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 Брянцева Юлия Юрьевна

Москва, 2022

Содержание

Введение3

1.Аналитическая часть5

1.1.Постановка задачи5

1.2.Описание используемых методов 6

1.3.Разведочный анализ данных14

2.Практическая часть19

2.1.Предобработка данных19

2.2.Разработка и обучение модели 21

2.3.Тестирование модели22

2.4.Разработка нейронной сети31

2.5.Разработка приложения37

3.Использование репозитория38

Заключение39

4.Библиографический список40

**Введение**

Пояснительная записка подготовлена в рамках выпускной квалификационной работы по курсу «Data Science». При выполнении данной работы проводился анализ данных с использованием методов машинного обучения, реализованных в программной среде Python. Исследуемый набор данных содержал значения определенных параметров компонент композиционных материалов.

Композицио́нный материа́л — неоднородный сплошной материал, состоящий из двух или более компонентов, среди которых можно выделить армирующие элементы, обеспечивающие необходимые механические характеристики материала, и матрицу (или связующее), обеспечивающую совместную работу армирующих элементов.

Механическое поведение композита определяется соотношением свойств армирующих элементов и матрицы, а также прочностью связи между ними. Эффективность и работоспособность материала зависят от правильного выбора исходных компонентов и технологии их совмещения, призванной обеспечить прочную связь между компонентами при сохранении их первоначальных характеристик.

В результате совмещения армирующих элементов и матрицы образуется комплекс свойств композита, не только отражающий исходные характеристики его компонентов, но и включающий свойства, которыми изолированные компоненты не обладают. В частности, наличие границ раздела между армирующими элементами и матрицей существенно повышает трещиностойкость материала, и в композитах, в отличие от металлов, повышение статической прочности приводит не к снижению, а, как правило, к повышению характеристик вязкости разрушения.

Композиционные материалы (КМ) - это материалы, обладающие следующей совокупностью признаков:

* состоят из двух или более компонентов, различающихся по своему химическому составу и разделенных выраженной границей;
* имеют новые свойства, отличающиеся от свойств, составляющих эти материалы компонентов;
* неоднородны в микромасштабе и однородны в макромасштабе;
* свойства определяются каждым из компонентов, которые в связи с этим должны содержаться в материале в достаточно большом количестве (больше некоторого критического значения).

Традиционно разработка композитных материалов – это долгосрочный процесс, так как по характеристикам отдельных компонентов невозможно рассчитать итоговые свойства композита. Для получения заданных свойств требуется большое количество испытаний различных комбинаций, что делает актуальной задачу прогнозирования успешное решение которой позволило бы снизить расходы и трудозатраты по разработке новых материалов.

С появлением композиционных материалов стал возможным селективный выбор свойств композитов, необходимых для каждой конкретной области их применения, и возникла потребность в проектировании таких материалов. Процесс создания композиционного материала включает следующие стадии: формирование проектных исходных данных; выбор состава композита и технологии его производства; оценка основных свойств созданного материала и сравнение их с заданием на проектирование. Для оценки основных свойств нового материала 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 (матрицы и армирующих компонентов).

1. **Аналитическая часть**
   1. **Постановка задачи**

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

**Входные данные:** начальные свойства компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.).

**На выходе** необходимо спрогнозировать три параметра: модуль упругости при растяжении, прочность при растяжении, соотношение матрица-наполнитель.

Поставленная задача в рамках классификации категорий машинного обучения относится к машинному обучению с учителем, задача регрессии.

Анализ, предобработка данных, построение моделей выполнены посредством языка программирования Python с использованием библиотек Pandas, Matplotlib и Sklearn.

Порядок выполнения выпускной работы:

1. Изучение теоретических основ и методов решения поставленной задачи;
2. Проведение разведочного анализа данных, в т.ч.:
3. Создание для каждой переменной гистограмм распределения и диаграмм «ящиков с усами», попарных графиков рассеяния точек;
4. Получение для каждой колонки датасета среднего и медианного значение;
5. Проведение анализа и исключения выбросов;
6. Проверка на наличие пропусков.
7. Проведение предобработки данных (удаление шумов, нормализация и т.д.);
8. Обучение нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении. При построении модели необходимо 30% данных оставить на тестирование модели, на остальных происходит обучение моделей. При построении моделей провести поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10;
9. Написание нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель;
10. Разработка приложения с графическим интерфейсом или интерфейсом командной строки, которое будет выдавать прогноз, полученный в задании 4 или 5 (один или два прогноза, на выбор);
11. Оценка точности модели на тренировочном и тестовом датасете;
12. Создание репозитория в GitHub/GitLab и размещение в нем кода исследования. Оформление файла README.
    1. **Описание используемых методов**

В данной работе решается типичная задача обучения с учителем - прогноз целевого числового значения переменной, имеющей связь с одной или несколькими независимыми переменными, называемыми также предикторами. Задачу подобного рода также называют регрессией.

При выполнении работы были использованы следующие алгоритмы машинного обучения:

1. линейная регрессия (linear regression),
2. случайный лес (random forest),
3. k ближайших соседей ((k-nearest neighbors),
4. градиентный бустинг (gradient boosting),
5. нейронная сеть (многослойный персептрон, MLP).

**Линейная регрессия.**

В модели линейной регрессии есть два вида переменных:

* вход или переменная предиктора (X) - переменная, которая помогает предсказать значение выходной переменной*.*
* выходная переменная (Y) - переменная, которую мы хотим предсказать.

Оценка *Y* выполняется через линейную функцию:

*Yₑ = α + β X*

где *Y*ₑ является оценочной или прогнозируемой стоимостью *Y* на основе нашего линейного уравнения.

Цель - найти статистически значимые значения параметра *α,* а также *β,* что минимизирует разницу между *Y*а и *Y*ₑ.

Если мы сможем определить оптимальные значения этих двух параметров, то у нас будет линия наилучшего соответствия, которую мы можем использовать для прогнозирования значений *Y*, учитывая значения *X*.

Преимущества:

* Просто и легко понять
* Дешевые вычислительные затраты
* Основа для более сложных алгоритмов машинного обучения

Недостатки:

* Чувствителен к выбросам и шумам
* Не может использоваться, когда связь между зависимой и независимой переменной не является линейной.

**Случайный лес.**

Это тип контролируемого алгоритма машинного обучения, основанного на ансамблевом обучении, при котором объединяют различные типы алгоритмов или один и тот же алгоритм несколько раз, чтобы сформировать более мощную модель прогнозирования. Данный алгоритм включает несколько алгоритмов одного и того же типа, т.е. несколько решений *деревьев*, в результате чего получается *лес деревьев*, отсюда и название “Случайный лес”. Алгоритм случайного леса может быть использован как для регрессионных, так и для классификационных задач.

Основные шаги, связанные с выполнением алгоритма случайного леса:

1. Выберите N случайных записей из набора данных.
2. Постройте дерево решений на основе этих N записей.
3. Выберите нужное количество деревьев в вашем алгоритме и повторите шаги 1 и 2.
4. В случае регрессионной задачи для новой записи каждое дерево в лесу предсказывает значение Y (выход). Конечное значение можно вычислить, взяв среднее значение всех значений, предсказанных всеми деревьями в лесу. Или, в случае проблемы классификации, каждое дерево в лесу предсказывает категорию, к которой принадлежит новая запись. Наконец, новый рекорд присваивается той категории, которая получает большинство голосов.

Преимущества:

* Не является предвзятым, поскольку существует несколько деревьев, и каждое дерево обучается на подмножестве данных. В принципе, алгоритм случайного леса опирается на силу “толпы”, поэтому общая предвзятость алгоритма уменьшается.
* Очень стабилен. Даже если новая точка данных введена в набор данных, общий алгоритм не сильно пострадает, так как новые данные могут повлиять на одно дерево, но ему очень трудно повлиять на все деревья.
* Хорошо работает, когда есть как категориальные, так и числовые признаки.
* Хорошо работает, когда данные имеют пропущенные значения или они не были хорошо.

Недостатки:

* Сложность. Этот метод требует гораздо больше вычислительных ресурсов из-за большого количества деревьев решений, соединенных вместе.
* Из-за своей сложности они требуют гораздо больше времени для обучения, чем другие сопоставимые алгоритмы.

**Метод k-ближайших соседей**

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

В случае использования метода для регрессии, объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны.

Для реализации метода необходима метрика расстояния между объектами. Используется, например, эвклидово расстояние для количественных признаков или расстояние Хэмминга для категориальных.

Этот метод — пример непараметрической регрессии.

Преимущества:

* Простая реализация;
* Как правило, метод хорош для первого решения задачи, причем не только классификации или регрессии, но и, например, рекомендации;

Недостатки:

* Метод считается быстрым в сравнении, например, с ансамблевыми алгоритмами, но в реальных задачах, как правило, число соседей, используемых для классификации, будет большим (100-150), и в таком случае алгоритм будет работать не так быстро, как дерево решений;
* Если в наборе данных много признаков, то трудно подобрать подходящие веса и определить, какие признаки не важны для классификации/регрессии;
* Зависимость от выбранной метрики расстояния между примерами;
* Нет теоретических оснований выбора определенного числа соседей - только перебор (впрочем, чаще всего это верно для всех гиперпараметров всех моделей). В случае малого числа соседей метод чувствителен к выбросам, то есть склонен переобучаться;
* Как правило, плохо работает, когда признаков много, из-за "проклятия размерности".

**Градиентный бустинг**

Градиентный бустинг - 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.).

Преимущества:

* Алгоритм работает с любыми функциями потерь.
* Предсказания в среднем лучше, чем у других алгоритмов.
* Самостоятельно справляется с пропущенными данными.

Недостатки:

* Алгоритм крайне чувствителен к выбросам и при их наличии будет тратить огромное количество ресурсов на эти моменты.
* Большие затраты времени на вычисления
* Необходимо грамотно подбирать гиперпараметры.

**Нейронный сети**

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

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

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

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

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

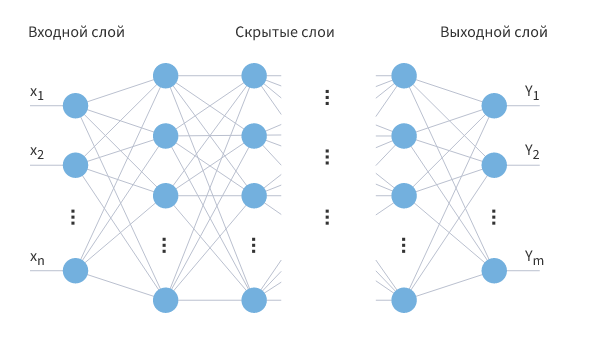
* входной слой - его размер соответствует входным параметрам;
* скрытые слои - их количество и размерность определяем специалист;
* выходной слой - его размер соответствует выходным параметрам.

Рисунок 1.1 – Нейронная сеть

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.

Многослойный персептрон — это класс искусственных нейронных сетей прямого распространения, состоящих как минимум из трех слоёв: входного, скрытого и выходного. За исключением входных, все нейроны использует нелинейную функцию активации. Необходимость в большом количестве обучаемых слоёв отпадает, так как теоретически единственного скрытого слоя достаточно, чтобы перекодировать входное представление таким образом, чтобы получить линейную разделимость для выходного представления. Существует предположение, что, используя большее число слоёв, можно уменьшить число элементов в них, то есть суммарное число элементов в слоях будет меньше, чем если использовать один скрытый слой.

Персептроны часто применяются для решения контролируемых задач обучения: они тренируются по набору пар входных/выходных объектов и учатся моделировать корреляции (т. е. зависимости) между этими данными. Обучение включает в себя настройку параметров модели (весовых коэффициентов, смещений) для минимизации погрешности. Для корректировки этих параметров относительно погрешности используется алгоритм обратного распространения, а сама погрешность может быть вычислена различными способами, в том числе путем вычисления среднеквадратичного отклонения (RMSE).

Сети прямого распространения, такие как многослойный персептрон, похожи на теннис или пинг-понг. Они в основном состоят из двух видов движений: вперед и назад. Получается своеобразная игра в пинг-понг между догадками и ответами, поскольку каждая догадка – это проверка того, что мы знаем, а каждый ответ – это обратная связь, позволяющая нам узнать, насколько сильно мы ошибаемся.

При шаге *вперед* поток сигнала перемещается от входного слоя через скрытые к выходному, а решение, полученное на выходном слое, сравнивается с априорно известным верным ответом.

При шаге *назад* с использованием правила дифференцирования сложных функций через персептрон в обратном направлении распространяются частные производные функции, погрешности по весовым коэффициентам и смещениям. Данный акт дифференцирования дает нам градиент погрешности, с использованием которого могут быть скорректированы параметры модели, так как они приближают МП на один шаг ближе к минимуму погрешности. Это можно сделать с помощью любого алгоритма градиентной оптимизации, например, методом стохастического градиентного спуска. Сеть продолжает играть в пинг-понг, пока погрешность не исчезнет. В этом случае, как говорят, наступает сходимость.

**Недостатки:**

Выигрывая в скорости работы путем вытравливания алгоритмов, вы проигрываете в возможности их модификации. Это является реальной проблемой в машинном обучении, где алгоритмы самопроизвольно изменяются по мере обработки данных. Задача состоит в том, чтобы найти те части алгоритма, которые остаются стабильными даже при изменении параметров, например, операции с линейной алгеброй, которые в настоящее время обрабатываются GPU быстрее всего.

* 1. **Разведочный анализ данных**

Для получения представления о характере распределения переменных в датасете, формирования оценки качества исходных данных (наличия пропусков, выбросов), выявления характера взаимосвязи между переменными с целью последующего выдвижения гипотез о наиболее подходящих для решения задачи моделях машинного обучения проведем разведочный анализ данных.

Входные данные предоставлены в виде двух Excel-файлов.

Файл X\_bp.xlxs содержит таблицу с 1023 наборами измерений десяти свойств композитов. Пустых значений нет, все переменные вещественного типа.

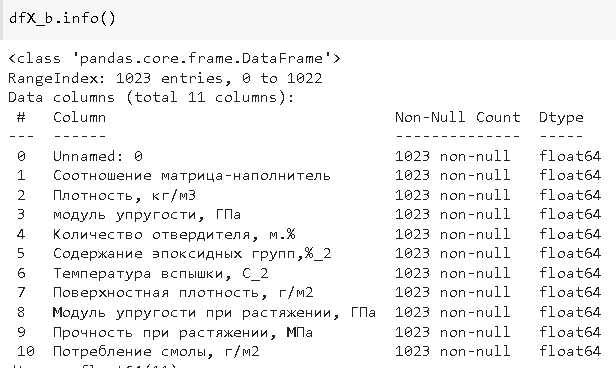


Рисунок 1.2 – Информация о таблице X\_bp

Файл X\_nup.xlxs содержит таблицу с 1040 наборами измерений трех свойств композитов. Пустых значений нет, все переменные вещественного типа.

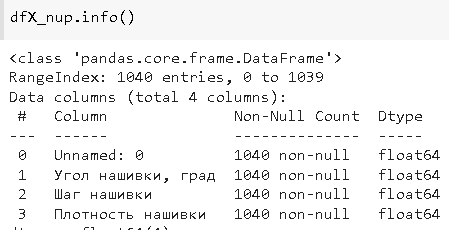


Рисунок 1.3 – Информация о таблице X\_nup

Согласно постановке задачи, эти таблицы необходимо объединить по индексам в единый датасет, используя тип объединения INNER. Таким образом, семнадцать наборов измерений из файла X\_nup.xlxs не были добавлены в единый датасет. Это составило 1,6% от общего числа данных в этом файле, что можно принять за несущественную потерю.



Рисунок 1.4 – Код объединения таблиц

В итоге искомый рабочий датасет состоит из 1023 строк (наборов измерений параметров композитов) и 13 колонок (параметров композитов). Три параметра в зависимости от решаемой задачи будут становится попеременно выходными прогнозируемыми переменными, а именно:

1. Модуль упругости при растяжении, Гпа;
2. Прочность при растяжении, Мпа;
3. Соотношение матрица-наполнитель.

В качестве инструментов разведочного анализа используется оценка статистических характеристик данных (см. Таблицу 1), а также матрица попарной корреляции, тепловая карта корреляции, гистограммы нормального распределения, поиск выбросов через ящик с усами. Таблица 1.1



Выполним проверку на наличие повторений (дубликатов данных). Дубликаты в объединённом датафрейме не были найдены – это видно на рисунке 1.5.

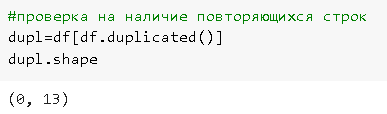


Рисунок 1.5 – Наличие дубликатов записей

Метод df\_join.nunique() возвращает количество уникальных значений для каждого столбца.

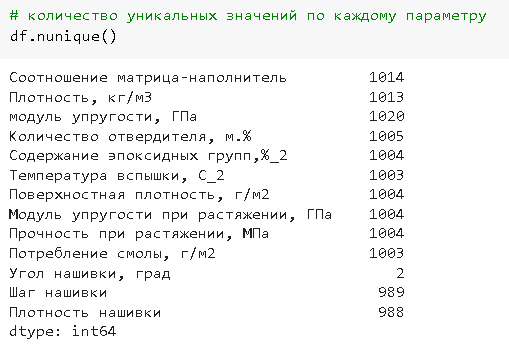


Рисунок 1.6 – Информация об уникальных значениях записей

На следующем этапе сделаем визуализацию данных.

Гистограммы распределения переменных отображены на Рисунке 1.7. Они показывают, что данные всех параметров, кроме *Поверхностной плотности* имеют распределение, близкое к нормальному*.* Параметр *Поверхностная плотность* имеет распределение со смещением влево, что говорит о преобладании данных с меньшим показателем поверхностной плотности. Для параметра *Угол нашивки*, имеющего только два значения, гистограмма и диаграмма размаха не выводились в виду их малой информативности.

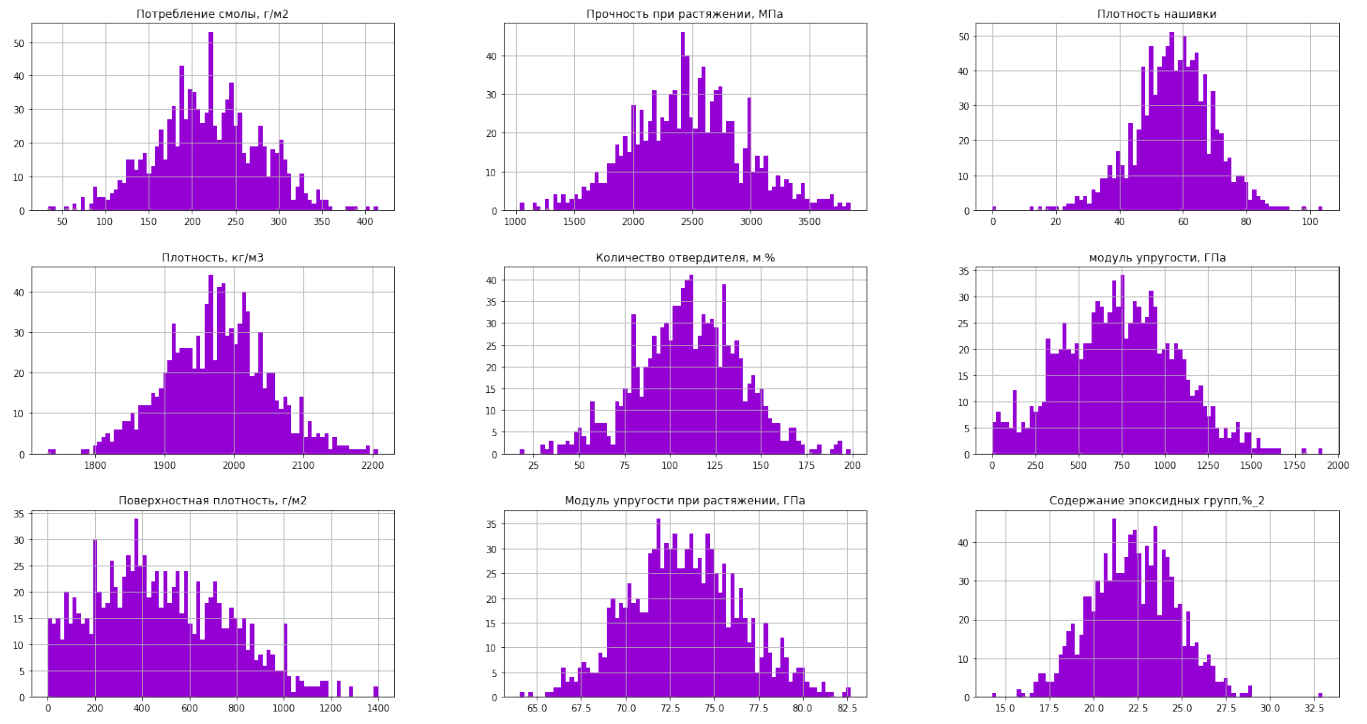


Рисунок 1.7 – Гистограммы распределения переменных

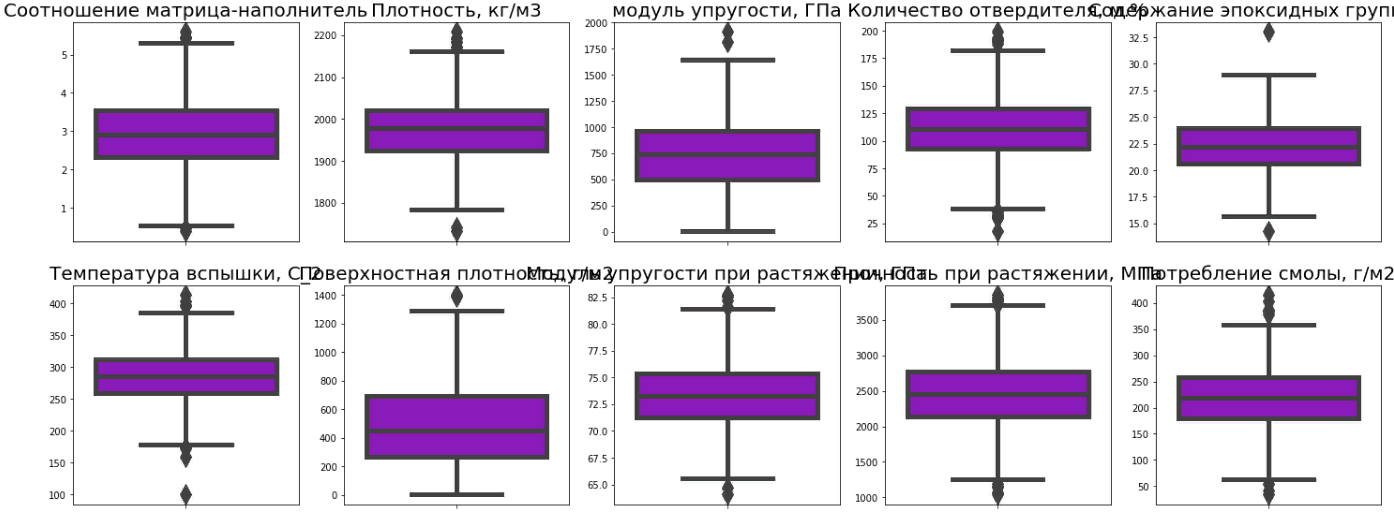


Рисунок 1.8 – Диаграммы «Ящик с усами»

### Диаграммы размаха, или "ящики с усами" приведены на Рисунке 1.8. Из них видно, что все параметры имеют выбросы, причем большинство из них с обоих сторон - как в наименьших, так и в наибольших значениях этих параметров.

Следующим шагом разведочного анализа построим попарные графики рассеяния точек и тепловую карту матрицы корреляции для визуализации наличия зависимости признаков.

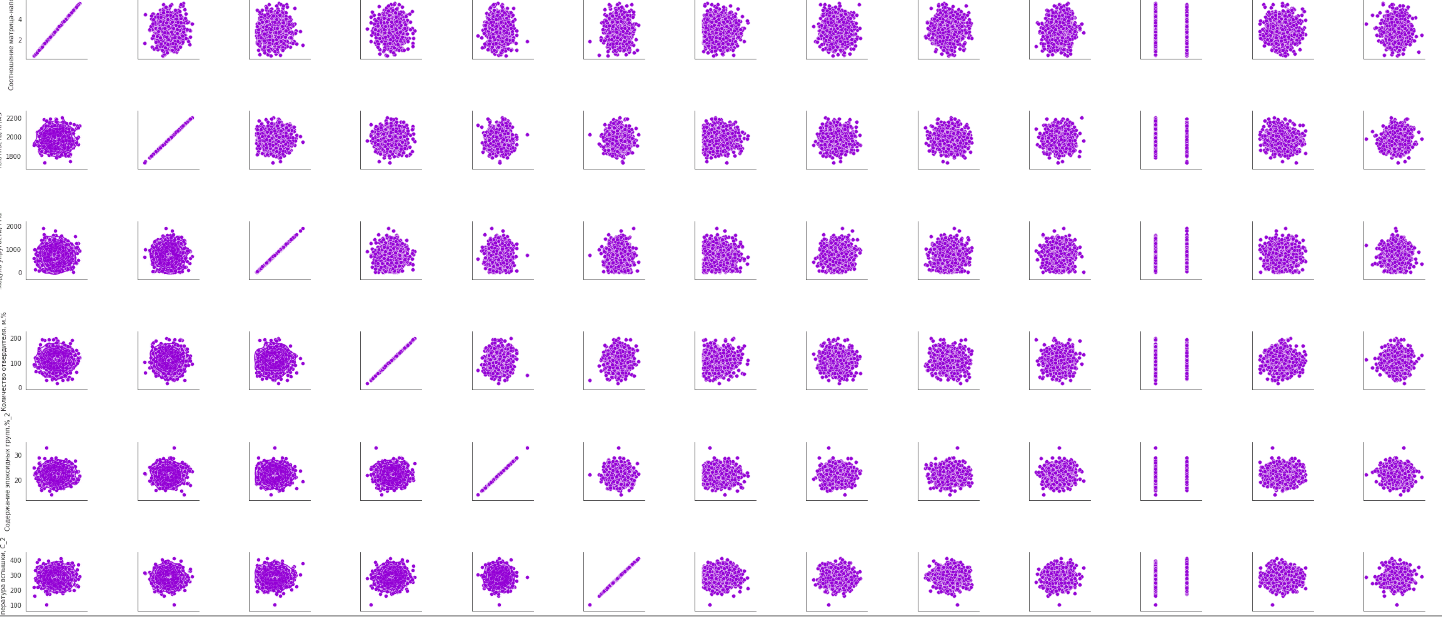


Рисунок 1.9 – Попарные графики рассеяния

Попарные графики рассеяния показывают отсутствие явно выраженной линейной зависимости между параметрами.

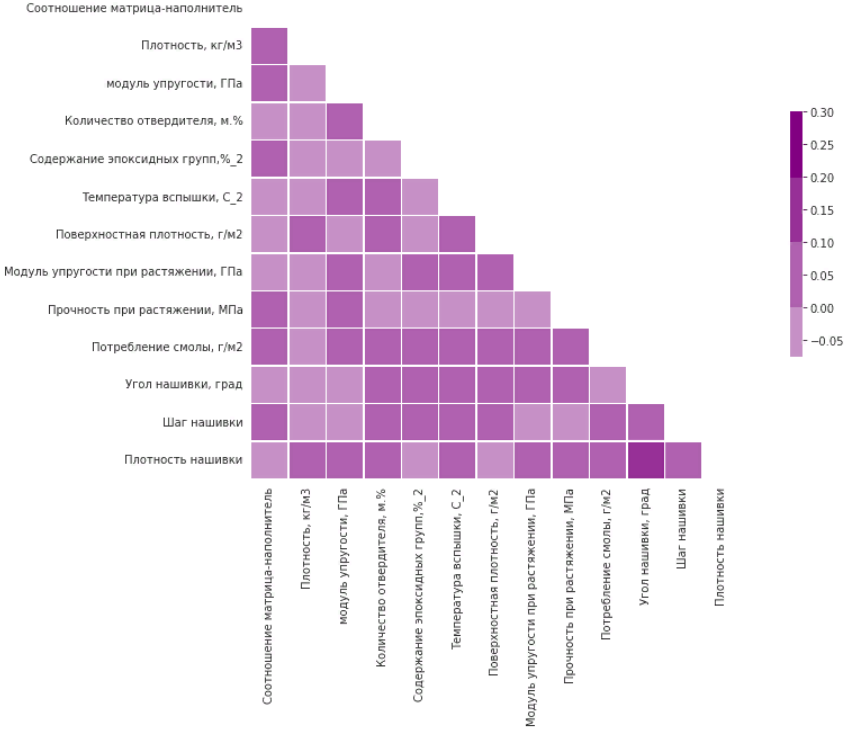


Рисунок 1.10 – Тепловая карта

По тепловой карте также видно, что зависимости между параметрами набора данных выражены очень слабо. На этом разведочный анализ данных завершен.

1. **Практическая часть**
   1. **Предобработка данных**

**Удаление выбросов**

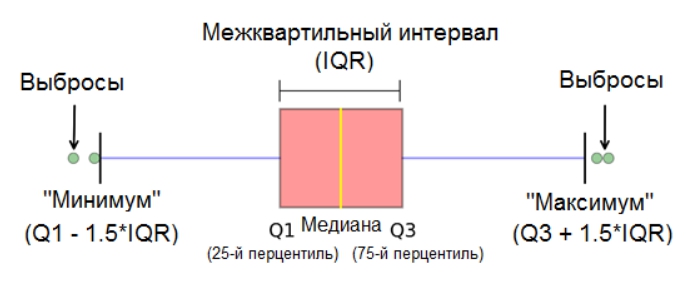
В ходе проведения разведочного анализа с помощью диаграмм размаха было визуально установлено наличие в рабочем наборе данных выбросов. Для их нахождения был использован метод межквартильных интервалов (InterQuartile Range, IQR).

Рисунок 2.1 Расчет выбросов по методу IQR

С его помощью было установлено, что в рабочем наборе данных имеется 93 выброса, 9% от общего количества данных.

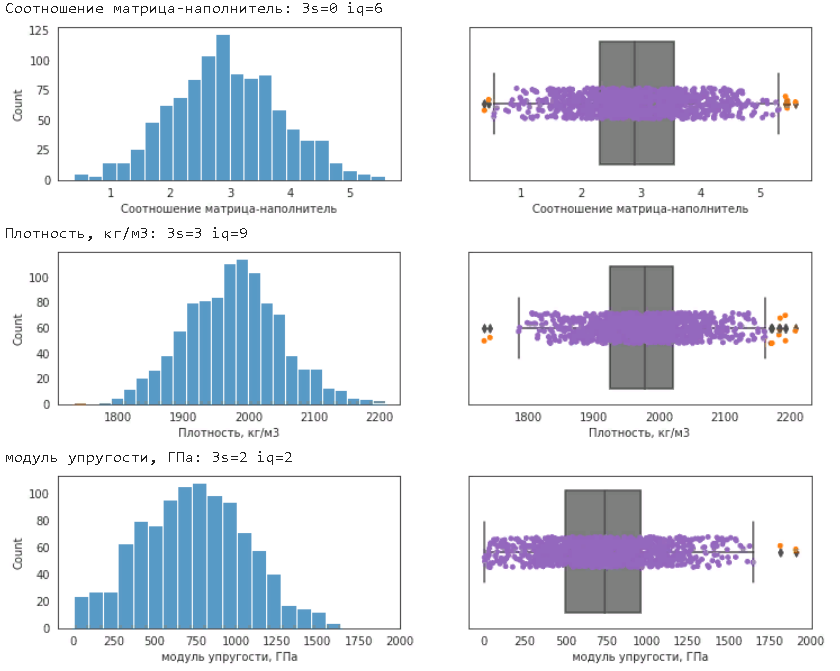


Рисунок 2.2 Визуализация выбросов

Все выбросы были помечены как NaN («не число») и удалены с помощью функции библиотеки pandas dropna().

**Нормализация**

Также при выполнении разведочного анализа данных было замечено, что значения данных изменяются в очень больших диапазонах и также у разных параметров отличаются на порядки. Это может приводить к некорректной работе моделей машинного обучения – большой дисбаланс между значениями признаков может ухудшать результаты обучения и замедлять сам процесс моделирования. Поэтому данные были нормализованы с использованием метода MinMaxScaler из библиотеки Sklearn. Т.к. в нашем наборе данных нет отрицательных значений, то этот метод отмасштабировал все данные от 0 до 1.

В таблице 2.1 приведена описательная статистика после нормализации.

Таблица 2.1 – Описательная статистика после нормализации

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Параметр** | **count** | **mean** | **std** | **min** | **0.25** | **0.50** | **0.75** | **max** |
| Соотношение матрица-наполнитель | 936 | 0.50 | 0.19 | 0 | 0.37 | 0.49 | 0.63 | 1 |
| Плотность, кг/м3 | 936 | 0.50 | 0.19 | 0 | 0.37 | 0.51 | 0.63 | 1 |
| модуль упругости, ГПа | 936 | 0.45 | 0.20 | 0 | 0.30 | 0.45 | 0.58 | 1 |
| Количество отвердителя, м.% | 936 | 0.50 | 0.19 | 0 | 0.38 | 0.51 | 0.64 | 1 |
| Содержание эпоксидных групп,%\_2 | 936 | 0.49 | 0.18 | 0 | 0.37 | 0.49 | 0.62 | 1 |
| Температура вспышки, С\_2 | 936 | 0.52 | 0.19 | 0 | 0.39 | 0.52 | 0.65 | 1 |
| Поверхностная плотность, г/м2 | 936 | 0.37 | 0.22 | 0 | 0.21 | 0.35 | 0.54 | 1 |
| Модуль упругости при растяжении, ГПа | 936 | 0.49 | 0.19 | 0 | 0.36 | 0.49 | 0.62 | 1 |
| Прочность при растяжении, МПа | 936 | 0.50 | 0.19 | 0 | 0.37 | 0.49 | 0.61 | 1 |
| Потребление смолы, г/м2 | 936 | 0.52 | 0.20 | 0 | 0.39 | 0.52 | 0.65 | 1 |
| Угол нашивки, град | 936 | 0.51 | 0.50 | 0 | 0.00 | 1.00 | 1.00 | 1 |
| Шаг нашивки | 936 | 0.50 | 0.18 | 0 | 0.37 | 0.50 | 0.62 | 1 |
| Плотность нашивки | 936 | 0.51 | 0.19 | 0 | 0.39 | 0.52 | 0.64 | 1 |

Предобработку данных закончили. Удалили выбросы и нормализовали значения данных.

* 1. **Разработка и обучение модели**

В соответствии с поставленной задачей, нужно осуществить разработку и обучение моделей машинного обучения для двух выходных параметров: «Прочность при растяжении» и «Модуль упругости при растяжении». Для каждого признака построение моделей осуществляется раздельно. Разделение нормализованных данных на обучающую и тестовую выборки (в соотношении 70 на 30%, согласно поставленной задаче).

Для признака «Модуль упругости при растяжении» были разработаны и обучены следующие модели:

− модель на основе линейной регрессии (метод LinearRegression);

− модель k ближайших соседей (метод KNeighborsRegressor());

− модель Стохастический градиентный спуск (метод Gradient Boosting Regressor ());

− модель случайный лес (метод RandomForestRegressor());

- Модель Многослойный перцептрон (Multilayered perceptron (MLP))

Общий алгоритм работы с моделями следующий:

1. Вызываем объект метода регрессии

2. Задаем возможные параметры оценки. Все модели будем прогонять по сетке GridSearchCV для получения наилучших параметров оценки целевых переменных.

3. Вызываем метод GridSearchCV, передавая в него сам объект метода регрессии и возможные параметры. Получаем лучшее возможное решение.

4. Считаем ошибки модели, записываем их в датасет ошибок.

5. Визуализируем результат работы модели.

* 1. **Тестирование модели**

Ниже представлены результаты работы моделей для двух целевых переменных в виде соотношения тест/прогноз.

**Линейная регрессия**

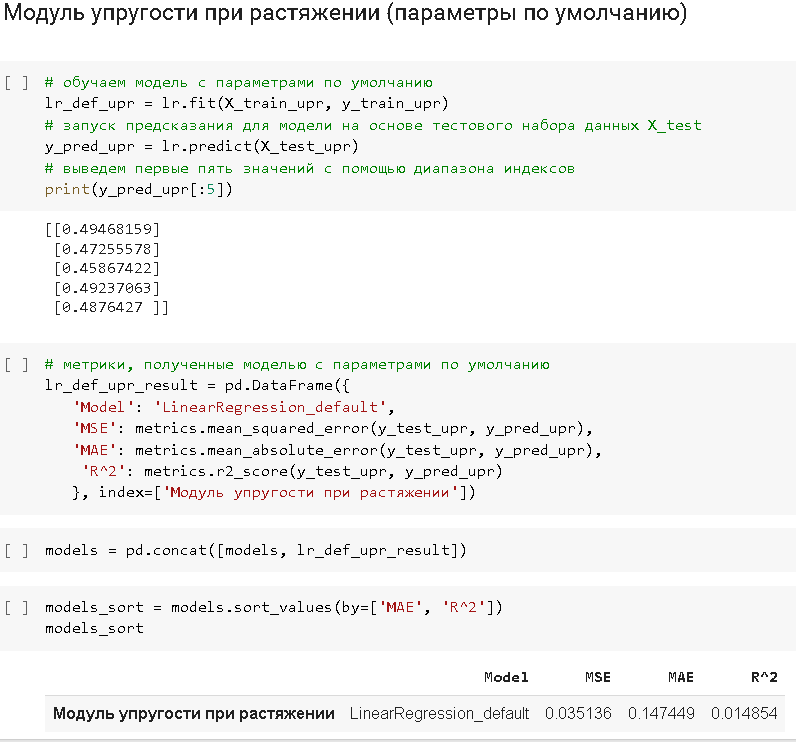


Рисунок 2.3 Обучение модели. Упругость при растяжении

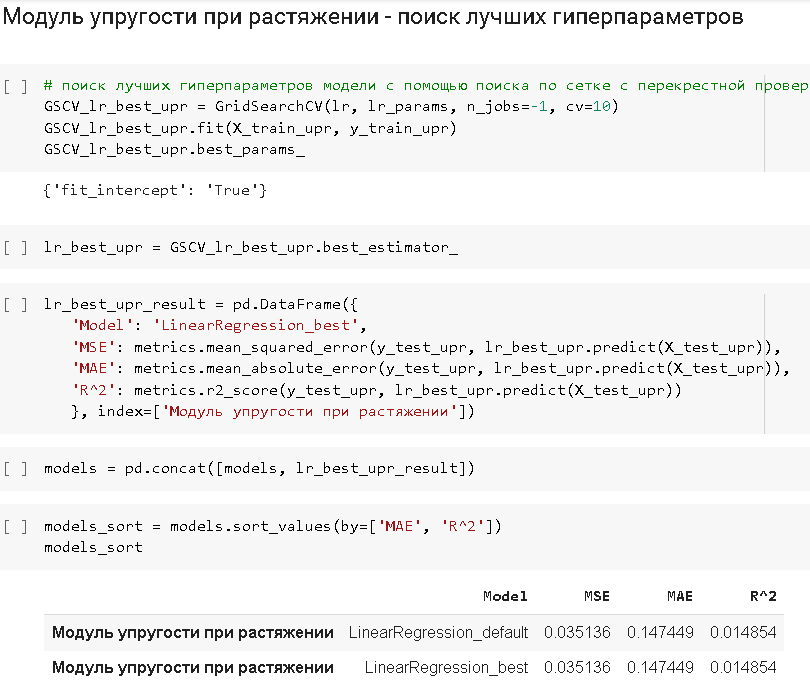
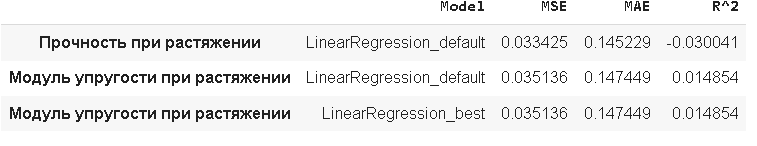


Рисунок 2.4 Поиск лучших гиперпараметров. Упругость при растяжении

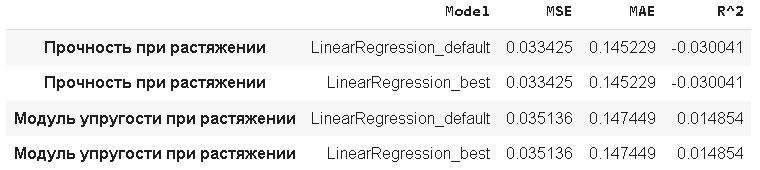


Рисунок 2.5 Визуализация. Упругость при растяжении

### **Прочность при растяжении - параметры по умолчанию**



### **Прочность при растяжении - поиск лучших гиперпараметров**



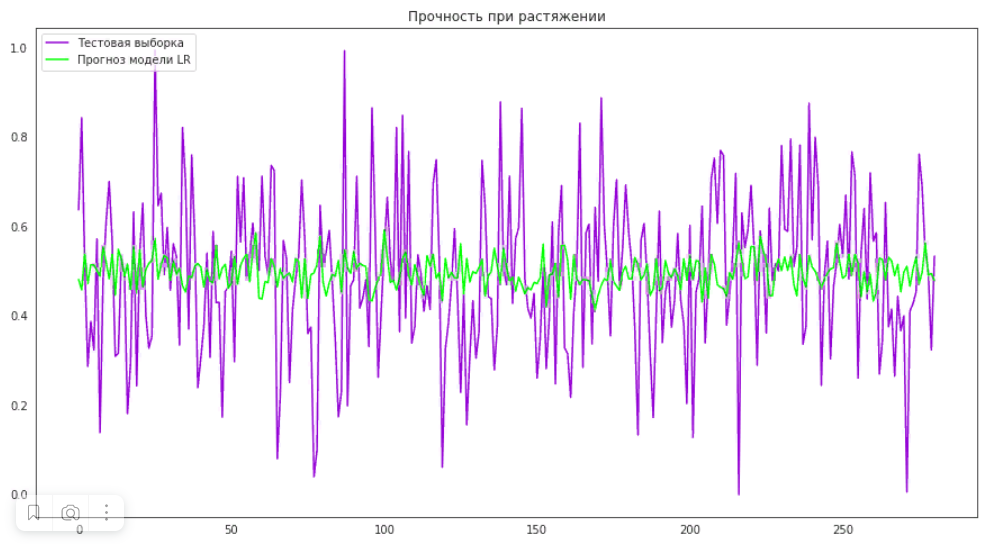
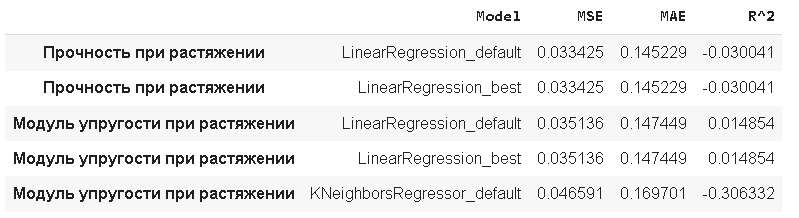


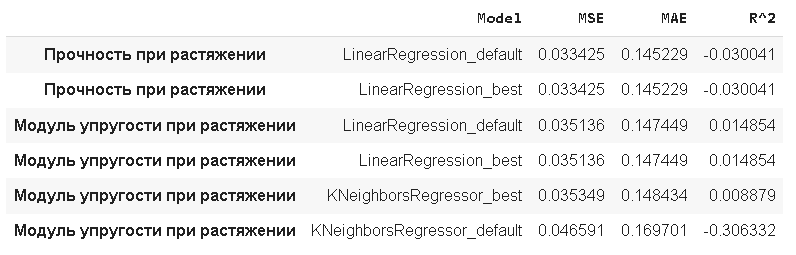
Рисунок 2.6 Визуализация. Прочность при растяжении

## **K-Neighbors Regressor**

### **Модуль упругости при растяжении - параметры по умолчанию**



### **Модуль упругости при растяжении - поиск лучших гиперпараметров**



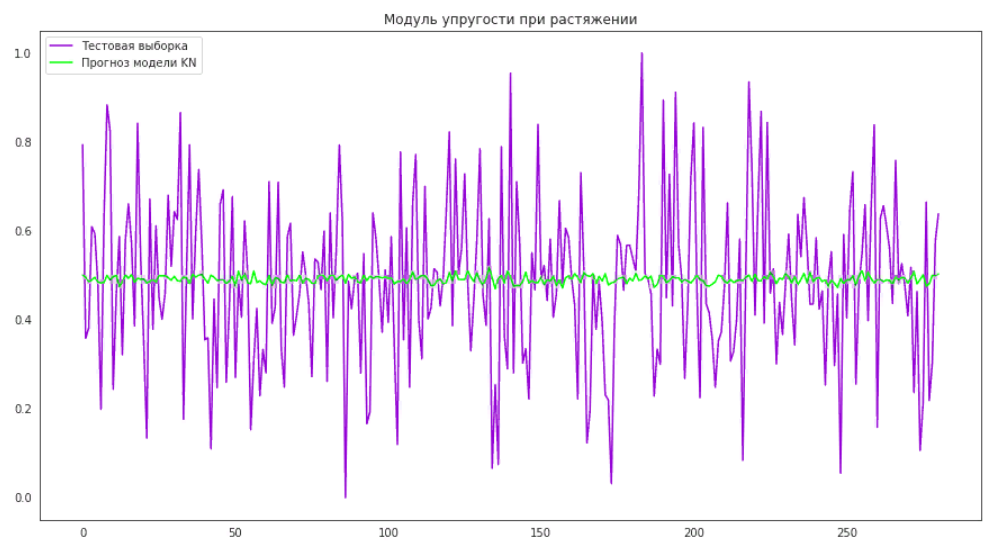


Рисунок 2.7 Визуализация. Упругость при растяжении

### **Прочность при растяжении - параметры по умолчанию**

### **Прочность при растяжении - поиск лучших гиперпараметров**



## 

Рисунок 2.8 Визуализация. Прочность при растяжении

## **Gradient Boosting Regressor**

### **Модуль упругости при растяжении - параметры по умолчанию**

### **Модуль упругости при растяжении - поиск лучших гиперпараметров**

### 

Рисунок 2.9 Визуализация. Упругость при растяжении

### **Прочность при растяжении - параметры по умолчанию**

### **Прочность при растяжении - поиск лучших гиперпараметров**

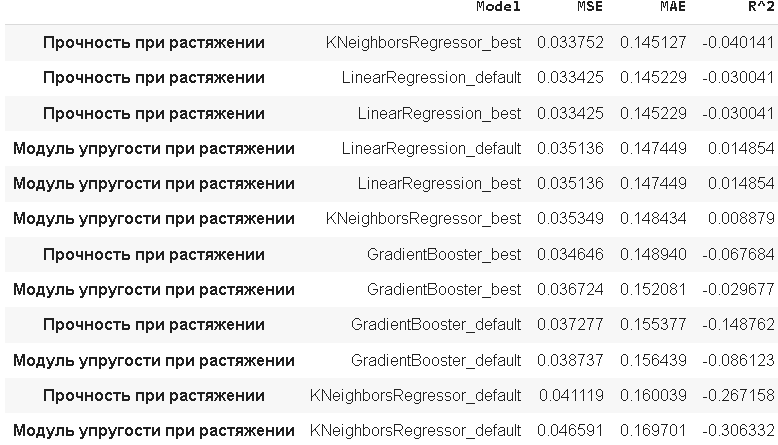
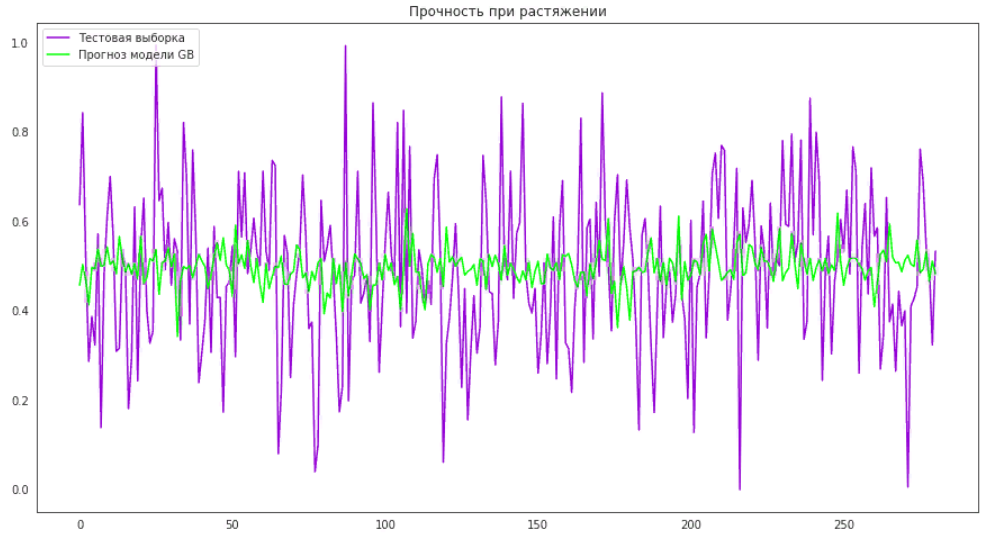
 

Рисунок 2.10 Визуализация. Прочность при растяжении

## **Random Forest Regressor**

## 

Рисунок 2.11 Визуализация. Прочность при растяжении

## 

Рисунок 2.11 Визуализация. Прочность при растяжении

## **Multilayered perceptron (MLP)**

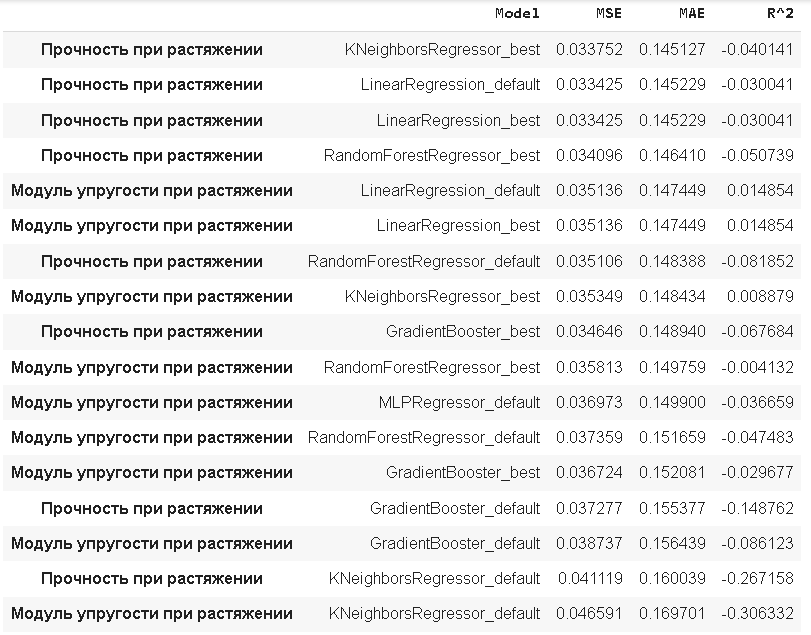
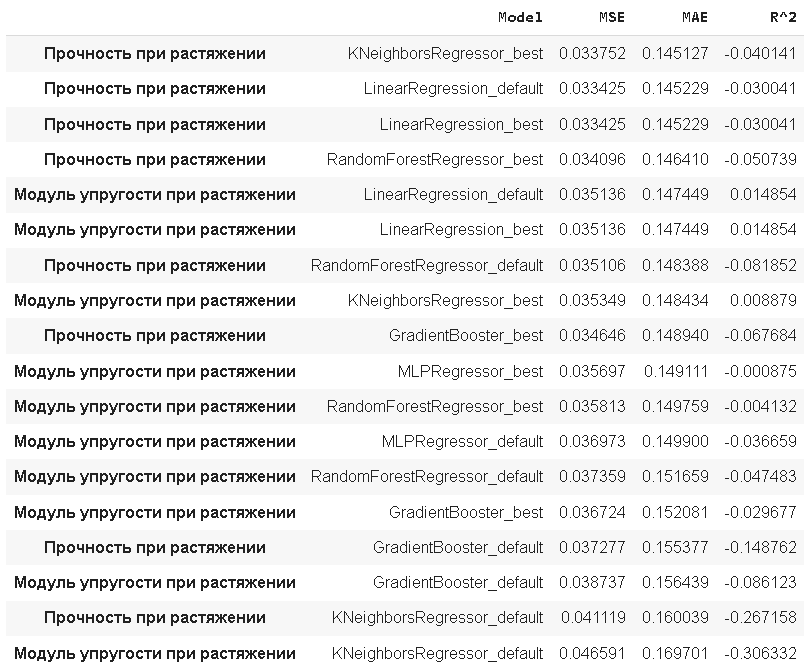
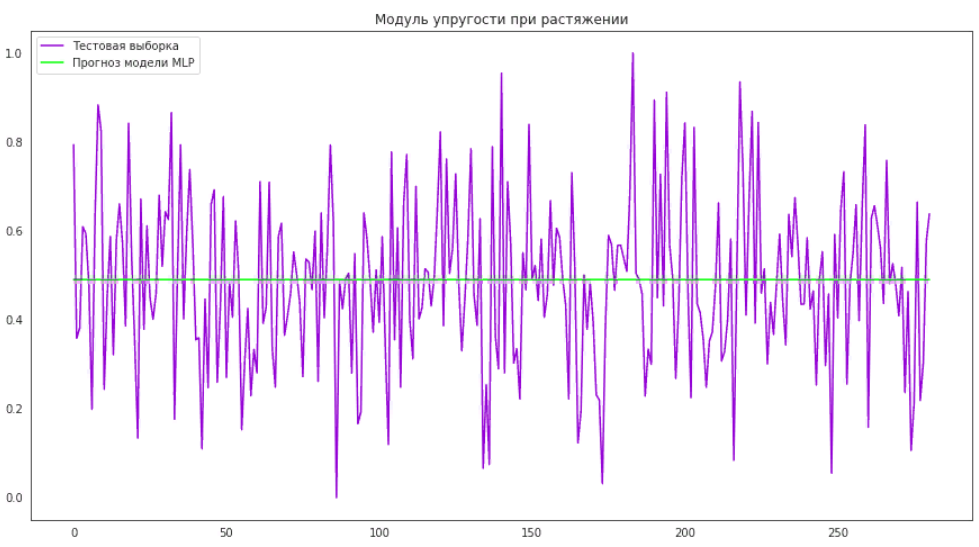
  

Рисунок 2.12 Визуализация. Упругость при растяжении

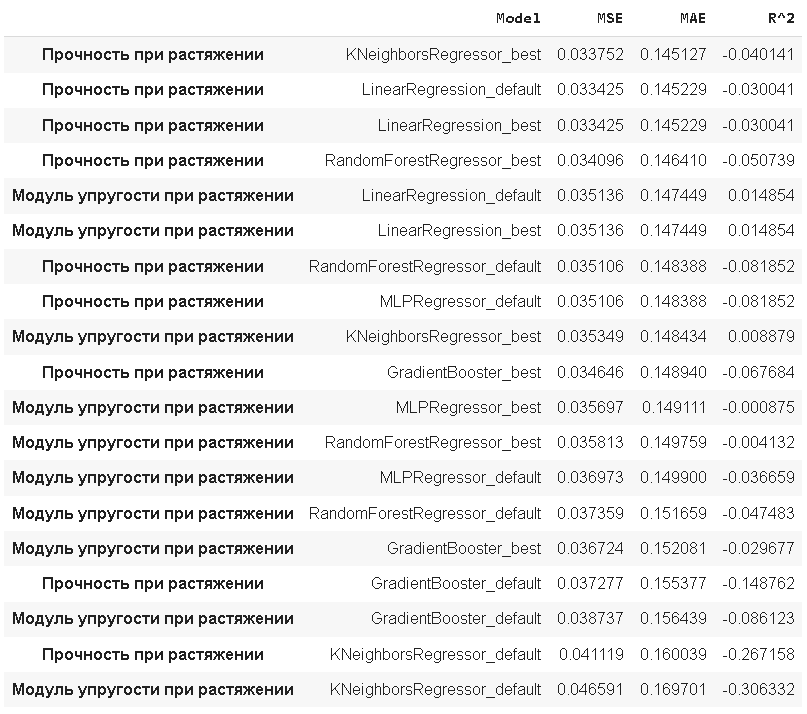
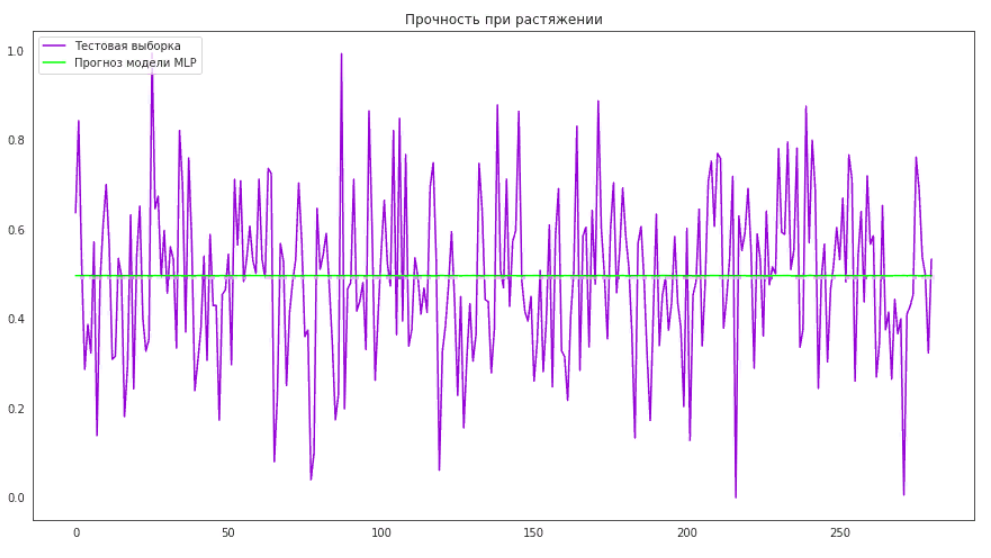
  

Рисунок 2.13 Визуализация. Прочность при растяжении

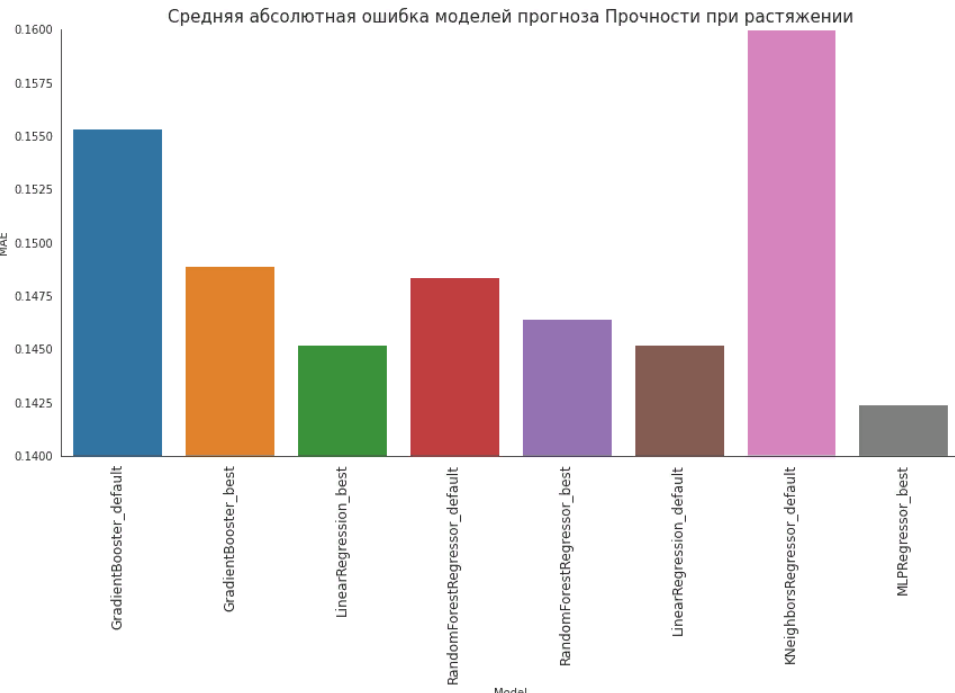


Рисунок 2.15 Визуализация. Прочность при растяжении

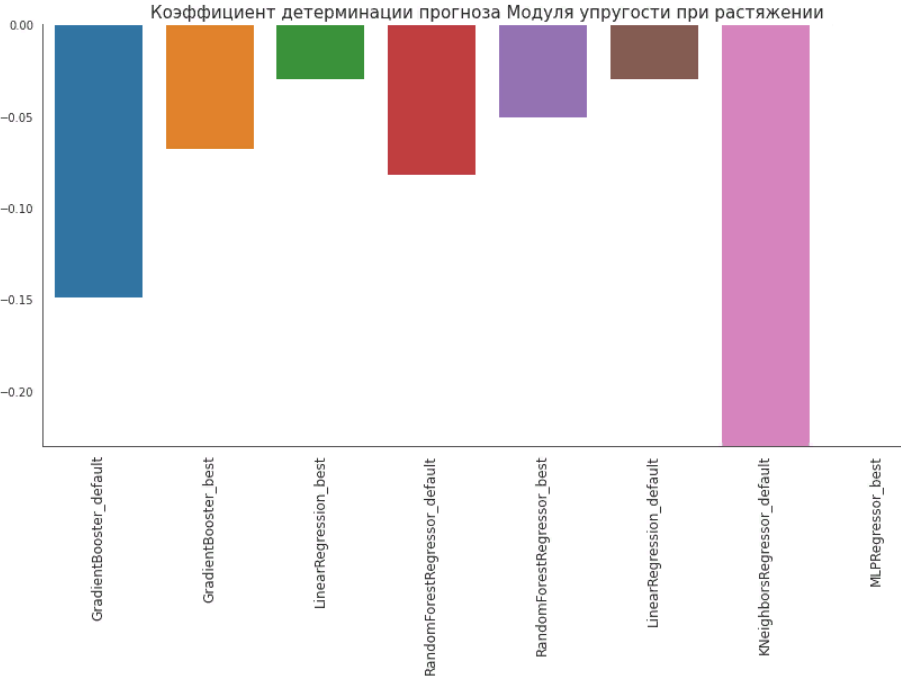


Рисунок 2.16 Визуализация. Прочность при растяжении

У всех моделей коэффициент детерминации имеет отрицательные значения. Т.о., модели не дают прогнозов, которые были бы лучше простого расчета среднего значения. Также зафиксированы большие значения ошибок (MAE и MSE), что также свидетельствуют о низком качестве моделей.

Таблица 2.2 – Метрики качества обучения выбранных моделей

|  |  |  |  |
| --- | --- | --- | --- |
| Модуль упругости при растяжении |  |  |  |
| **Model** | **MSE** | **MAE** | **R^2** |
| RandomForestRegressor\_best | 0.03472 | 0.15124 | -0.00263 |
| KNeighborsRegressor\_best | 0.03472 | 0.15123 | -0.00280 |
| LinearRegression\_default | 0.03481 | 0.15221 | -0.00522 |
| LinearRegression\_best | 0.03481 | 0.15221 | -0.00522 |
| **DummyRegression\_mean** | **0.03483** | **0.15120** | **-0.00593** |
| MLPRegressor\_best | 0.03483 | 0.15121 | -0.00599 |
| GradientBooster\_best | 0.03510 | 0.15179 | -0.01367 |
| RandomForestRegressor\_default | 0.03558 | 0.15344 | -0.02753 |
| MLPRegressor\_default | 0.03643 | 0.15627 | -0.05194 |
| GradientBooster\_default | 0.03780 | 0.15633 | -0.09154 |
| KNeighborsRegressor\_default | 0.04169 | 0.16648 | -0.20410 |
| Прочность при растяжении |  |  |  |
| **Model** | **MSE** | **MAE** | **R^2** |
| KNeighborsRegressor\_best | 0.03481 | 0.14658 | -0.02799 |
| **DummyRegression\_mean** | **0.03486** | **0.14675** | **-0.02967** |
| RandomForestRegressor\_best | 0.03495 | 0.14725 | -0.03239 |
| LinearRegression\_default | 0.03545 | 0.14790 | -0.04700 |
| LinearRegression\_best | 0.03545 | 0.14790 | -0.04700 |
| MLPRegressor\_best | 0.03562 | 0.14855 | -0.05194 |
| GradientBooster\_best | 0.03567 | 0.15002 | -0.05363 |
| RandomForestRegressor\_default | 0.03599 | 0.15050 | -0.06305 |
| GradientBooster\_default | 0.03789 | 0.15570 | -0.11903 |
| MLPRegressor\_default | 0.03789 | 0.15570 | -0.11903 |
| KNeighborsRegressor\_default | 0.04143 | 0.15725 | -0.22358 |

Настройка гиперпараметров с помощью GridSearchCV и RandomizedSearchCV не повлияло на метрики линейной регрессии, что было ожидаемо, и улучшило результат работы остальных моделей (случайный лес, k-ближайших соседей, градиентный бустинг и многослойный персептрон).

Наилучшие показатели после настройки гиперпараметров выдали следующие модели:

Случайный лес для Модуля упругости при растяжении с параметрами

К-ближайщих соседей для Прочности при растяжении с параметрами:

* 1. **Разработка нейронной сети**

Для прогнозирования Соотношения матрица-наполнитель с помощью нейронной сети использовалась библиотека Keras фреймворка TensorFlow. Были смоделированы три варианта

* сеть со стандартными параметрами;
* сесть с ранней остановкой обучения;
* сеть с Dropout.

Количество скрытых слоев для первых двух вариантов было установлено равным варианту, полученному после настройки гиперпараметров многослойный персептрона.

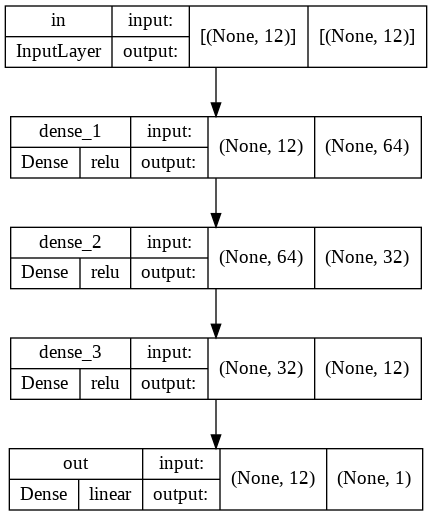


Рисунок 2.17 – Архитектура сети (стандарт и ранняя остановка)

График обучения нейросети со стандартными параметрами приведен ниже.

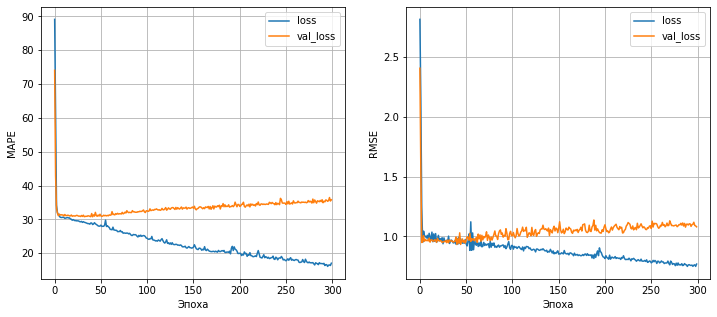


Рисунок 2.18 – Архитектура сети (стандарт и ранняя остановка)

Из него видно, что, начиная примерно с 10-й эпохи сеть начала переобучаться. Значение loss на тестовых выборках продолжило уменьшаться, а на валидационной стало расти.

Одним из способов борьбы с переобучением может быть ранняя остановка обучения. Для этого в Tensorflow используются обратные вызовы (callbacks). Для сети с той же архитектурой было запущено обучение с ранней остановкой. График обучения приведен на рисунке ниже.

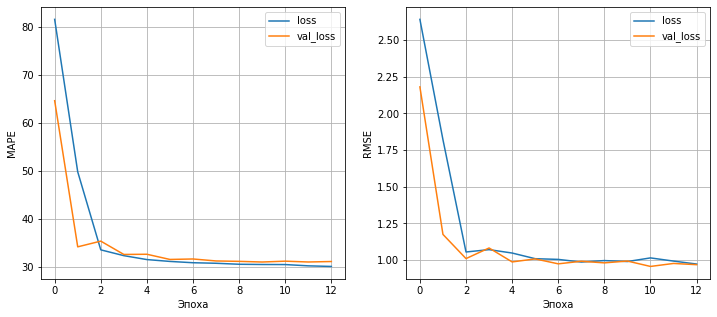


Рисунок 2.19 – Архитектура сети (стандарт и ранняя остановка)

Из этого графика можно сделать вывод, что ранняя остановка решает проблему переобучения и повышает точность модели на новых данных.

Еще одним методом борьбы с переобучением является добавление Dropout-слоев. Построим модель аналогичной архитектуры, только после каждого скрытого слоя добавим слой Dropout с параметром 0.05. Такой слои слои выключат 5% случайных нейронов на каждом слое.

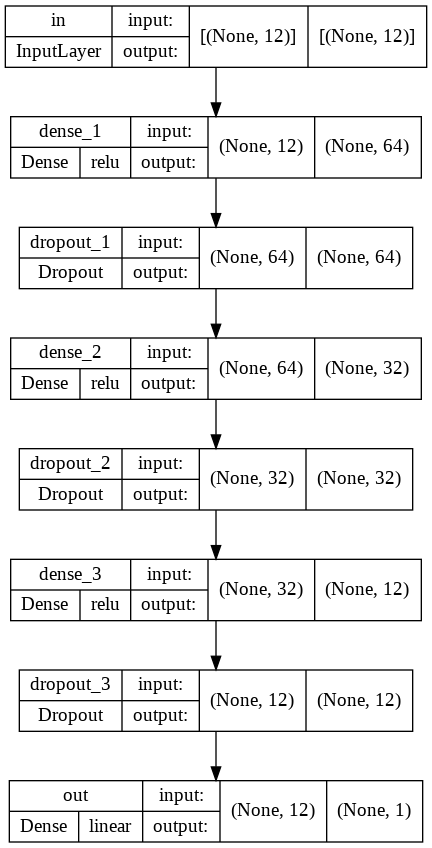


Рисунок 2.20– Архитектура сети (стандарт и ранняя остановка)

График обучения приведен на рисунке ниже.

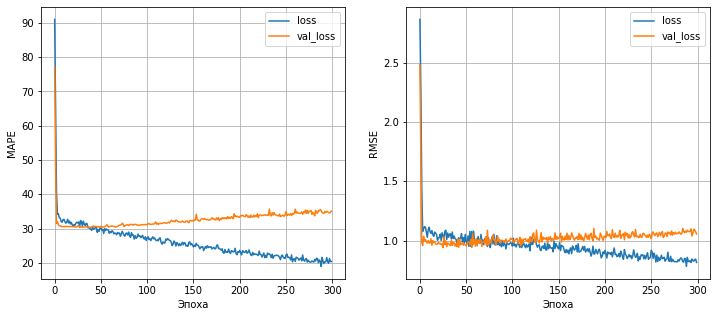


Рисунок 2.21– Архитектура сети (стандарт и ранняя остановка)

Видно, что Dropout-слои справились с переобучением лучше, чем сеть со без них (первый вариант). Но при увеличении количества проходов значение loss на валидационных выборках снова ой стало расти.

Метрики, полученные при всех трех вариантах

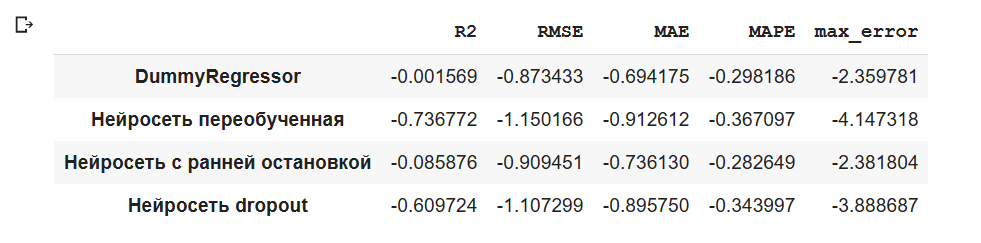
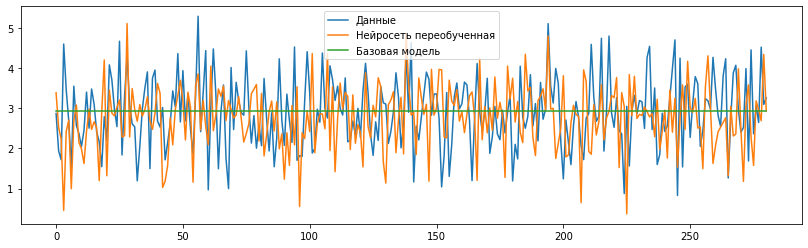
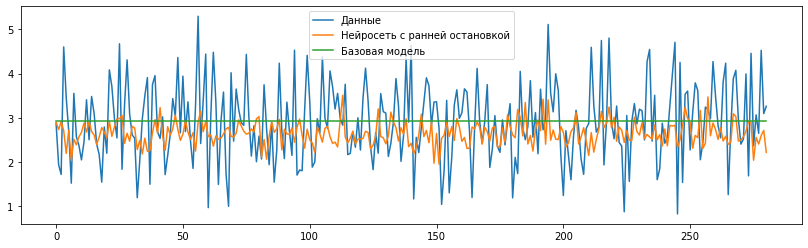


Рисунок 2.22– Сравнение метрик смоделированных нейронных сетей

Использование ранней остановки сокращает время на обучение модели, а использование Dropout увеличивает. Но уменьшается риск, что мы остановились слишком рано.





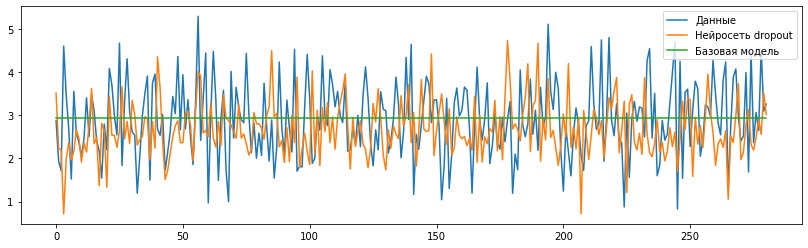


Рисунок 2.23 – Сравнение прогнозов трех вариантов нейронных сетей

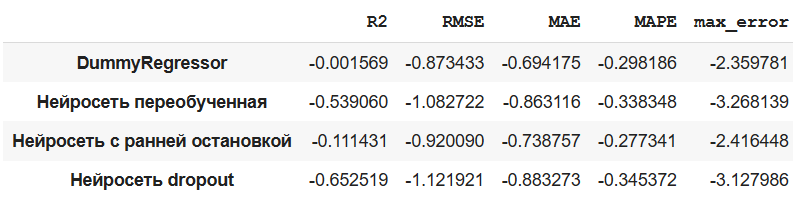


Рисунок 2.24 – Метрики нейронных сетей

Из сводной таблицы метрик и графиков на рисунке видно, что лучшая обобщающая способность и меньшие значения ошибок на тестовом множестве оказались у нейронной сети, обученной с ранней остановкой. Но и она предсказывает гораздо хуже базовой модели (среднего значения). Ошибка по любой из нейронных сетей больше, чем у рассмотренных выше моделей библиотеки Scikit-learn. Лучший прогноз по-прежнему дает метод К-ближайших соседей. По графику делаем вывод что для обучения спроектированной нейронной сети достаточно 30 эпох. Ошибки модели следующие: MSE=1.251727, R^2 = -0.488577. Результаты неудовлетворительны.

Согласно заданию, необходимо было сравнить ошибку модели на тренировочной и тестирующей части выборки. Это было сделано для лучшей модели К-ближайших соседей и для нейронной сети с ранней остановкой.

Таблица 2.3 - Сравнение ошибок модели для модуля упругости при растяжении на тренировочном и тестовом датасете

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **R2** | **RMSE** | **MAE** | **max\_error** |
| **Случайный лес** |  |  |  |  |
| Модуль упругости при растяжении, тренировочный | 0.01238 | -0.19223 | -0.15483 | -0.51702 |
| Модуль упругости при растяжении, тестовый | -0.00263 | -0.18633 | -0.15124 | -0.49743 |
| **К-ближайших соседей** |  |  |  |  |
| Прочность при растяжении, тренировочный | 0.00650 | -0.18943 | -0.15111 | -0.50857 |
| Прочность при растяжении, тестовый | -0.02799 | -0.18656 | -0.14658 | -0.49695 |
| **Сеть с ранней остановкой** |  |  |  |  |
| Соотношение матрица-наполнитель, тренировочный | -0.18256 | -0.98054 | -0.79410 | -3.03017 |
| Соотношение матрица-наполнитель, тестовый | -0.11143 | -0.92009 | -0.73876 | -2.41645 |

Случайный лес и Метод К-ближайших соседей показали положительный коэффициент детерминации на тренировочной выборке. Возможно модели чему-то все-таки научилось. Но даже на тренировочном датасете они не нашли закономерностей во входных данных.

У нейронной сети показатели для тестовой выборки сильнее отличаются в от показателей тренировочной. Т.е. с переобучением она справилась. Но требуется более тщательное и грамотное построение архитектуры нейронной сети, чтобы получить лучший результат. В целом все модели работают не точнее среднего, и бесполезны для применения в реальных условиях. Задачу прогнозирования параметров решить не удалось.

* 1. **Разработка приложения**

Несмотря на то, что пригодных к внедрению моделей получить не удалось, можно разработать тестовое веб-приложение на фреймворке Flask. Сохраняем модель keras в качестве логической составляющей приложения.

В приложении необходимо реализовать следующие функции:

* загрузка сохраненной модели;
* ввод входных параметров;
* получение и отображение прогноза выходных параметров.



Рисунок 2.25 – Написание приложения

Ввод входных параметров пользователем осуществляется через формы на веб странице.

**3.Использование репозитория**

Для данного исследования был создан удаленный репозиторий на GitHub, который находится по адресу:

[https://github.com/[Yuliya31](https://github.com/Yuliya31)/[VKR](https://github.com/Yuliya31/VKR)](https://github.com/leksaw/datascience_bmstu/)

В него загружены результаты работы: код анализа данных, обучения моделей, код веб-приложения, пояснительная записка и презентация.

**Заключение**

Использованные при разработке моделей подходы не позволили получить сколь-нибудь достоверных прогнозов. Данный факт не указывает на то, что прогнозирование характеристик композитных материалов на основании предоставленного набора данных невозможно, но может указывать на отдельные недостатки в данных, недостаточном числе подходов, использованных при прогнозе, а также на необходимости пересмотра инструментов для прогнозирования.

Возможные причины неудовлетворительной работы моделей:

* нечеткая постановка задачи, отсутствие дополнительной информации о зависимости признаков с точки зрения физики процесса;
* исследование предварительно обработанных данных, возможно, на не обработанных дополнительно вручную экспериментальных данных можно было бы получить более качественные модели;
* недостаток знаний и опыта, в результате чего были испробованы не все возможные способы анализа данных методы машинного обучения.

Тем не менее на основании проведенного исследования можно сделать следующие основные выводы по теме:

* распределение исходных данных близко к нормальному;
* корреляция между парами признаков практически отсутствует;
* примененные модели регрессии и нейронных сеть не показали высокой эффективности в прогнозировании свойств композитов в этом конкретном исследовании, необходимы дополнительные вводные данные для улучшения моделей;
* лучшие метрики на основании исключительно количественного сравнения показали модели, созданные с использованием алгоритма К-ближайших соседей.

В целом описанные выше подходы хорошо показали себя как обучающие.

**4.Библиографический список**

1. Гласснер Э. Глубокое обучение без математики. Т. 1: Основы. Пер. с анг. - М.: ДМК Пресс, 2019. – 584 с.
2. Грас Д. Data Science. Наука о данных с нуля: Пер. с англ. - СПб.: БХВ-Петербурr, 2021. - 416 с.
3. Рашка С., Мирджалили В. Python и машинное обучение: машинное и глубокое обучение с использованием Python, scikit-learn и TensorFlow 2: Пер. с англ. - СПб. : Диалектика, 2020. - 848 с.
4. Орельен Ж. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем: Пер. с англ. - СПБ.: Альфа-книга, 2018. - 688 с.
5. Библиотека scikit-learn [Электронный ресурс]: - Режим доступа: https://scikit-learn.org (дата обращения: 20.10.2022).
6. Библиотека Keras [Электронный ресурс]: - Режим доступа: https://keras.io/ (дата обращения: 20.10.2022).
7. Библиотека Tensorflow [Электронный ресурс]: - Режим доступа: https://www.tensorflow.org/overview (дата обращения: 20.10.2022).
8. Документация Pandas [Электронный ресурс]: - Режим доступа: https://pandas.pydata.org/docs/user\_guide/index.html#user-guide (дата обращения: 20.10.2022).
9. Документация Seaborn [Электронный ресурс]: - Режим доступа: https://seaborn.pydata.org/ (дата обращения: 20.10.2022).
10. Документация Flask [Электронный ресурс]: - Режим доступа: https://pypi.org/project/Flask/ (дата обращения: 20.10.2022).