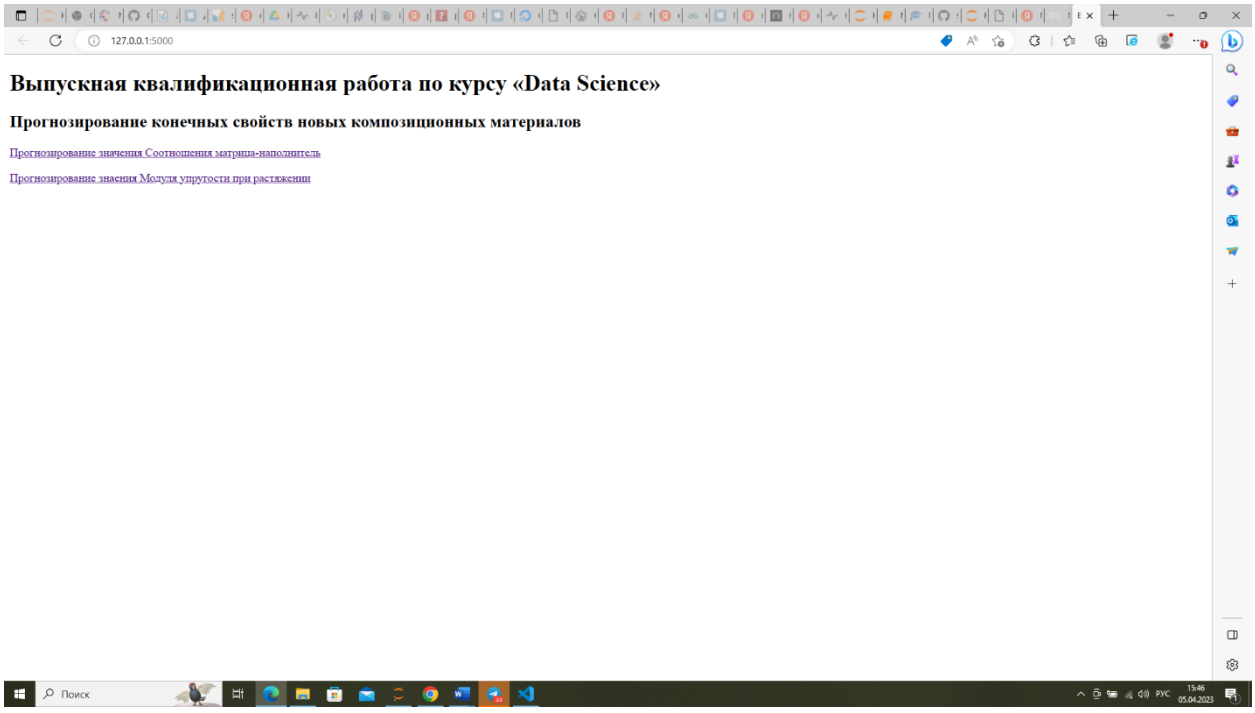


Рисунок 26 – Первая страница приложения

На первой странице предлагается выбрать прогнозирование какого параметра необходимо провести.

Далее необходимо ввести значения 12 входящих параметров.

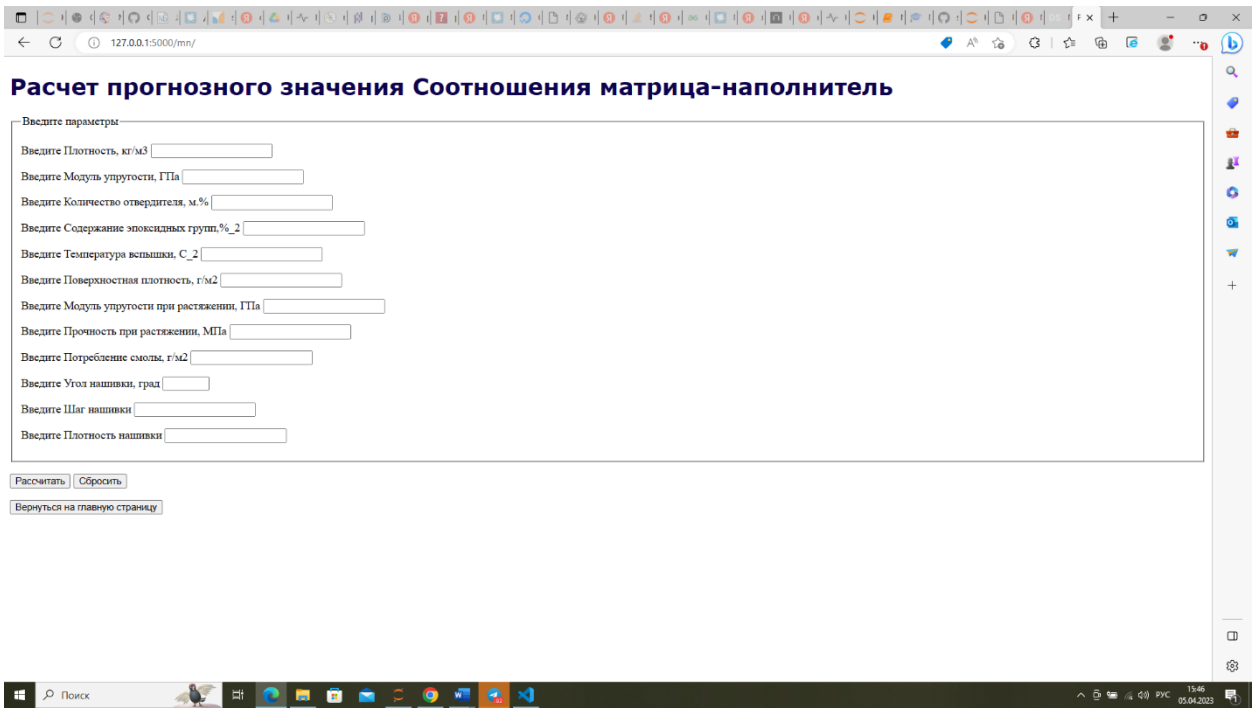


Рисунок 27 – Страница приложения для ввода входящих параметров для показателя Соотношение матрица-наполнитель

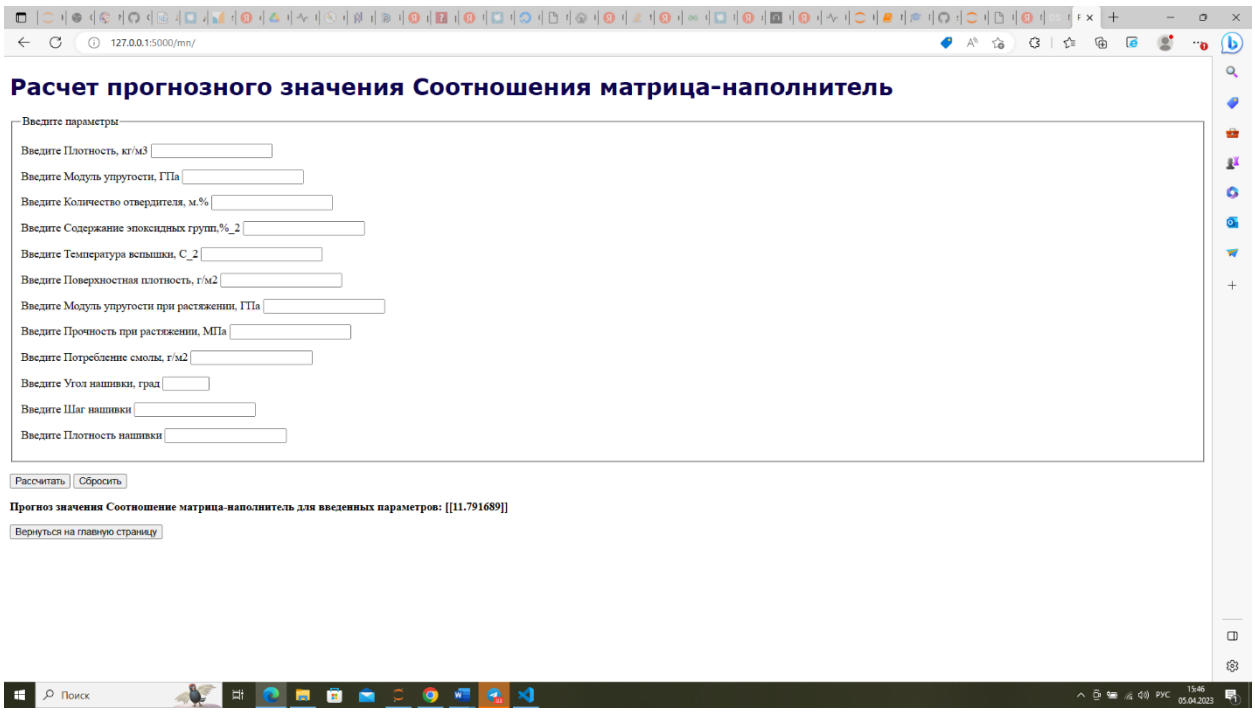


Рисунок 28 – Результат выполнения расчета для показателя Соотношение матрица-наполнитель

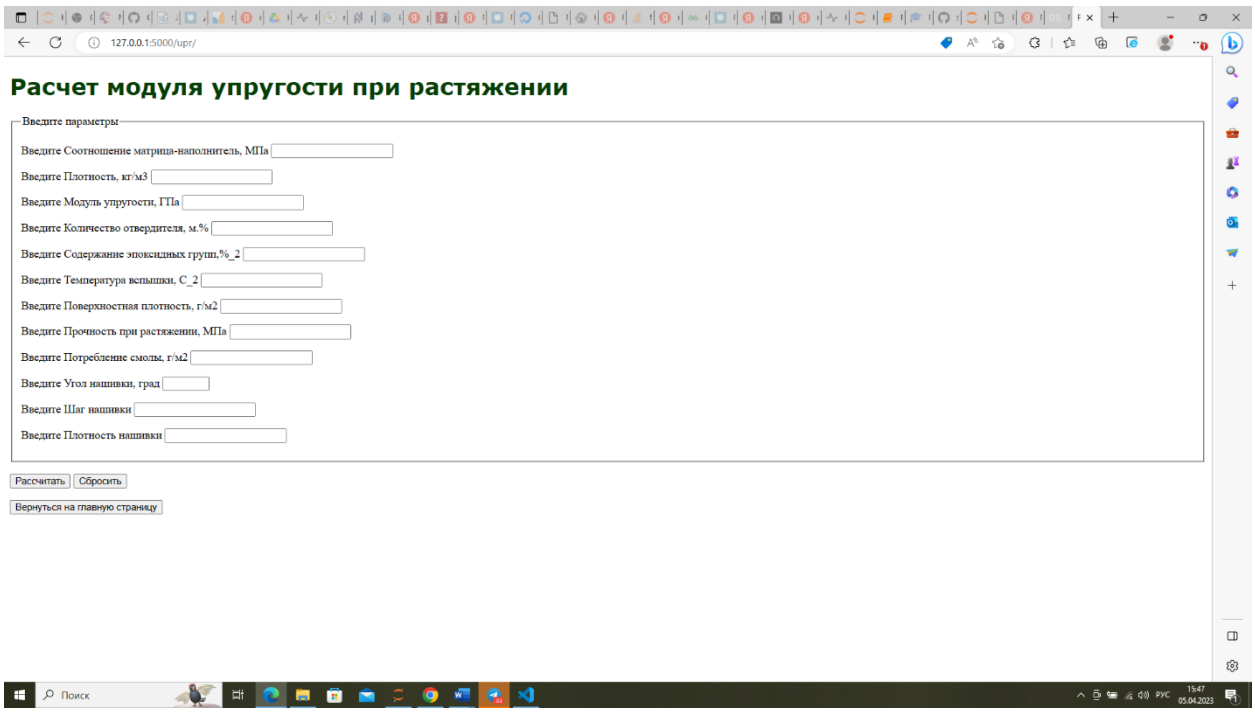


Рисунок 29 – Страница приложения для ввода входящих параметров для показателя Модуль упругости при растяжении

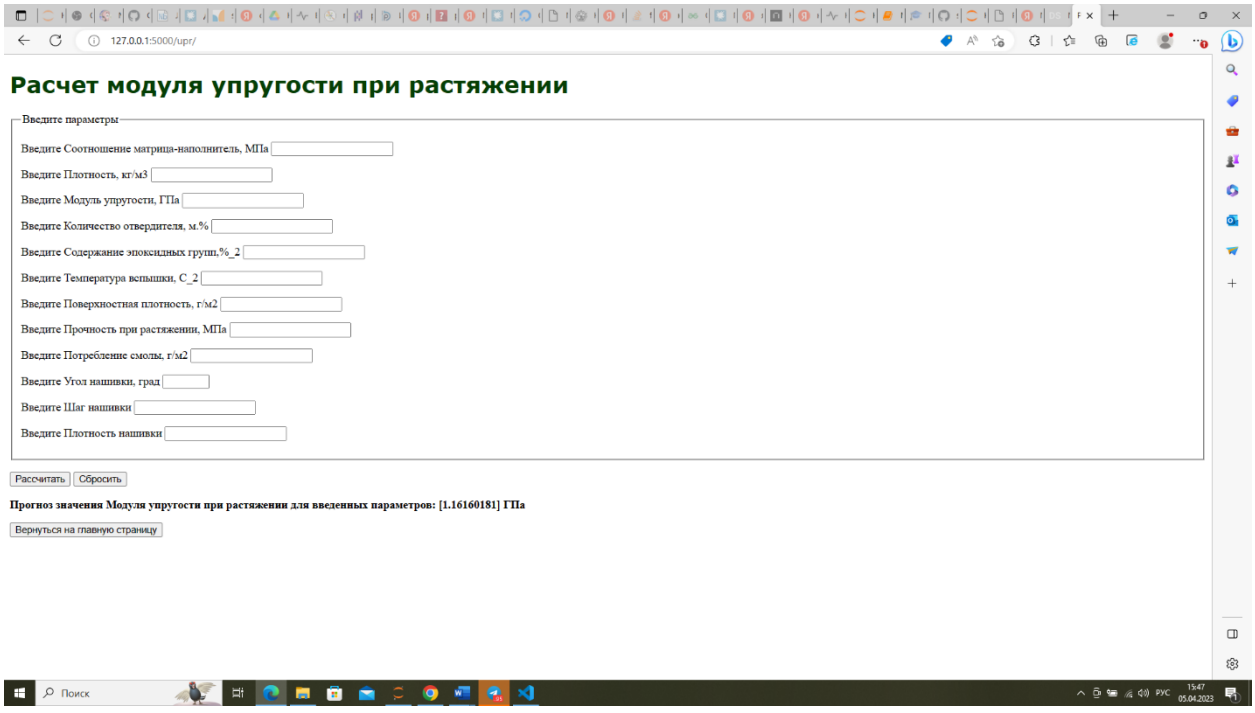


Рисунок 30 – Результат выполнения расчета для показателя Модуль упругости при растяжении

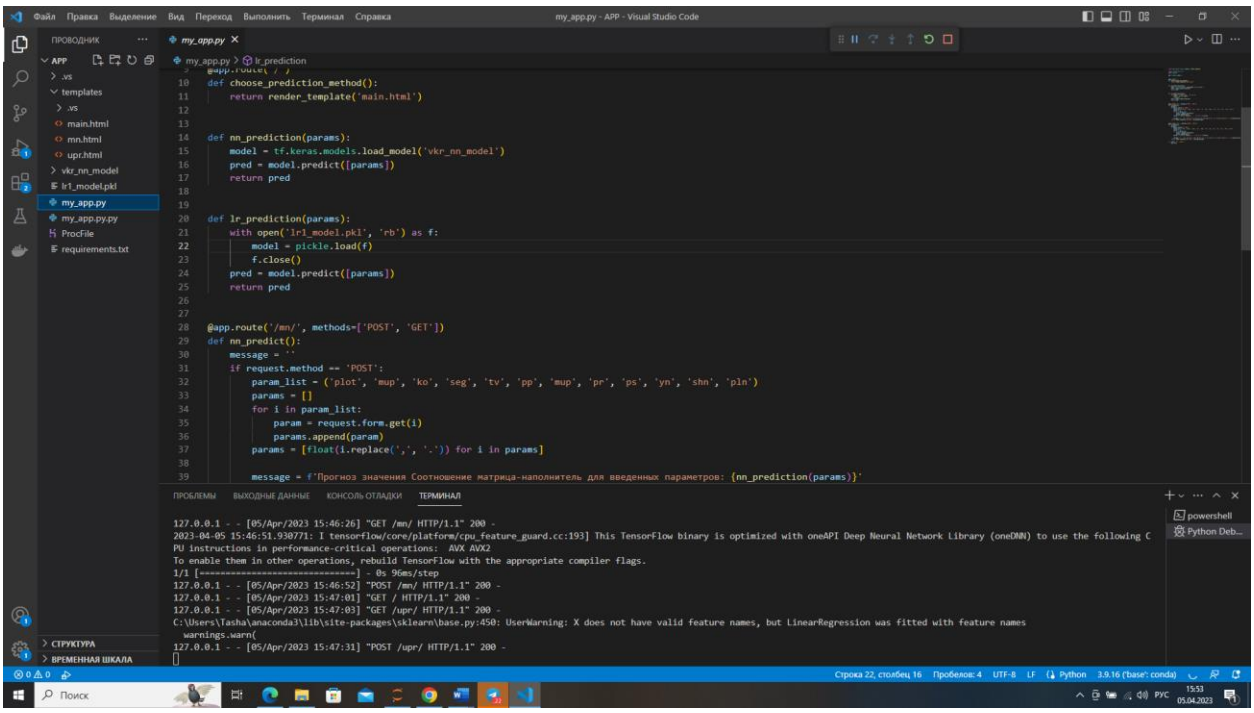


Рисунок 31 – Контроль работы приложения в среде Visual Studio Code

2.8. Создание удаленного репозитория с результатами проведенной работы

Репозиторий для выпускной квалификационной работы создан на ресурсе GitHub и доступен по ссылке https://github.com/TashaGit/Final_qualification_paper/.

Репозиторий содержит программный код, презентацию, исходные данные, пояснительную записку.

3. Заключение

В результате работы были разработаны и протестированы несколько моделей машинного обучения, позволяющих прогнозировать конечные свойства композитных материалов, в том числе: линейная регрессия, ансамбли лесов, XGBoost, полносвязная нейронная сеть и так далее.

Разработана нейронная сеть, которая рекомендует соотношение матрица-наполнитель для создания композиционного материала.

Разработано приложение, которое может быть использовано для прогноза параметра прочности при растяжении и соотношения матрица-наполнитель для композиционных материалов.

Модели могут быть использованы для переобучения и решения аналогичных задач на новых входных данных.

Однако, для более точной настройки моделей желательно провести более детальное исследование дата сета на наличие либо синтетических данных, либо комментария эксперта для вероятного разделения данных на потоки для подачи в модели.

4. Библиографический список

1 Артеменко, С.Е. Структура и свойства базальто-, стекло- и углепластиков, сформированных по интеркаляционной технологии / С.Е. Артеменко, О.Г. Васильева, Ю.А. Кадыкова, А.Н. Леонтьев // Полимеры-2004: III Всерос. Каргинская конф. М., 2004. Т. 2. С. 162.

2 Джулли, П.: Библиотека Keras - инструмент глубокого обучения / пер. с англ. А.А. Слинкин. – ДМК Пресс, 2017. – 249 с.

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

4 Жерон, Орельен. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем. Пер. с англ. - СпБ.: ООО «Альфа-книга»: 2018. – 688 с.

5 Кадыкова, Ю.А. Полимерные композиционные материалы на основе волокон различной химической природы / Ю.А. Кадыкова, А.Н. Леонтьев, О.Г. Васильева, С.Е. Артеменко // Строительные материалы, оборудование, технологии XX1 века. 2002. № 6. С. 10-11.

6 Николенко, С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей. - СПб.: Питер. - 2020. - 480 с. ISBN: 978-5-4461-1537-2.

7 Рассел С., Норвиг П. Искусственный интеллект: современный подход, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2007. - 1408 с.

8 Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. - М.: Горячая Линия - Телеком. - 2013. - 384 с. ISBN: 978-5-9912-0320-3.

9 Статистическая обработка данных, планирование эксперимента и случайные процессы : учебное пособие для вузов / Берикашвили В. Ш., Оськин С. П. - 2-е изд., испр. и доп. - М.: Юрайт, 2021. - 163 с.

10 Фостер Д. Генеративное глубокое обучение. Творческий потенциал нейронных сетей. - СПб.: Питер. - 2020. - 336 с. - ISBN: 978-5-4461-1566-2.

12 Шитиков В.К., Мастицкий С.Э. (2017) Классификация, регрессия и другие алгоритмы Data Mining с использованием R. 351 с. − Электронная книга, адрес доступа: https://github.com/ranalytics/data-mining.

12 Кашнитский, Ю. Открытый курс машинного обучения. Тема 3. Классификация, деревья решений и метод ближайших соседей [Электронный ресурс] : – Режим доступа: https://habr.com/ru/company/ods/blog/322534/. (дата обращения: 12.04.2023).

13 Константинов, М. Краткий курс машинного обучения или как создать нейронную сеть для решения скоринг задачи [Электронный ресурс] : – Режим доступа: https://habr.com/ru/post/340792/. (дата обращения: 12.04.2023).

14 Масзанский, А. Метод k-ближайших соседей (k-nearest neighbour) [Электронный ресурс] : – Режим доступа: https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19/. (дата обращения: 05.04.2023).

15 Метод обратного распространения ошибки: математика, примеры, код [Электронный ресурс] : – Режим доступа: https://neurohive.io/ru/osnovy-data-science/obratnoe-rasprostranenie/. (дата обращения: 15.04.2023).

16 Радченко, В. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес [Электронный ресурс] : – Режим доступа: https://habr.com/ru/company/ods/blog/324402. (дата обращения: 23.03.2023).

17 Регрессия в машинном обучении: оптимальный алгоритм [Электронный ресурс] : – Режим доступа: https://proglib.io/p/ml-regression. (дата обращения: 23.03.2023).

18 Федоров, А. Решение задачи регрессии полносвязной нейронной сетью [Электронный ресурс] : – Режим доступа: https://www.bizkit.ru/2019/11/05/14921/. (дата обращения: 23.03.2023).

19 Функции активации в нейронных сетях [Электронный ресурс] : – Режим доступа: http://www.aiportal.ru/articles/neural-networks/activation-function.html. (дата обращения: 12.04.2023).

20 Хлевнюк, А. Основы линейной регрессии [Электронный ресурс] : – Режим доступа: https://habr.com/ru/post/514818/. (дата обращения: 23.03.2023).