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 Штаньков Никита Игоревич

Москва, 2023

СОДЕРЖАНИЕ

Введение

В настоящее время, чтобы быть успешным, необходимо постоянно совершенствоваться и использовать науку и технологии, в том числе при создании новых материалов. Новые материалы являются ключевым фактором в технологическом прогрессе, позволяя создавать новые устройства и технику, которые ранее были недоступны. Однако, создание новых материалов является сложным процессом, требующим больших затрат времени и ресурсов. Поэтому, существует необходимость в разработке методов и технологий, которые бы позволили сократить время и средства, необходимые для создания новых материалов.

Одним из основных методов является прогнозирование конечных свойств новых материалов, в частности, композиционных материалов. Композиционные материалы представляют собой материалы, состоящие из двух или более химически разнородных компонентов с четкой границей раздела между ними. Эти материалы характеризуются свойствами, которые не присущи каждому компоненту в отдельности. За счет выбора и варьирования объемной доли армирующих элементов, свойства композитов можно регулировать в значительных пределах.

Однако, производство композитных материалов является дорогостоящим, и зная только характеристики компонентов, невозможно рассчитать свойства композита. Поэтому, для получения заданных свойств, требуется проводить множество испытаний различных комбинаций. Чтобы сократить время и затраты на создание определенного материала, возможно использование системы поддержки производственных решений, основанных на принципах машинного обучения. Такая система может помочь в прогнозировании свойств композитных материалов, ускорить процесс исследований, сделать процесс создания новых материалов более эффективным и безопасным.

1. Аналитическая часть

1.1 Постановка задачи

В этой работе исследуется композитный материал, который состоит из базальтопластика в качестве матрицы и углепластика в качестве нашивок. Мы получили датасет от специалистов в области, который содержит информацию о свойствах матрицы и наполнителя, параметрах производства и свойствах готового композита. Наша задача как специалистов в машинном обучении - разработать модели, которые могут прогнозировать значения определенных свойств композита на основе других свойств. [1]

Датасет состоит из двух файлов: X_bp.xlsx (признаки базальтопластика) и X_nup.xlsx (признаки углепластика).

Файл X_bp.xlsx содержит 1023 строки, индекс и 10 признаков.

Файл X_nup.xlsx содержит 1040 строк индекс и 3 признака.[2]

В условии задания сказано, что файлы требуют объединения с типом INNER по индексу.

Необходимо обучить алгоритм машинного обучения, который будет определять значения:

- Модуль упругости при растяжении, Гпа;
- Прочность при растяжении, Мпа;

Так же необходимо написать нейронную сеть, которая будет рекомендовать:

- Соотношение матрица-наполнитель.

Мы также должны создать приложение, которое позволит специалистам в предметной области удобно использовать эти модели.

---

[1] Банкрашков, А.В. Программирование для детей на языке Python / А.В. Банкрашков. - М.: АСТ, 2018. - 288 с.

[2] Композиционные материалы : учебное пособие для вузов / Д. А. Иванов, А. И. Ситников, С. Д. Шляпин ; под редакцией А. А. Ильина. — Москва : Издательство Юрайт, 2019 — 253 с. — (Высшее образование). — Текст : непосредственный.

1.2 Описание используемых методов

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

В настоящее время разработано множество методов регрессионного анализа. Например, простая и множественная линейная регрессия. Эти модели являются параметрическими в том смысле, что функция регрессии определяется конечным числом неизвестных параметров, которые оцениваются на основе данных.[3]

1. Линейная регрессия (LinearRegression);

Простая линейная регрессия имеет место, если рассматривается зависимость между одной входной и одной выходной переменными. Для этого определяется уравнение регрессии и строится соответствующая прямая, известная как линия регрессии:

$$y = ax + b$$

Коэффициенты a и b, называемые также параметрами модели, определяются таким образом, чтобы сумма квадратов отклонений точек, соответствующих реальным наблюдениям данных, от линии регрессии была бы минимальной. Коэффициенты обычно оцениваются методом наименьших квадратов.[4]

Если ищется зависимость между несколькими входными и одной выходной переменными, то имеет место множественная линейная регрессия. Соответствующее уравнение имеет вид:

$$Y = b0 + b1 * x1 + b2 * x2 + \cdots + bn * xn$$

где n - число входных переменных.

---

[3] Мэтиз, Э. Изучаем PYTHON.Программирование игр, визуализация данных, веб-приложения / Э. Мэтиз. - СПб.: Питер, 2017. - 496 с.

[4] Лутц, М. Программирование на Python. Т. 1 / М. Лутц. - М.: Символ, 2016. - 992 с.

Очевидно, что в данном случае модель будет описываться не прямой, а гиперплоскостью. Коэффициенты уравнения множественной линейной регрессии подбираются так, чтобы минимизировать сумму квадратов отклонения реальных точек данных от этой гиперплоскости.

Линейная регрессия — первый тщательно изученный метод регрессионного анализа. Его главное достоинство — простота. Такую модель можно построить и рассчитать даже без мощных вычислительных средств. Простота является и главным недостатком этого метода. [5]

2. Лассо (LASSO) регрессия;

LASSO регрессия — это метод регуляризации линейной регрессии, где коэффициенты модели ограничены с помощью штрафа L1 нормы. Это означает, что модель минимизирует сумму квадратов ошибок, при этом ограничивая величину суммы абсолютных значений коэффициентов. Этот метод позволяет автоматически отбирать наиболее значимые признаки и устранять шум в данных.

Преимущества LASSO регрессии заключаются в её способности к сокращению переобучения и возможность выбора наиболее значимых признаков. Кроме того, LASSO регрессия соответствует принципу "мотыги Мора", что означает уменьшение размерности модели за счёт исключения менее важных признаков.

Формула для LASSO регрессии: минимизация суммы квадратов ошибок с ограничением суммы абсолютных значений коэффициентов. Математически это записывается как:

min ||y - Xw||^2 + λ||w||_1

Где:

– y - вектор значений зависимой переменной;

– X - матрица значений признаков;

– w - вектор коэффициентов модели;

[5] Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. - М.: Символ-Плюс, 2011. - 608 с.

– ||.||_2 - норма L2;

– ||.||_1 - норма L1;

– λ - коэффициент регуляризации, который отвечает за величину штрафа.

Таким образом, LASSO регрессия становится полезной моделью при работе с данными, где существует большое число признаков, которые могут вносить шум в модель.

3. Гребневая (Ridge) регрессия;

Ridge регрессия — это метод множественной линейной регрессии, который используется для предсказания значений зависимой переменной на основе значений нескольких независимых переменных. Она отличается от обычной линейной регрессии тем, что в её формуле используется дополнительный параметр - коэффициент регуляризации.

Преимуществами Ridge регрессии являются её способность справляться с проблемами мультиколлинеарности, когда существует сильная корреляция между независимыми переменными. Ridge регрессия также помогает предотвратить переобучение модели, что позволяет увеличить её обобщающую способность. Кроме того, Ridge регрессия может улучшить качество модели, если данные содержат выбросы.

Формула для Ridge регрессии выглядит следующим образом:

$Y = \beta0 + \beta1X1 + \beta2X2 + ... + \betanXn + \lambda\Sigma(\betai^2)$

где Y - зависимая переменная, Xi - независимые переменные, βi - коэффициенты регрессии, λ - коэффициент регуляризации, который контролирует вклад регуляризации в общую ошибку модели. Чем больше значение λ, тем сильнее регуляризация и меньше свободы модели.

Гребневая регрессия — вариация линейной регрессии, очень похожая на регрессию LASSO. Она также применяет сжатие и хорошо работает для данных, которые демонстрируют сильную мультиколлинеарность. Самое большое различие между ними в том, что гребневая регрессия использует регуляризацию L2, которая взвешивает ошибки по их квадрату, чтобы сильнее наказывать за более значительные ошибки. Регуляризация позволяет интерпретировать модели. Если

коэффициент стал 0 (для Lasso) или близким к 0 (для Ridge), значит данный входной признак не является значимым.[6]

4. ElasticNet регрессия;

ElasticNet регрессия является одним из методов машинного обучения, который решает проблему мультиколлинеарности (сильной корреляции между факторами) в задачах регрессии. Данный метод базируется на двух видах регуляризации – L1 и L2, что позволяет улучшить качество модели и снизить влияние шума в данных. Такой алгоритм широко применяется в бизнесе, а также в научных исследованиях и финансово-экономических моделях.

Преимущества ElasticNet регрессии заключаются в ее универсальности и гибкости настройки. Формулы L1 и L2 регуляризации могут быть заданы произвольно, что позволяет аналитику настраивать метод под конкретные данные. Кроме того, данный метод предназначен для поиска оптимальных коэффициентов регрессии с учетом равновесия между шумом и сигналом в данных. Таким образом, улучшается точность прогнозирования и общего качества модели.

Формула ElasticNet регрессии представляет собой сумму двух элементов – L1 регуляризации (Lasso) и L2 регуляризации (Ridge). Они задаются соответствующим образом в формуле регрессии и зависят от настроек коэффициента α, который контролирует баланс между двумя методами регуляризации. Формула регрессии выглядит следующим образом:

$$y = \beta 0 + \beta 1x1 + \beta 2x2 + \ldots + \beta nxn + \varepsilon$$

где y – предиктор, βi – коэффициенты регрессии, xi – факторы регрессии, а ε – ошибка модели.

5. Случайный лес;

Случайный лес (RandomForest) — представитель ансамблевых методов.

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

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

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

Где:

N – количество деревьев;

i – счетчик для деревьев;

b – решающее дерево;

x – сгенерированная нами на основе данных выборка.

Для определения входных данных каждому дереву используется метод случайных подпространств. Базовые алгоритмы обучаются на различных подмножествах признаков, которые выделяются случайным образом.

Преимущества случайного леса:

– высокая точность предсказания;

– редко переобучается;

– практически не чувствителен к выбросам в данных;

– одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки, данные с большим числом признаков;

– высокая параллелизуемость и масштабируемость.

Из недостатков можно отметить, что его построение занимает больше времени. А также теряется интерпретируемость.[7]

6. Метод k-ближайших соседей (KNeighborsRegressor);

Еще один метод классификации, который адаптирован для регрессии — метод k-ближайших соседей (k Nearest Neighbors). На интуитивном уровне суть

[7] ГрасД. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.

метода проста: посмотри на соседей вокруг, какие из них преобладают, таковым ты и являешься.

Преимущество k-neighbors регрессии заключается в ее простоте и универсальности. Алгоритм может использоваться для решения широкого круга задач в области машинного обучения и анализа данных, таких как прогнозирование цен на недвижимость, распознавание рукописных символов, определение возраста или пола на фотографии, анализ данных геопозиционирования и т.д.

Формула k-neighbors регрессии выглядит следующим образом:

$y = 1/ k \sum y_m$,

где y - значение целевой переменной, k - количество ближайших соседей, y_m - значение данной переменной у каждого из соседей. Количество соседей, на основе которого происходит прогнозирование, задается заранее и является параметром алгоритма. Чем меньше значение k, тем более гибкой будет модель и тем больше она будет склонна к переобучению. При увеличении значения k, модель становится более устойчивой и менее склонной к переобучению.

В случае использования метода для регрессии, объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны. Для реализации метода необходима метрика расстояния между объектами. Используется, например, эвклидово расстояние для количественных признаков или расстояние Хэмминга для категориальных. Этот метод — пример непараметрической регрессии.

7. Метод опорных векторов для регрессии (SVR);

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

Чаще всего он применяется в постановке бинарной классификации.

Основная идея заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Интуитивно, хорошее разделение достигается за счет гиперплоскости, которая имеет самое большое расстояние до

ближайшей точки обучающей выборке любого класса. Максимально близкие объекты разных классов определяют опорные вектора.

Если в исходном пространстве объекты линейно неразделимы, то выполняется переход в пространство большей размерности.

Решается задача оптимизации.

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

– линейная;

– полиномиальная;

– гауссовская (rbf).

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

Преимущество метода — его хорошая изученность.

Недостатки:

– чувствительность к выбросам;

– отсутствие интерпретируемости.

8. Градиентный бустинг;

Градиентный бустинг (GradientBoosting) — еще один представитель ансамблевых методов.

В отличие от случайного леса, где каждый базовый алгоритм строится независимо от остальных, бустинг воплощает идею последовательного построения линейной комбинации алгоритмов. Каждый следующий алгоритм старается уменьшить ошибку предыдущего.

Чтобы построить алгоритм градиентного бустинга, нам необходимо выбрать базовый алгоритм и функцию потерь или ошибки (loss). Loss-функция — это мера, которая показывает, насколько хорошо предсказание модели соответствуют данным. Используя градиентный спуск и обновляя предсказания, основанные на скорости обучения (learning rate), ищем значения, на которых loss минимальна.

Бустинг, использующий деревья решений в качестве базовых алгоритмов, называется градиентным бустингом над решающими деревьями. Он отлично работает на выборках с «табличными», неоднородными данными и способен эффективно находить нелинейные зависимости в данных различной природы. На настоящий момент это один из самых эффективных алгоритмов машинного обучения. Благодаря этому он широко применяется во многих конкурсах и промышленных задачах. Он проигрывает только нейросетям на однородных данных (изображения, звук и т. д.).

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

9. Деревья решений;

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

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

Дерево состоит из элементов двух типов: узлов (node) и листьев (leaf).

В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу. В результате проверки множество примеров, попавших в узел, разбивается на два подмножества: удовлетворяющие правилу и не удовлетворяющие ему. Затем к каждому подмножеству вновь применяется правило и процедура рекурсивно повторяется пока не будет достигнуто некоторое условие остановки алгоритма. В последнем узле проверка и разбиение - не производится и он объявляется листом.

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

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут, по которому это будет сделано

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

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

Недостаток деревьев решений - склонность переобучаться. Переобучение в случае дерева решений — это точное распознавание примеров, участвующих в обучении, и полная несостоятельность на новых данных.

10. Нейронная сеть;

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

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

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

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

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

– входной слой — его размер соответствует входным параметрам;

– скрытые слои — их количество и размерность определяем специалист;

– выходной слой — его размер соответствует выходным параметрам.

Прямое распространение – это процесс передачи входных значений в нейронную сеть и получения выходных данных, которые называются прогнозируемым значением.

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

Для обновления весов в модели используются различные оптимизаторы. Количество эпох показывает, сколько раз выполнялся проход для всех примеров обучения.

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

1.3 Разведочный анализ данных

Рассмотрим первый вариант.

Провели ряд действий в ходе анализа данных, включая объединение двух исходных датасетов, сравнили два метода удаления выбросов («Метод 3-х сигм» и «Метод межквартильных расстояний»), применили метод межквартильных расстояний для удаления выбросов и получили 936 строки. После этого повторно проверили данные на наличие выбросов, удалили строки с выбросами и тем самым уменьшили количество строк до 926. Мы повторили процедуру еще раз, удалили еще несколько строк и получили 922 строки. Были использованы различные методы для проверки выбросов и корреляций. После тщательной проверки и удаления всех выбросов, оценили средние и медианные значения датасета, а также

плотность ядра. В ходе оценки было замечено, что значения находятся в разных диапазонах, что привело к необходимости нормализации данных. Однако после нормализации, зависимости не были обнаружены, что заставило нас начать сначала.

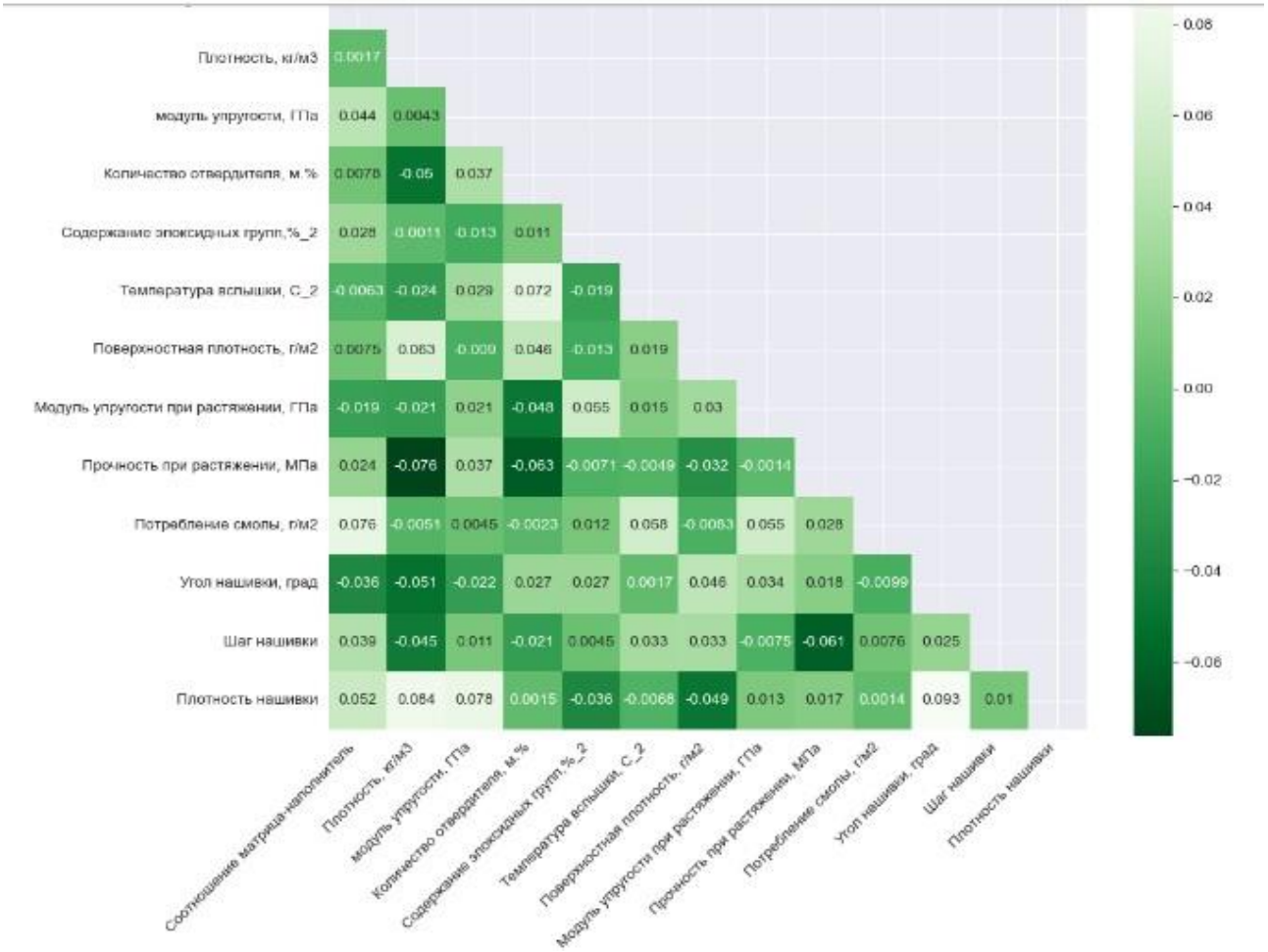

Рисунок 1 - Матрица корреляции

Рассмотрим второй вариант.

В ходе нашего исследования мы приняли ряд мер по обработке данных. Сначала были вручную удалены 57 строк, содержащих некорректные целые числовые значения. После этого объединили данные и получили 983 строки. Далее, сравнили два основных метода для удаления выбросов: "Метод 3-х сигм", который позволил нам выявить 22 выброса, и "Метод межквартильных расстояний", который обнаружил 88 выбросов. Было решено использовать метод межквартильных расстояний для удаления выбросов, что привело к уменьшению количе-

ства строк до 899. Однако при визуализации корреляционной матрицы, не получилось обнаружить явных зависимостей, что привело к пересмотру подхода и началу работы сначала.

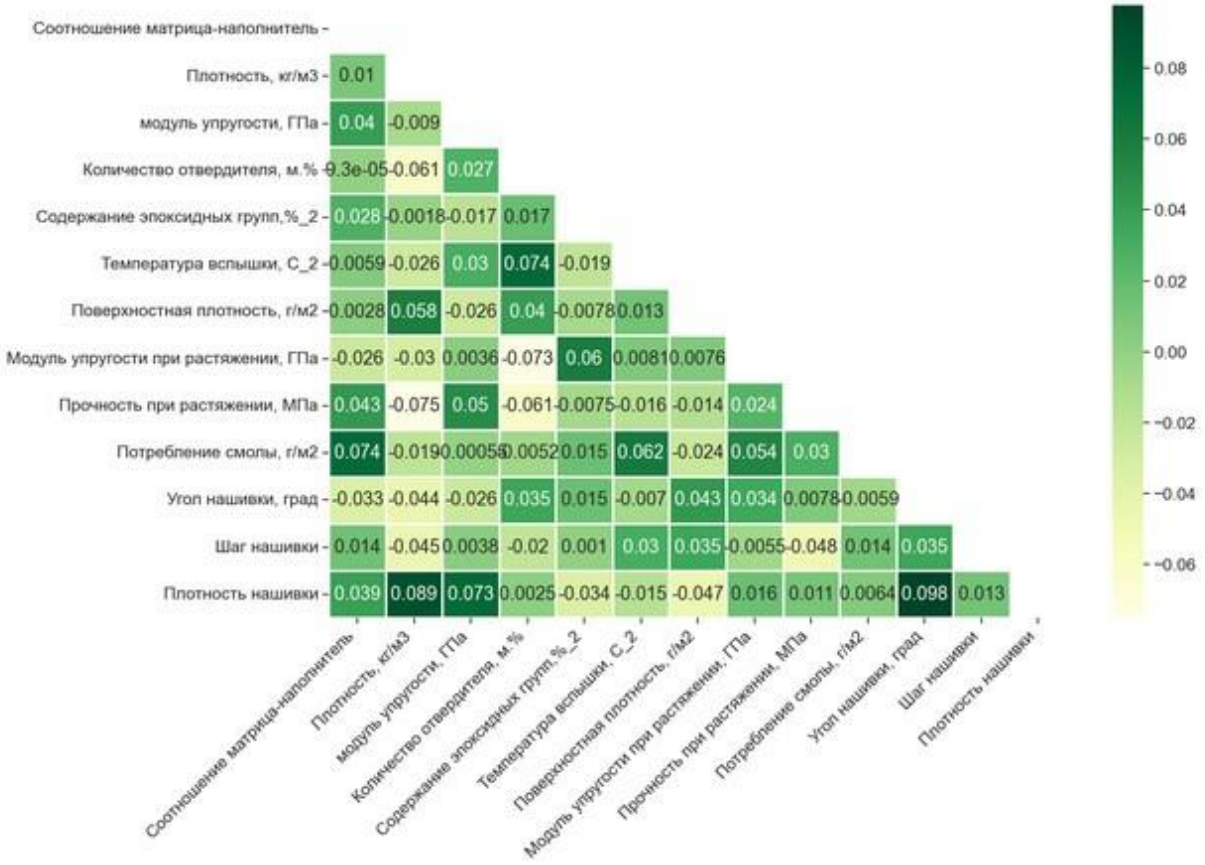

Рисунок 2 - Матрица корреляции

Рассмотрим третий вариант.

Проанализировав ещё раз данный нам датасет, мы пришли к выводу что в столбце "Температура вспышки, С_2" имеется всего 13 значений с показателем точности менее чем до сотых. Исходя из этого, мы проделали всю процедуру подготовки и анализа данных заново.

Сначала мы объединили два исходных датасета, X_bp.xlsx и X_nup.xlsx, с помощью метода INNER, не производя никаких манипуляций над ними. Это позволило получить новый датасет, размер которого представлен на графическом изображении.

```
#объединяем
df = df_bp.join(df_nup, how='inner')
df.shape
```

```
(1023, 13)
```

Рисунок 3 — Размерность нового датасета

Затем очистили наш датасет от синтезированных данных (рисунок 4).

```
# Удаляем данные в которые не верим.
df_clean = df[(df["Температура вспышки, С_2"] == 100) | (df["Температура вспышки, С_2"] == 300)].reset_index(drop=True)
df_clean.shape
```

Рисунок 4 — Очистка от синтезированных данных

После объединения двух исходных датасетов X_bp.xlsx и X_nup.xlsx с помощью метода INNER без каких-либо манипуляций, мы получили новый датасет, размер которого представлен на рисунке. Далее мы обнаружили, что столбец "Угол нашивки, град" содержит только одно уникальное значение, равное 0, и приняли решение удалить данный столбец, так как он не влияет на выборку. После удаления размерность датасета изменилась, как показано на рисунке 5.

```
df_clean.shape
```

```
(13, 13)
```

Рисунок 5 — Размер чистого датасетика

В процессе исследования очищенного датасета было обнаружено, что признаки имеют разные диапазоны значений, что может негативно повлиять на работу моделей машинного обучения. Для решения этой проблемы был использован метод MinMaxScaler, который масштабирует значения признаков таким образом, чтобы они находились в интервале от 0 до 1. Это позволяет уравнять диапазоны значений признаков и улучшить качество моделей. На рисунке 6 представлена описательная статистика очищенного датасета.

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	13.0	3.133151	0.973349	1.598174	2.877358	2.934783	3.557018	4.897959
Плотность, кг/м3	13.0	1986.923077	82.197386	1880.000000	1930.000000	1980.000000	2030.000000	2160.000000
модуль упругости, ГПа	13.0	784.959514	386.777556	302.000000	540.000000	738.736842	889.000000	1628.000000
Количество отвердителя, м.%	13.0	121.384615	27.457660	30.000000	129.000000	129.000000	129.000000	129.000000
Содержание эпоксидных групп,%_2	13.0	21.328297	0.282303	21.250000	21.250000	21.250000	21.250000	22.267857
Температура вспышки, С_2	13.0	284.615385	55.470020	100.000000	300.000000	300.000000	300.000000	300.000000
Поверхностная плотность, г/м2	13.0	493.846154	311.194967	210.000000	210.000000	380.000000	470.000000	1010.000000
Модуль упругости при растяжении, ГПа	13.0	73.769231	3.104312	70.000000	70.000000	73.333333	75.000000	78.000000
Прочность при растяжении, МПа	13.0	2366.666667	499.567714	1800.000000	2000.000000	2455.555556	3000.000000	3000.000000
Потребление смолы, г/м2	13.0	215.384615	63.850788	120.000000	220.000000	220.000000	220.000000	300.000000
Шаг нашивки	13.0	8.153846	1.993579	4.000000	7.000000	9.000000	10.000000	10.000000
Плотность нашивки	13.0	58.384615	8.211156	47.000000	57.000000	57.000000	60.000000	70.000000

Рисунок 6 — Очищенный датасет

Применение метода MinMaxScaler позволило нормализовать данные в мини-датасете, что имеет важное значение для анализа и использования данных в машинном обучении. Нормализация данных позволяет сравнивать признаки между собой, так как они находятся в одном масштабе, и также обеспечивает корректную работу алгоритмов машинного обучения, которые требуют нормализованных данных.

Давайте рассмотрим описательную статистику нашего очищенного датасета после применения метода нормализации данных - MinMaxScaler. Эти данные представлены на рисунке 7. Мы можем сравнить эти результаты с тем, как выглядел наш датасет до нормализации на рисунке 6.

```
# Нормализованный датасет (после нормализации значения принимают диапазон от 0 до 1)
df_minmax.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	13.0	0.465175	0.294974	0.0	0.387657	0.405059	0.593628	1.0
Плотность, кг/м3	13.0	0.381868	0.293562	0.0	0.178571	0.357143	0.535714	1.0
модуль упругости, ГПа	13.0	0.364223	0.291687	0.0	0.179487	0.329364	0.442685	1.0
Количество отвердителя, м.%	13.0	0.923077	0.277350	0.0	1.000000	1.000000	1.000000	1.0
Содержание эпоксидных групп,%_2	13.0	0.076923	0.277350	0.0	0.000000	0.000000	0.000000	1.0
Температура вспышки, С_2	13.0	0.923077	0.277350	0.0	1.000000	1.000000	1.000000	1.0
Поверхностная плотность, г/м2	13.0	0.354808	0.388994	0.0	0.000000	0.212500	0.325000	1.0
Модуль упругости при растяжении, ГПа	13.0	0.471154	0.388039	0.0	0.000000	0.416667	0.625000	1.0
Прочность при растяжении, МПа	13.0	0.472222	0.416306	0.0	0.166667	0.546296	1.000000	1.0
Потребление смолы, г/м2	13.0	0.529915	0.354727	0.0	0.555556	0.555556	0.555556	1.0
Шаг нашивки	13.0	0.692308	0.332263	0.0	0.500000	0.833333	1.000000	1.0
Плотность нашивки	13.0	0.494983	0.357007	0.0	0.434783	0.434783	0.565217	1.0

Рисунок 7 — Очищенный датасет после нормализации.

После применения нормализации к нашим данным, значения в выборке изменились, как видно на рисунках 6 и 7. Это позволит нам приступить к обучению моделей, разделению данных на тренировочные и тестовые и применению различных методов обучения. На данном этапе мы не будем применять стандартизацию, так как она не требуется в нашей работе. Далее мы переходим к решению задачи, используя матрицу корреляции, построенную на основе очищенного и отмасштабированного датасета. Это позволило обнаружить некоторые сильные зависимости между признаками, которые представлены на рисунке 8.

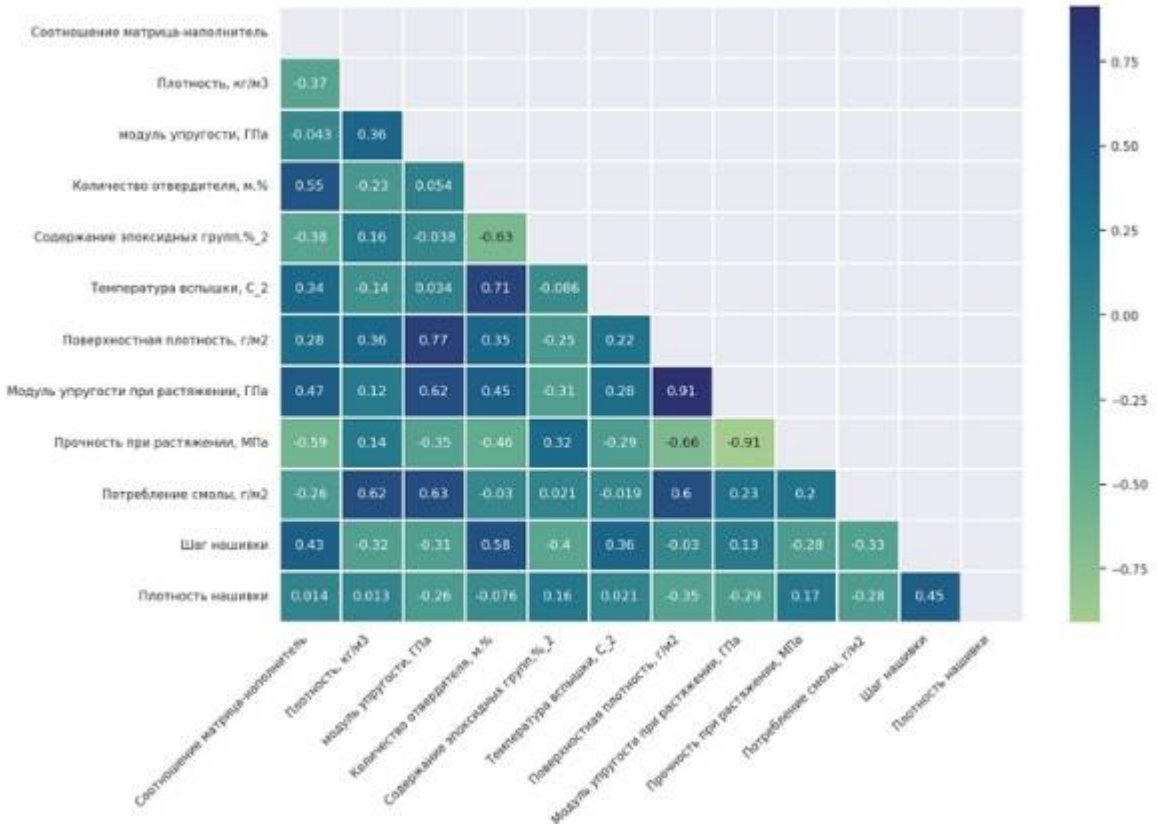

Рисунок 8 — Матрица корреляции на основе очищенного и отмасштабированного датасета

Мы обнаружили явные зависимости между некоторыми признаками на основе построенной матрицы корреляции из очищенного и отмасштабированного датасета. Несмотря на то, что у нас мало данных, они все же пригодны для обучения моделей, так как имеют сильную корреляцию. Однако, стоит отметить, что модели, полученные на таких данных, не будут пригодны для промышленного использования. В любом случае, мы применим 10 методов (включая нейросеть) для прогнозирования трех целевых признаков:

– «Модуль упругости при растяжении, ГПа»;

– «Прочность при растяжении, МПа»;

– «Соотношение матрица-наполнитель».

2. Практическая часть

2.1 Предобработка данных

Предварительная обработка данных также может включать в себя работу с выбросами, заполнение пропущенных значений, кодирование категориальных признаков и другие действия, зависящие от конкретного набора данных и задачи машинного обучения.

Графики ядерной оценки плотности (Kernel Density Estimate) могут помочь нам оценить форму распределения наших признаков, как до, так и после нормализации данных. Это может быть полезно для выбора соответствующей модели машинного обучения, которая может работать лучше с определенными типами распределений.

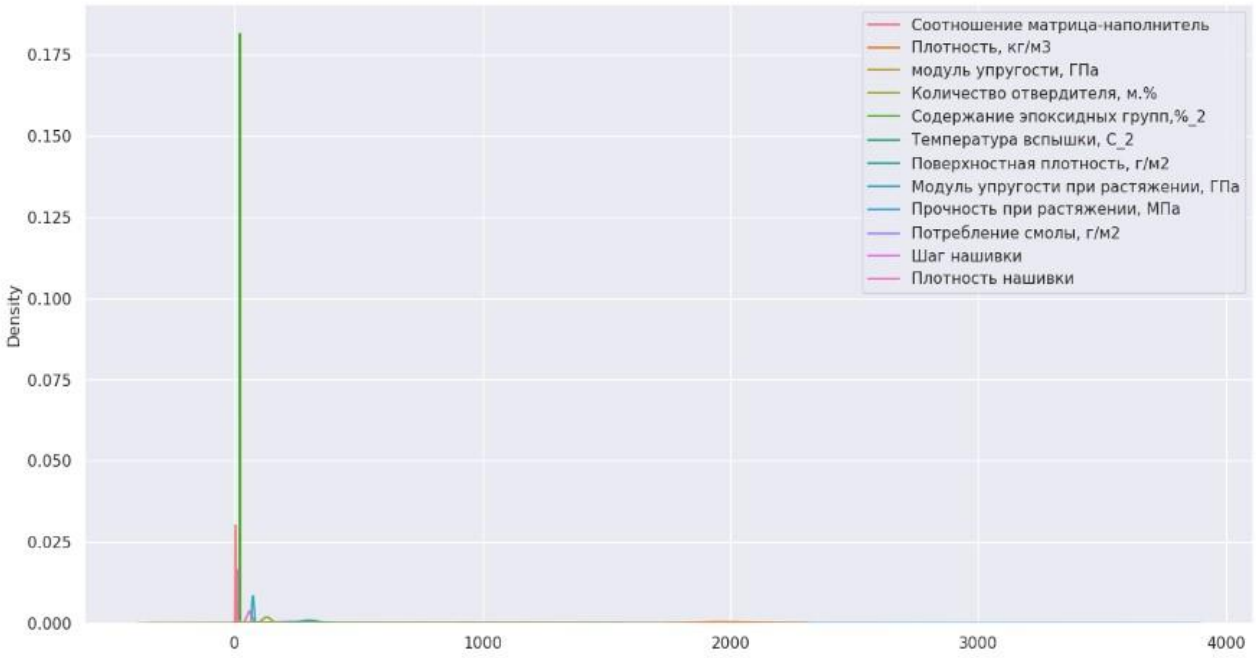

Рисунок 9 — График ядерной оценки плотности признаков очищенного датасета до нормализации

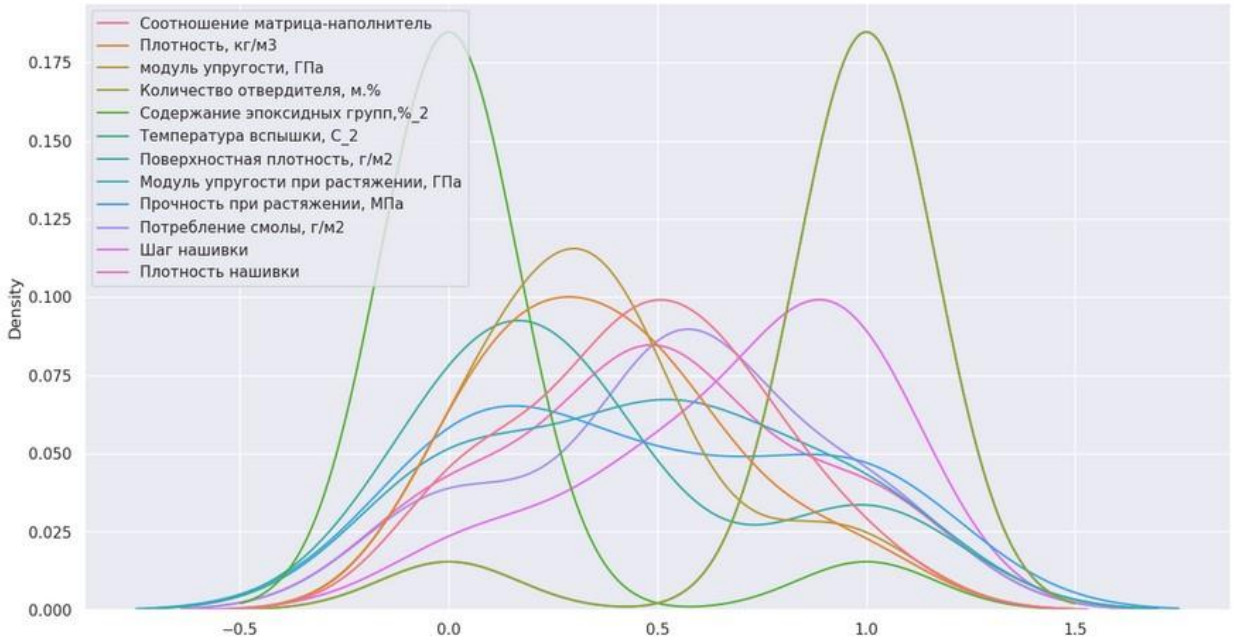

Рисунок 10 — График ядерной оценки плотности признаков очищенного датасета после нормализации

2.2 Разработка, обучение и тестирование десяти моделей (включая нейросеть) для прогнозирования трёх целевых признаков по отдельности.

1. Для прогнозирования признака «Модуль упругости при растяжении, ГПа»;

Правильно, разделение данных на входную и выходную части, а затем на тренировочную и тестовую выборки является важным шагом при обучении моделей машинного обучения. Это помогает измерить производительность моделей на новых, ранее не виданных данных и позволяет определить, насколько хорошо модели обобщают данные. Разделения выборки на входную, выходную, а также, тренировочную и тестовую часть представлены на рисунке 11.

```
#X_upr = df_minmax.drop(['Модуль упругости при растяжении, ГПа', 'Угол нашивки, град'], axis=1)
X_upr = df_minmax.drop(['Модуль упругости при растяжении, ГПа'], axis=1)
y_upr = df_minmax[['Модуль упругости при растяжении, ГПа']]
```

```
# Разбиваем на обучающую и тестовую выборки
X_train_upr, X_test_upr, y_train_upr, y_test_upr = train_test_split(X_upr, y_upr, test_size=0.1, random_state=15)
```

Рисунок 11 — Разделение выборки на входное, выходное, тренировочное и тестовое множество.

На рисунке 12 показаны размерности полученных наборов данных после разделения на входную и выходную части, а также на тренировочную и тестовую части.

```
# Проверяем размерность получившихся выборок
print(X_train_upr.shape, X_test_upr.shape, y_train_upr.shape, y_test_upr.shape)
```

```
(11, 11) (2, 11) (11, 1) (2, 1)
```

Рисунок 12 — Размерности тренировочных и тестовых множеств после разбиения

После разделения данных на входные и выходные части, а также на тренировочный и тестовый наборы, мы перейдем к созданию и обучению моделей. Затем мы протестируем и сравним все модели сразу.

Из раздела 1.2 нам известны все методы, которые будут применяться. Некоторые из них являются непараметрическими и полностью зависят от данных, не имея гибких настроек для улучшения результатов модели для конкретной задачи путём подбора гиперпараметров. В нашей работе используются как параметрические, так и непараметрические методы. Рассмотрим примеры процесса обучения различных моделей на рисунке 13, включая классическую линейную регрессию (LinearRegression).[8]

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

Линейная регрессия

```
#Линейная регрессия.
lr = LinearRegression()
lr.fit(X_train_upr, y_train_upr.iloc[:, 0])
lr_pred_upr = lr.predict(X_test_upr)
lr_mae_upr = mean_absolute_error(lr_pred_upr, y_test_upr)
lr_mse_upr = mean_squared_error(y_test_upr, lr_pred_upr)
lr_r2_train_upr = lr.score(X_train_upr, y_train_upr)
lr_r2_upr = r2_score(y_test_upr, lr_pred_upr)
print('LinearRegression Results')
print(f'mae: {lr_mae_upr.round(3)}')
print(f'mse: {lr_mse_upr.round(3)}')
print(f'R2 на тренировочной выборке: {lr_r2_train_upr.round(3)}')
print(f'R2 на тестовой выборке: {lr_r2_upr.round(3)}')
```

```
LinearRegression Results
mae: 0.0
mse: 0.0
R2 на тренировочной выборке: 1.0
R2 на тестовой выборке: 1.0
```

Рисунок 13 — Обучение модели методом линейной регрессии для предсказания признака «Модуль упругости при растяжении, ГПа»

На рисунке 14 приведен пример процесса обучения модели с использованием метода дерева решений (DecisionTreeRegressor), в котором был произведен подбор гиперпараметров.

Деревья решений (DecisionTreeRegressor) для прогноза

```
dtr_param_grid = {'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
                  'splitter': ['best', 'random'],
                  'max_depth': [1, 2, 3, 4, 5, 7, 9, 12],
                  'max_features': range(1, 15, 1)}

dtr_gs_upr = GridSearchCV(DecisionTreeRegressor(), dtr_param_grid, cv=5,
                          n_jobs=-1, verbose=1)

dtr_gs_upr.fit(X_train_upr, y_train_upr.iloc[:, 0])
print(f"Лучшие параметры для предсказания признака 'Модуль упругости при растяжении' {dtr_gs_upr.best_params_}")
```

```
Fitting 5 folds for each of 896 candidates, totalling 4480 fits
Лучшие параметры для предсказания признака 'Модуль упругости при растяжении' {'criterion': 'squared_error', 'max_depth': 4, 'max_features': 6, 'splitter': 'best'}
```

```
dtr = DecisionTreeRegressor(criterion='squared_error', max_depth=1, max_features=1, splitter='best')
dtr.fit(X_train_upr, y_train_upr.iloc[:, 0])
dtr_pred_upr = dtr.predict(X_test_upr)
dtr_mae_upr = mean_absolute_error(dtr_pred_upr, y_test_upr)
dtr_mse_upr = mean_squared_error(y_test_upr, dtr_pred_upr)
dtr_r2_train_upr = dtr.score(X_train_upr, y_train_upr)
dtr_r2_upr = r2_score(y_test_upr, dtr_pred_upr)
print('DecisionTreeRegressor Results')
print(f'mae: {dtr_mae_upr.round(3)}')
print(f'mse: {dtr_mse_upr.round(3)}')
print(f'R2 на тренировочной выборке: {dtr_r2_train_upr.round(3)}')
print(f'R2 на тестовой выборке: {dtr_r2_upr.round(3)}')
```

```
DecisionTreeRegressor Results
mae: 0.312
mse: 0.098
R2 на тренировочной выборке: 0.664
R2 на тестовой выборке: 0.0
```

Рисунок 14 — Подбор гиперпараметров и обучение модели дерева решений для предсказания признака «Модуль упругости при растяжении, ГПа»

Для поиска наилучших гиперпараметров мы использовали метод GridSearchCV из библиотеки sklearn. Мы передали этому методу выбранный нами регрессор и набор параметров, которые он последовательно передал регрессору, обучал модель и определял лучшую комбинацию параметров. Затем мы установили полученные гиперпараметры в наш регрессор и сохранили модель.

Мы проектируем и обучаем все модели, после чего проводим сравнение результатов каждой из них и выбираем лучшую, которую затем сохраняем и интегрируем в наше веб-приложение для прогнозирования значений целевых признаков. В задаче оценки качества моделей используется множество различных метрик.

В данной работе мы используем следующие метрики:

– MAE (Mean Absolute Error) средняя абсолютная ошибка — среднее абсолютное отклонение между прогнозируемыми значениями модели и реальными значениями на тестовых данных;

– MSE (Mean Squared Error) среднеквадратичная ошибка — среднее арифметическое квадратов разностей между предсказанными и реальными значениями модели машинного обучения;

– R2 или коэффициент детерминации — это статистический показатель, который используется для измерения того, насколько хорошо модель подходит для данных. Он определяет долю дисперсии в данных, которую модель может объяснить. Значение R2 находится в диапазоне от 0 до 1, где 1 означает идеальное соответствие модели с данными, а 0 означает полное расхождение. Если значение R2 низкое, то это может означать, что модель не соответствует данным, либо что данные сами по себе имеют большую ошибку. Отрицательное значение коэффициента детерминации означает, что разработанная вами модель даёт прогноз даже хуже, чем простое усреднение.

Результаты работы наших моделей, для прогноза признака «Модуль упругости при растяжении, ГПа», в виде таблицы, представлены на рисунке 15, а в виде столбчатых диаграмм на рисунке 16.

	Регрессор	MAE	MSE	R2_train	R2
0	LinearRegression	3.260124e-15	1.393587e-29	1.000000	1.000000
5	KNeighbors	0.000000e+00	0.000000e+00	1.000000	1.000000
1	LCV	4.155481e-04	4.155481e-04	0.999998	0.999938
3	ElasticNet	9.933212e-04	9.933212e-04	0.999992	0.999663
2	Ridge	4.147117e-02	4.147117e-02	0.989178	0.447135
4	RandomForest	8.333333e-02	8.333333e-02	0.663423	-0.125000
9	Нейросеть	1.217349e-01	1.510862e-02	0.916144	-1.447597
8	DecisionTree	3.726852e-01	3.726852e-01	0.542693	-32.751302
7	GradientBoosting	4.647840e-01	4.647840e-01	0.023921	-34.995908
6	SVR	5.351852e-01	5.351852e-01	-0.027835	-46.400552

Рисунок 15 — Результаты моделей для прогноза признака «Модуль упругости при растяжении, ГПа», в виде таблицы

Наилучший результат коэффициента детерминации R2 равен единице, что говорит о том, что наша модель идеально подходит для решения задачи. Наихудший результат показала модель, основанная на методе опорных векторов.

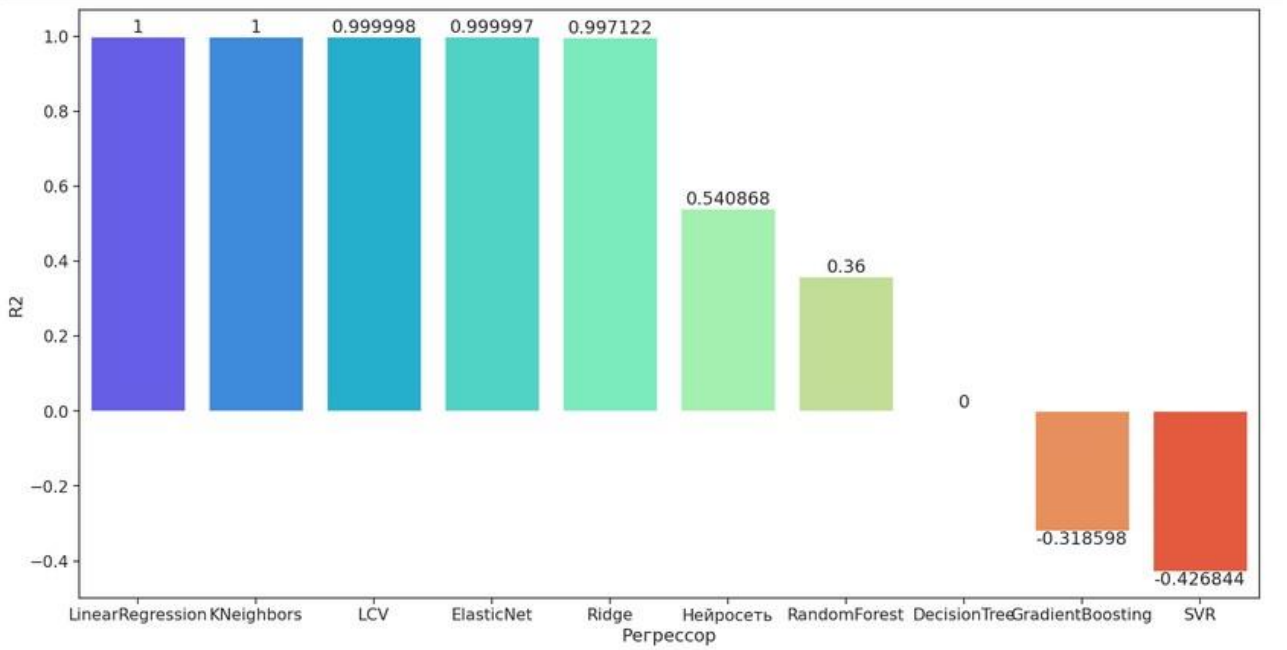

Рисунок 16 — Результаты моделей для прогноза признака «Модуль упругости при растяжении, ГПа», в виде столбчатых диаграмм

На основе результатов, можно заключить, что мы успешно разработали модель, способную прогнозировать целевой признак "Модуль упругости при растяжении, ГПа". Теперь мы переходим к следующей задаче - созданию моделей для прогнозирования признака "Прочность при растяжении, МПа".

2. Для прогнозирования признака «Прочность при растяжении, МПа»;

В этом разделе мы проводим аналогичные действия, что и в предыдущем разделе, но на этот раз нашей целевой переменной является признак "Прочность при растяжении, МПа". Мы также покажем, как мы разделили нашу выборку для этой задачи и сразу перейдем к результатам наших моделей.

На рисунке 17 представлено разделение нашей выборки для прогноза признака «Прочность при растяжении, МПа».

Разделение на обучающую и тестовую выборки для прогноза прочности при растяжении

```
X_pr = df_minmax.drop(['Прочность при растяжении, МПа'], axis=1)
y_pr = df_minmax[['Прочность при растяжении, МПа']]
X_train_pr, X_test_pr, y_train_pr, y_test_pr = train_test_split(X_pr, y_pr, test_size=0.2, random_state=1)
```

```
print(X_train_pr.shape, X_test_pr.shape, y_train_pr.shape, y_test_pr.shape)
```

```
(10, 11) (3, 11) (10, 1) (3, 1)
```

Рисунок 17 — Разделение выборки для прогноза признака «Прочность при растяжении, МПа»

Результаты работы моделей, прогнозирующих «Прочность при растяжении, МПа», в виде таблицы, приведены на рисунке 18.

	Перрессор	MAE	MSE	R2_train	R2
0	LinearRegression	3.260124e-15	1.393587e-29	1.000000	1.000000
5	KNeighbors	0.000000e+00	0.000000e+00	1.000000	1.000000
1	LCV	4.155481e-04	4.155481e-04	0.999998	0.999938
3	ElasticNet	9.933212e-04	9.933212e-04	0.999992	0.999663
2	Ridge	4.147117e-02	4.147117e-02	0.989178	0.447135
4	RandomForest	8.333333e-02	8.333333e-02	0.663423	-0.125000
9	Нейросеть	1.217349e-01	1.510862e-02	0.916144	-1.447597
8	DecisionTree	3.726852e-01	3.726852e-01	0.542693	-32.751302
7	GradientBoosting	4.647840e-01	4.647840e-01	0.023921	-34.995908
6	SVR	5.351852e-01	5.351852e-01	-0.027835	-46.400552

Рисунок 18 — Результаты моделей, прогнозирующих «Прочность при растяжении, МПа», в виде таблицы

Рисунок под номером 19 отображает результаты тех же моделей, но в виде столбчатых диаграмм.

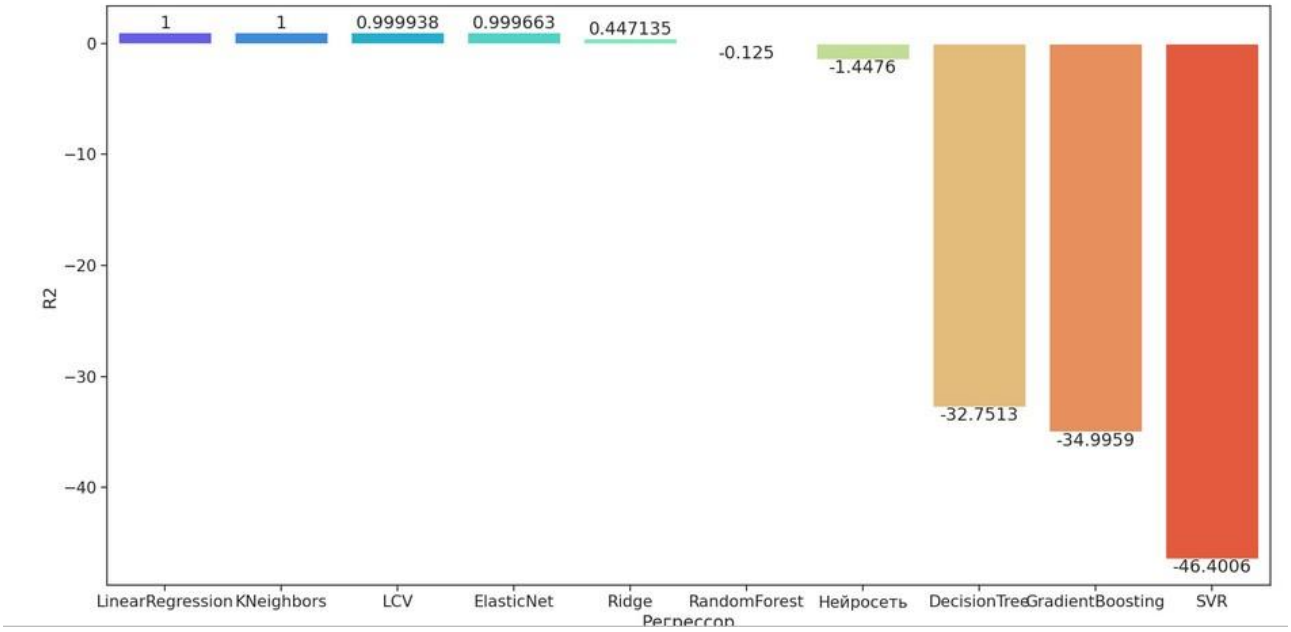

Рисунок 19 — Результаты моделей, прогнозирующих «Прочность при растяжении, МПа», в виде столбчатых диаграмм

Полученные результаты отличаются от предыдущих, поскольку помимо модели с линейной регрессией, модель, основанная на методе KNeighbors, достигла коэффициента детерминации R2, равного единице. В общем, ситуация улучшилась для большинства других методов.

2.3 Написание нейронной сети для прогнозирования соотношения матрица-наполнитель

Мы решили использовать библиотеку tensorflow для построения нейросети. Наша нейронная сеть будет полносвязной и состоять из двух скрытых слоев с 8 нейронами в каждом. В качестве функции активации для скрытых слоев мы выбрали "relu", а для выходного слоя - "linear". Мы выбрали данную архитектуру, поскольку она дала наилучший результат для решения нашей задачи.

В качестве оптимизатора берём «Adam», в качестве функции потерь берём «mse».

Архитектура нейросети приведена на рисунке 20.

```
print(net_mn.summary())
```

```
Model: "sequential_34"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_120 (Dense)           (None, 128)               1536

 dense_121 (Dense)           (None, 8)                 1032

 dense_122 (Dense)           (None, 8)                 72

 dense_123 (Dense)           (None, 1)                 9

=================================================================
Total params: 2,649
Trainable params: 2,649
Non-trainable params: 0
_________________________________________________________________
None
```

Рисунок 20 — Архитектура нейросети

Запускаем обучение нейросети :

График обучения нейросети приведен на рисунке 21.

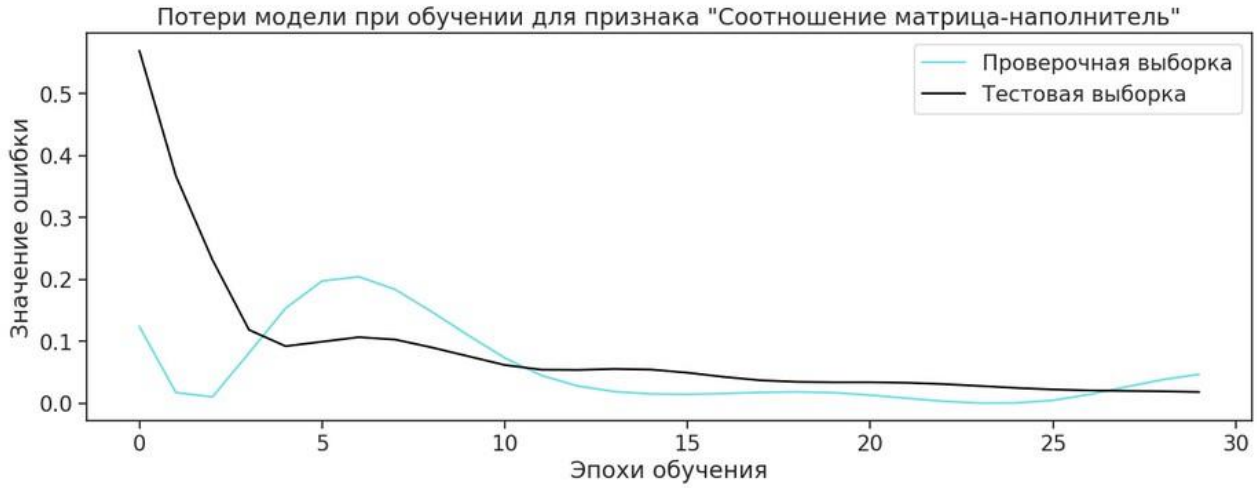

Рисунок 21 — График обучения нейросети

Из графика видно, что значение ошибки на валидационной выборке стремится к значению ошибки на тестовой выборке с каждой эпохой, что говорит в пользу эффективности нейросети.

На рисунке 22 представлены результаты тестирования нейросети и других моделей, используемых для прогнозирования признака «Соотношение матрица-наполнитель».

	Регрессор	MAE	MSE	R2_train	R2
6	SVR	0.096217	0.096217	-0.019204	-0.163373
7	GradientBoosting	0.095979	0.095979	0.094217	-0.435477
3	ElasticNet	0.125405	0.125405	0.669923	-1.302248
2	Ridge	0.095314	0.095314	0.676002	-1.371421
1	LCV	0.134678	0.134678	0.680203	-1.601210
5	KNeighbors	0.134739	0.134739	1.000000	-2.206340
8	DecisionTree	0.205382	0.205382	0.495878	-5.561308
4	RandomForest	0.225341	0.225341	0.492243	-7.583770
9	Нейросеть	0.231584	0.077735	0.806667	-8.739433
0	LinearRegression	1.105978	1.715571	0.979646	-213.943852

Рисунок 22 — Результаты тестирования нейросети и всех остальных моделей для прогноза признака «Соотношение матрица-наполнитель»

Все модели не смогли предсказать признак "Соотношение матрица-наполнитель", как показывает отрицательный коэффициент детерминации. Однако, мы успешно разработали нейронную сеть, которая рекомендует определенное соотношение матрица-наполнитель, выполнив тем самым нашу задачу.

Мы не можем считать результат работы нашей нейронной сети для данной задачи удовлетворительным, однако, учитывая крайне ограниченный объем доступных пригодных данных, мы можем быть довольны полученными результатами. Поэтому мы переходим к следующему разделу, где мы расскажем о функционале нашего веб-приложения и предоставим краткую инструкцию по его использованию.

2.4 Разработка приложения

Для разработки приложения мы использовали библиотеку Flask, которая позволила нам создать и реализовать функционал нашего приложения. Кроме того, мы также применили библиотеки tensor-flow и pickle для интеграции наших моделей в приложение. После запуска приложения мы переходим на главную страницу, где находится функция "Рекомендация соотношения матрица-наполнитель". Для получения рекомендации необходимо заполнить все поля числовыми значениями и нажать кнопку "Рассчитать", расположенную ниже полей. После этого мы получаем прогнозируемое значение для нашего признака. Таким образом, этот раздел можно считать завершенным. В приложении А приведены скриншоты разработанного веб-приложения.

2.5 Создание удаленного репозитория

В дополнение к данной работе, был создан удаленный репозиторий на GitHub, который находится по адресу https://github.com/NikitaShtankov/Diplom. На данном ресурсе были предоставлены все необходимые материалы для выполнения нашего задания, включая исследовательский ноутбук, пояснительную записку, файлы приложения и необходимые файлы для работы с ноутбуком.

Заключение

В итоге данной работы были выполнены все поставленные задачи. Было создано много моделей, которые были обучены и сравнены между собой. Одна из моделей была внедрена в веб-приложение для предсказания "Рекомендация соотношения матрица-наполнитель"(https://application-flask.onrender.com/index)

Однако, следует отметить, что данные модели не пригодны для использования в промышленной среде, так как на этапе препроцессинга было урезано практически весь датасет, и осталось очень мало данных для обучения моделей.

В процессе работы мы изучили множество различных статистических методов, которые используются для прогноза. Мы применили каждый выбранный метод в наших моделях, подобрали гиперпараметры и сравнили результаты работы каждой модели в виде таблиц и графиков с диаграммами.

Также мы освоили множество методов и библиотек на языке программирования Python для решения различных задач. Мы научились создавать нейросети, подбирать архитектуры для разных задач и создавать приложения для браузера, а также оформлять их.

Список используемой литературы

1. Банкрашков, А.В. Программирование для детей на языке Python / А.В. Банкрашков. - М.: АСТ, 2018. - 288 с.
2. Вордерман, К. Программирование на Python. Иллюстрированное руководство для детей / К. Вордерман, К. Стили, К. Квигли. - М.: Манн, Иванов и Фербер, 2017. - 346 с.
3. Композиционные материалы : учебное пособие для вузов / Д. А. Иванов, А. И. Ситников, С. Д. Шляпин ; под редакцией А. А. Ильина. — Москва : Издательство Юрайт, 2019 — 253 с. — (Высшее образование). — Текст : непосредственный.
4. Лутц, М. Программирование на Python. Т. 2 / М. Лутц. - М.: Символ, 2016. - 992 с.
5. Лутц, М. Программирование на Python. Т. 1 / М. Лутц. - М.: Символ, 2016. - 992 с.
6. Лутц, М. Программирование на Python, II том / М. Лутц. - СПб.: Символ-плюс, 2015. - 992 с.
7. Лутц, М. Программирование на Python, I том / М. Лутц. - СПб.: Символ-плюс, 2015. - 992 с.
8. Лутц, М. Программирование на Python т.2 / М. Лутц. - М.: Символ-Плюс, 2011. - 992 с.
9. МакГрат, М. Программирование на Python для начинающих / М. МакГрат. - М.: Эксмо, 2015. - 192 с.
10. Мэтиз, Э. Изучаем PYTHON.Программирование игр, визуализация данных, веб-приложения / Э. Мэтиз. - СПб.: Питер, 2017. - 496 с.
11. Мэтиз, Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения / Э. Мэтиз. - СПб.: Питер, 2017. - 320 с.
12. Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. - М.: Символ-Плюс, 2011. - 608 с.
13. Саммерфилд, М. Программирование на Python 3. Подробное руководство / М. Саммерфилд. - М.: Символ, 2016. - 608 с.

14. Рибаник, В. Л. Справочник для программиста: – Режим доступа: http://www.codenet.ru/webmast/php/php3/php3_45.php. (дата обращения: 29.02.2011).

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

16. ГрасД. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.

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

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

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

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

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

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

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

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

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

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

27. Alex Maszański. Метод k-ближайших соседей (k-nearest neighbour): – Режим доступа: https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19.

28. Yury Kashnitsky. Открытый курс машинного обучения. Тема 3. Классификация, деревья решений и метод ближайших соседей: – Режим доступа: https://habr.com/ru/company/ods/blog/322534/.

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

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

31. Alex Maszański. Решаем задачи машинного обучения с помощью алгоритма градиентного бустинга: – Режим доступа: https://proglib.io/p/reshaem-zadachi-mashinnogo-obucheniya-s-pomoshchyu-algoritma-gradientnogo-bustinga-2021-11-25.

Приложение А. Скриншоты веб-приложения

Скриншоты веб-приложения, иллюстрирующие его работу. Реализованы были следующие функции:

– Рекомендация соотношения матрица-наполнитель;

– Ввод входных параметров;