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 Кроткевич Анна Владимировна

Москва, 2023

Содержание

Введение

Тема данной работы: Прогнозирование конечных свойств новых материалов (композиционных материалов).

Актуальность темы: созданные прогнозные модели помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов возможными новыми характеристиками материалов, и цифровыми двойниками новых композитов. Кейс основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

Композиционный матери́ал (КМ), сокращённо компози́т — многокомпонентный материал, изготовленный (человеком или природой) из двух или более компонентов с существенно различными физическими и/или химическими свойствами, которые, в сочетании, приводят к появлению нового материала с характеристиками, отличными от характеристик отдельных компонентов и не являющимися простой их суперпозицией. В составе композита принято выделять матрицу/матрицы и наполнитель/наполнители, последние выполняют функцию армирования.

Сочетание разных компонентов позволяет улучшить характеристики материала и делает его одновременно лёгким и прочным. При этом отдельные компоненты остаются таковыми в структуре композитов, что отличает их от смесей и затвердевших растворов.

Варьируя состав матрицы и наполнителя, их соотношение, ориентацию наполнителя, получают широкий спектр материалов с требуемым набором свойств. Многие композиты превосходят традиционные материалы и сплавы по своим механическим свойствам и в то же время они легче.

Использование композитов обычно позволяет уменьшить массу конструкции при сохранении или улучшении её механических характеристик. Этим обусловлено широкое применение композиционных материалов в различных областях техники.

1. Аналитическая часть

1.1. Постановка задачи

На входе имеются данные о начальных свойствах компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.). Необходимо спрогнозировать ряд конечных свойств получаемых композиционных материалов, а именно:

- модуль упругости при растяжении,
- прочность при растяжении,
- соотношение матрица-наполнитель.

То есть требуется разработать модели, прогнозирующие значения данных свойств в зависимости от остальных.

Также требуется разработать приложение, делающее удобным использование данных моделей.

Представлено два датасета в файлах X_bp.xlsx и X_nup.xlsx .

Файл X_bp содержит: признаков: 10 и индекс; строк: 1023.

Файл X_nup содержит: признаков: 3 и индекс; строк: 1040.

Известно, что файлы требуют объединения с типом INNER по индексу. После объединения часть строк из файла X_nup была отброшена. Дальнейшие исследования проводится с объединенным датасетом, содержащим 13 признаков и 1023 строк или объектов.

Описание признаков объединенного датасета приведено на рисунке 1. Все признаки имеют тип float64, то есть вещественный. Пропусков в данных нет. Все признаки, кроме «Угол нашивки», являются непрерывными, количественными. «Угол нашивки» принимает только два значения. Выбросы в данных имеются, об этом сказано подробнее в подразделе 1.3 Разведочный анализ данных.

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

Рисунок 1 — Описание признаков датасета

1.2. Описание используемых методов

Поставленная задача является задачей регрессии, так как необходимо выявить закономерности (зависимости) между наблюдаемыми переменными (их также называют регрессорами, признаками) и целевыми (модуль упругости при растяжении, Гпа, прочность при растяжении, МПа; соотношение матрица-наполнитель.) и сделать предсказание целевых переменных.

Зависимости между наблюдаемыми и целевой переменными могут быть любыми, в том числе сколь угодно сложными. Поэтому будем подбирать модель машинного обучения, которая по значениям наблюдаемых переменных будет выдавать значение целевой переменной, “близкое” к истинному, из ограниченного конкретного набора моделей.

Модель в машинном обучении – это математическая функция, которая (в контексте задачи регрессии) получает на вход значения наблюдаемых переменных, преобразует их, комбинирует и выдаёт значение целевой переменной.

В настоящее время разработано много методов регрессионного анализа.

В данной работе применялись следующие модели: LinearRegression, Lasso, Ridge, SVR, KNeighborsRegressor, DecisionTreeRegressor RandomForestRegressor, GradientBoostingRegressor и нейронная сеть.

Линейная регрессия

Одним из наиболее простых классов (наборов) моделей является класс линейных моделей.

Линейная регрессия — это метод анализа данных, который математически моделирует неизвестную или зависимую переменную и известную или независимую переменную в виде линейного уравнения (1) и строится соответствующая прямая, известная как линия регрессии.

$$y = ax + b \quad (1)$$

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

Если ищется зависимость между несколькими входными и одной выходной переменными, то имеет место множественная линейная регрессия. Соответствующее уравнение имеет вид (2).

$$Y = b0 + b1 * x1 + b2 * x2 + \cdots + bn * xn, \quad (2)$$

где n - число входных переменных.

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

Линейная регрессия — 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. Тем не менее, именно с линейной регрессии целесообразно начать подбор подходящей модели.

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

Лассо (LASSO) и гребневая (Ridge) регрессия

Метод регрессии лассо (LASSO, Least Absolute Shrinkage and Selection Operator) — это вариация линейной регрессии, специально адаптированная для данных, которые имеют сильную сильную корреляцию признаков друг с другом.

LASSO использует сжатие коэффициентов (shrinkage) и этим пытается уменьшить сложность данных, искривляя пространство, на котором они лежат. В этом процессе лассо автоматически помогает устранить или исказить сильно коррелированные и избыточные функции в методе с низкой дисперсией.

Регрессия лассо использует регуляризацию L1, то есть взвешивает ошибки по их абсолютному значению.

Гребневая регрессия или ридж-регрессия — так же вариация линейной регрессии, очень похожая на регрессию LASSO. Она так же применяет сжатие и хорошо работает для данных, которые демонстрируют сильную мультиколлинеарность.

Самое большое различие между ними в том, что гребневая регрессия использует регуляризацию L2, которая взвешивает ошибки по их квадрату, чтобы сильнее наказывать за более значительные ошибки.

Регуляризация позволяет интерпретировать модели. Если коэффициент стал 0 (для Lasso) или близким к 0 (для Ridge), значит данный входной признак не является значимым.

Эти методы реализованы в sklearn.linear_model.Lasso и sklearn.linear_model.Ridge.

Метод опорных векторов для регрессии

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

Чаще всего он применяется в постановке бинарной классификации.

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

Если в исходном пространстве объекты линейно неразделимы, то 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.

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

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

Метод k-ближайших соседей

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

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

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

Этот метод — пример непараметрической регрессии.

Он реализован в sklearn.neighbors.KneighborsRegressor.

Деревья решений

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

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

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

В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу. В результате проверки множество примеров, попавших в узел, разбивается на два подмножества: удовлетворяющие правилу и не удовлетворяющие ему. Затем к каждому подмножеству вновь применяется

правило и процедура рекурсивно повторяется пока не будет достигнуто некоторое условие остановки алгоритма. В последнем узле проверка и разбиение не производится, и он объявляется листом.

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

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут, по которому это будет сделано. Общее правило для классификации можно сформулировать так: выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, а количество объектов из других классов в каждом из этих множеств было как можно меньше. Для этого были выбраны различные критерии, например, теоретико-информационный и статистический.

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

Огромное преимущество деревьев решений в том, что они легко интерпретируемы, понятны человеку. Они могут использоваться для извлечения правил на естественном языке. Еще преимущества — высокая точность работы, 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. В худшем случае, дерево будет большой глубины и сложной структуры, а в каждом листе будет только один объект. Для решения этой проблемы используют разные критерии остановки алгоритма.

Деревья решений реализованы в sklearn.tree.DecisionTreeRegressor.

Случайный лес

Случайный лес (RandomForest) — представитель ансамблевых методов.

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

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

где

N – количество деревьев;

i – счетчик для деревьев;

b – решающее дерево;

x – сгенерированная нами на основе данных выборка.

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

Преимущества случайного леса:

– высокая точность предсказания;

– редко переобучается;

– практически не чувствителен к выбросам в данных;

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

– высокая параллелизуемость и масштабируемость.

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

Метод реализован в sklearn.ensemble.RandomForestRegressor.

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.

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

Нейронная сеть

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

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

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

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

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

– входной слой — его размер соответствует входным параметрам;

– скрытые слои — их количество и размерность определяем специалист;

– выходной слой — его размер соответствует выходным параметрам.

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

Разведочный анализ данных — анализ основных свойств данных, нахождение в них общих закономерностей, распределений и аномалий, построение начальных моделей, зачастую с использованием инструментов визуализации.

В рамках данной работы выполнено следующее: описательная статистика датасета (число непропущенных значений, среднее, стандартное отклонение, диапазон, медиану, 0.25 и 0.75 квартили), см. рисунок 2, гистограммы распределения и диаграммы ящика с усами каждой переменной, см. рисунки 3-4, попарные графики рассеяния точек, см. рисунок 5, матрица корреляции признаков, см. рисунок 6, и коэффициенты корреляции каждой целевой переменной ('Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа', 'Соотношение матрица-наполнитель') в виде отдельного перечня по возрастанию для большей наглядности.

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

Рисунок 2 - описательная статистика датасета

Гистограммы распределения переменных и диаграммы «ящик с усами» приведены на рисунках 3-4. По ним видно, что практически все признаки, кроме «Угол нашивки», имеют нормальное распределение и принимают неотрицательные значения. «Угол нашивки» принимает значения: 0, 90.

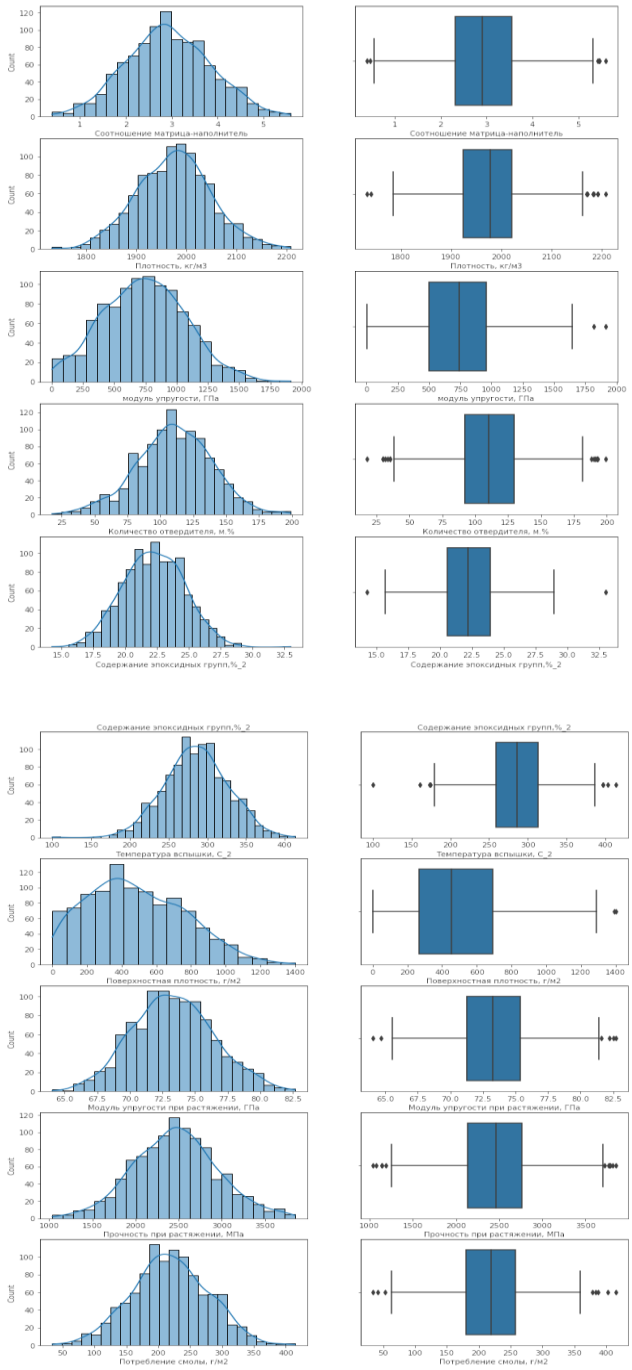

Рисунок 3 - Гистограммы распределения переменных и диаграммы «ящик с усами»

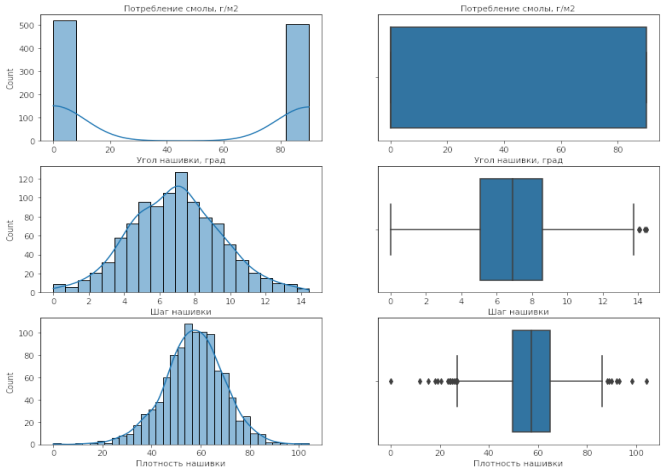

Рисунок 4 - Гистограммы распределения переменных и диаграммы «ящик с усами»

Для корректной работы большинства моделей желательна сильная зависимость выходных (целевых) переменных от входных и отсутствие зависимости между входными переменными.

К сожалению, очевидных зависимостей между переменными не наблюдается ни на графиках рассеяния, ни в корреляционной матрице.

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

Выбросы определены методом межквартильных расстояний, то есть выбросами считаются те значения, которые на 1,5 межквартильного размаха меньше первого квартиля распределения (25% выборки) и на 1,5 межквартильного размаха большего третьего квартиля (75% выборки). Таким образом, найдено и удалено 93 выброса.

После удаления выбросов в датасете осталось 936 строк и 13 признаков-переменных.

После удаления выбросов ситуация принципиально не изменилась, коэффициенты корреляции целевых переменных с входными не приблизились к 0,2, корреляция близка к 0, что говорит об отсутствии линейных связей между признаками.

Далее применены освоенные на курсе методы - нескольких алгоритмов (моделей) и сравнение результатов по ключевой метрике R2 (коэффициенту детерминации).

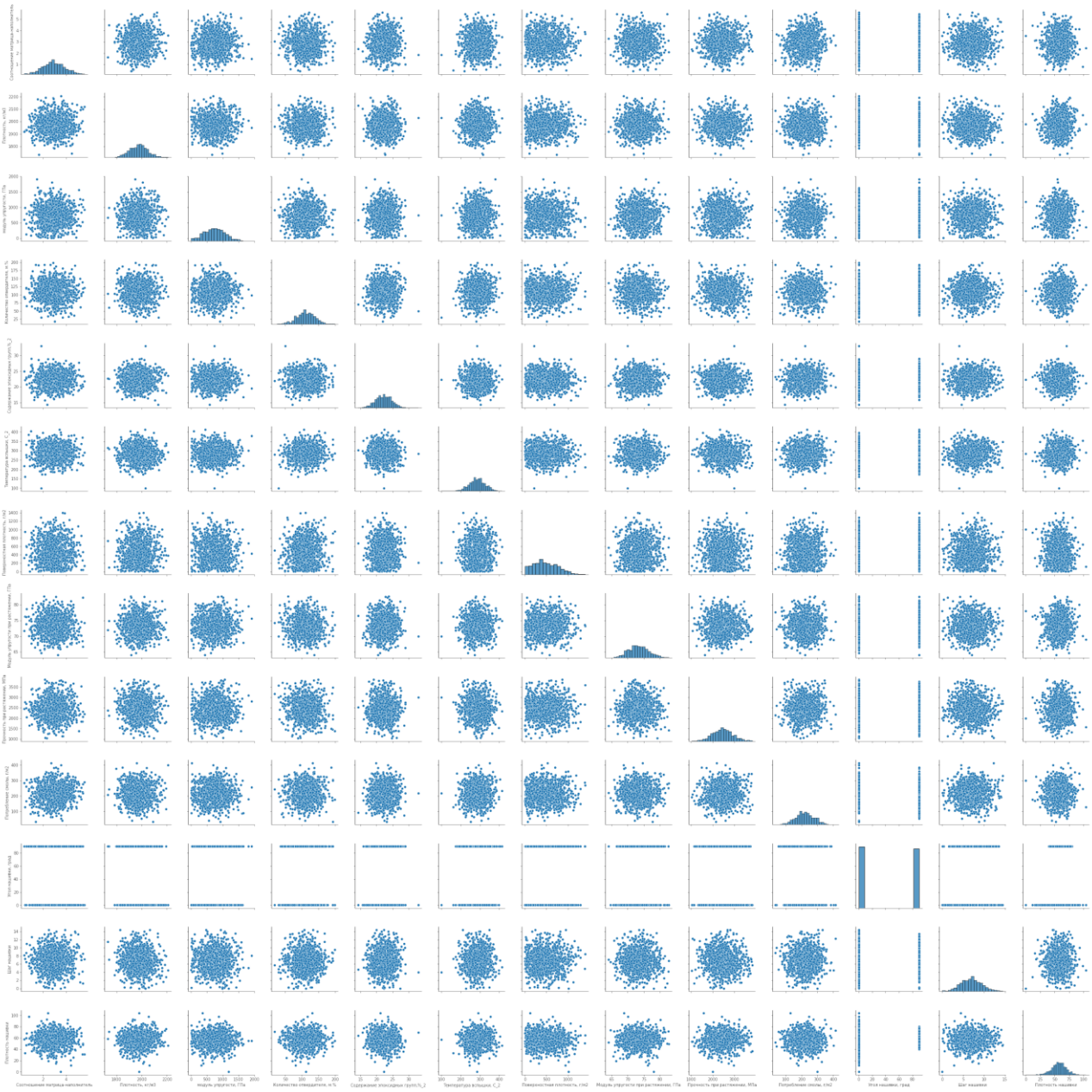

Рисунок 5 - Попарные графики рассеяния точек

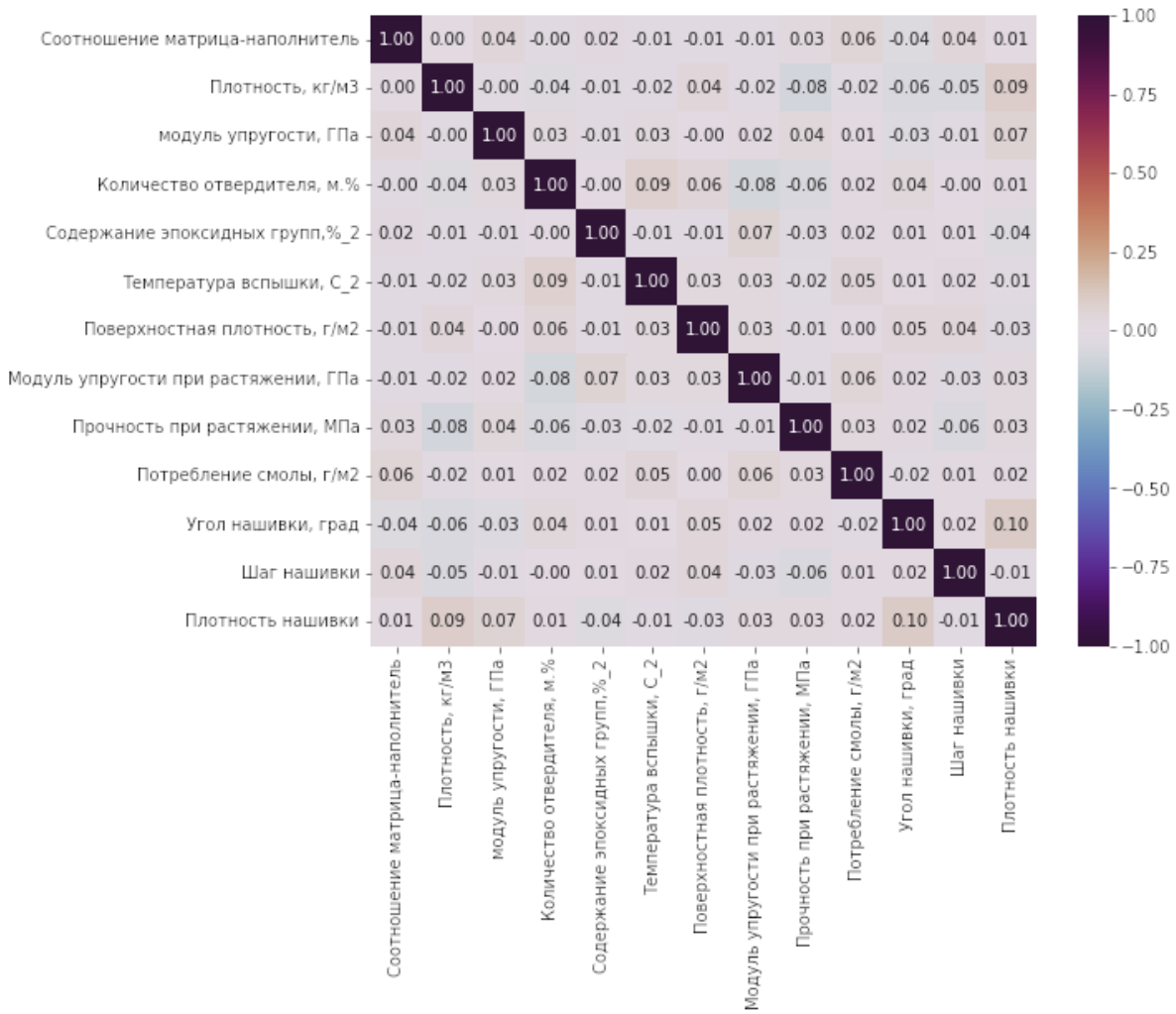

Рисунок 6 — Матрица корреляции

2. Практическая часть

2.1. Предобработка данных

Для обработки с помощью моделей данные были разделены следующим образом: - для каждой целевой переменной была сформирована отдельная выборка из оставшихся двенадцати признаков.

Далее выборки входных данных для модуля упругости и прочности при растяжении были разделены на обучающую и тестовую выборки в соотношении 70% и 30%, и подвергнуты масштабированию (нормализации) - приведению каждого признака к диапазону от 0 до 1 при помощи метода MinMaxScaler (минимальное значение признака делится на диапазон (разницу между максимальным и минимальным), выходные переменные остались без изменений .

Масштабирование — это необходимый шаг, потому что признаки измеряются в разных единицах, а значит покрывают разные диапазоны. Это влияет на работу алгоритмов, которые учитывают расстояния между измерениями. Масштабирование позволяет этого избежать. Масштабировать буду с помощью приведения каждого признака к диапазону от 0 до 1 при помощи метода MinMaxScaler (минимальное значение признака делится на диапазон (разницу между максимальным и минимальным).

О предобработке выборки для нейронной сети для прогнозирования соотношения матрица-наполнитель указано в подразделе 2.1.3 .

2.2. Разработка и обучение моделей

В данной работе для прогнозирования модуля упругости и прочности при растяжении применялись следующие модели: LinearRegression, Lasso, Ridge, SVR, KNeighborsRegressor, DecisionTreeRegressor RandomForestRegressor, GradientBoostingRegressor, а нейронная сеть - для прогнозирования соотношения матрица-наполнитель.

Модели запускались одной функцией (написанной при помощи метода def) с метриками - коэффициент детерминации (R2), MSE и MAE, использовалась кросс-валидация, количество блоков равно 10, возвращающей список моделей и средние значения указанных метрик, сортировка по уменьшению R2:

```
def run_regressors(models, x, y):
    stat = pd.DataFrame()
    cv = KFold(10, shuffle=True, random_state=42)
    scoring = ['r2', 'neg_root_mean_squared_error', 'neg_mean_absolute_error']
    for model_name, model in models.items():
        scores = cross_validate(model, x, y, cv=cv, scoring=scoring)
        stat.loc[model_name, 'R2'] = scores['test_r2'].mean()
        stat.loc[model_name, 'RMSE'] = scores['test_neg_root_mean_squared_error'].mean()
        stat.loc[model_name, 'MAE'] = scores['test_neg_mean_absolute_error'].mean()
    return print(stat.sort_values(by='R2', ascending = False))
```

Сначала модели запускались с гиперпараметрами по умолчанию, потом осуществлялся подбор гиперпараметров также по новой функции для запуска подбора гиперпараметров классом GridSearchCV, возвращающий список моделей с лучшими параметрами:

```
models_with_params = {}
def run_grid_search(model, parameters, X_train, y_train):
    grid = GridSearchCV(model, parameters, cv=10, n_jobs=-1)
    grid.fit(X_train, y_train)
    models_with_params[str(grid.best_estimator_)] = grid.best_estimator_
    return print(grid.best_estimator_)
```

Потом модели с подобранными гиперпараметрами запускались функцией def run_regressors для вывода метрик на обучении, метрики сравнивались и тремя лучшими моделями делались предсказания, предсказания визуализировались на графике вместе с тестовыми значениями, снимались финальные метрики с результатов прогнозов и выбиралась лучшая модель отдельно для модуля упругости и для прочности при растяжении.

В этой работе использовались следующие основные метрики:

– R2 или коэффициент детерминации измеряет долю дисперсии, объясненную моделью, в общей дисперсии целевой переменной. Если он близок к единице, то модель хорошо объясняет данные, если же он близок к нулю, то прогнозы сопоставимы по качеству с константным предсказанием;

– RMSE (Root Mean Squared Error) или корень из средней квадратичной ошибки принимает значениях в тех же единицах, что и целевая переменная. Метрика использует возведение в квадрат, поэтому хорошо обнаруживает грубые ошибки, но сильно чувствительна к выбросам;

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

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

R2 в норме принимает положительные значения. Эту метрику надо максимизировать. Отрицательные значение коэффициента детерминации означают плохую объясняющую способность модели.

Отдельно пробовалось улучшить работу линейной регрессии с помощью Polynomial Features из Scikit-Learn, добавляя квадрат всех функций, присутствующих в обучающих данных, в качестве новых функций для модели. Но модель с Polynomial Features отработала на тестовых выборках хуже лучших моделей.

Функция запуска:

def polynomial_regression_model(degree, X_train, Y_train, X_test, Y_test):

```
poly_features = PolynomialFeatures(degree=degree)
X_train_poly = poly_features.fit_transform(X_train)
poly_model = LinearRegression()
poly_model.fit(X_train_poly, Y_train)
y_train_predict = poly_model.predict(X_train_poly)
y_test_predict = poly_model.predict(poly_features.fit_transform(X_test))
rmse_train = np.sqrt(mean_squared_error(Y_train, y_train_predict))
r2_train = r2_score(Y_train, y_train_predict)
rmse_test = np.sqrt(mean_squared_error(Y_test, y_test_predict))
r2_test = r2_score(Y_test, y_test_predict)
print("Метрики на обучении")
print("----------------------------------------")
print("RMSE {}".format(rmse_train))
print("R2 {}".format(r2_train))
print("\n")
print("Метрики на тесте")
print("----------------------------------------")
print("RMSE {}".format(rmse_test))
print("R2 {}".format(r2_test))
```

Результаты далее.

2.2.1 Модели для прогнозирования модуля упругости при растяжении

	R2	RMSE	MAE
Lasso	-0.011974	-3.129735	-2.546156
LinearRegression	-0.050729	-3.185634	-2.586308
SVR	-0.082517	-3.229031	-2.620525
RandomForestRegressor	-0.089148	-3.243690	-2.622511
KNeighborsRegressor	-0.305670	-3.541082	-2.868406
DecisionTreeRegressor	-1.186810	-4.592840	-3.753044

Рисунок 7 – Метрики на обучении моделей с параметрами по умолчанию

```
                                                      R2      RMSE  \
Lasso(alpha=0.1)                                  -0.011974 -3.129735
SVR(C=0.01, kernel='sigmoid')                     -0.013537 -3.132032
DecisionTreeRegressor(max_depth=3, max_features... -0.018151 -3.138941
RandomForestRegressor(bootstrap='True', max_dep... -0.037446 -3.167249
LinearRegression(fit_intercept='True')            -0.050729 -3.185634
KNeighborsRegressor(n_neighbors=29)               -0.054548 -3.191233

                                                       MAE
Lasso(alpha=0.1)                                  -2.546156
SVR(C=0.01, kernel='sigmoid')                     -2.547788
DecisionTreeRegressor(max_depth=3, max_features... -2.551844
RandomForestRegressor(bootstrap='True', max_dep... -2.578357
LinearRegression(fit_intercept='True')            -2.586308
KNeighborsRegressor(n_neighbors=29)               -2.602919
```

Рисунок 8 – Метрики на обучении моделей с подобранными гиперпараметрами

Вывод: подбором гиперпараметром не удалось достичь положительного коэффициента детерминации R2 при обучении моделей для предстаказания Модуля упругости при растяжении, тем не менее подбор улучшил показатели моделей, но, к сожалению, R2 у всех отрицательный, а значит работают даже хуже простого усреднения.

Не изменились метрики у LinearRegression и Lasso, т.к. у LR параметр (fit_intercept='True') является параметром по умолчанию, а вот результаты lasso вызывает подозрение. Модель Lasso сработала некорректно, вероятно она получает на выходе прямую линию, поэтому её исключаю из сравнения.

Лучшими моделями себя показали SVR, DecisionTreeRegressor, RandomForestRegressor с подобранными параметрами:

```
SVR(C=0.01, kernel='sigmoid'),
DecisionTreeRegressor(max_depth=3, max_features=2, min_samples_leaf=7,
                      random_state=42, splitter='random'),
RandomForestRegressor(bootstrap='True', max_depth=2, n_estimators=50,
                      random_state=42) .
```

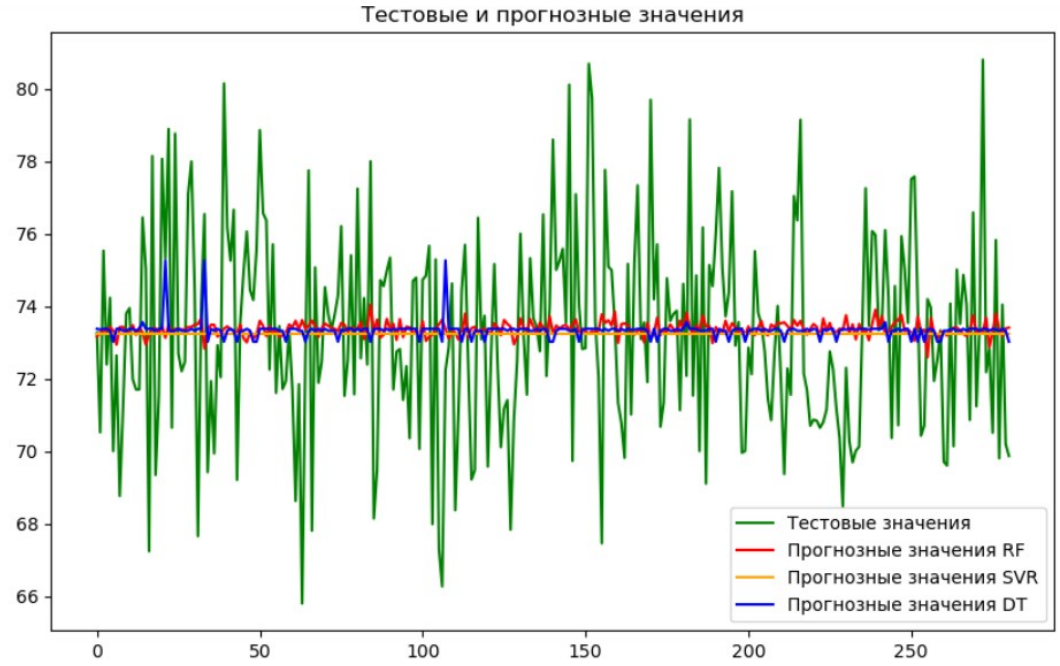

Рисунок 9 – Визуализация предсказаний вместе с тестовыми значениями

	R2	RMSE	MAE
RandomForest	-0.004116	-2.789714	-2.224350
SVR	-0.000030	-2.784032	-2.228550
DecisionTree	0.002073	-2.781103	-2.223499

Рисунок 10 – Метрики на тесте моделей с подобранными гиперпарамет-рами

```
Метрики на обучении
-----------------------------------------
RMSE 2.8637516673670156
R2 0.1667209710199583


Метрики на тесте
-----------------------------------------
RMSE 3.0909661095843077
R2 -0.23268738762106822
```

Рисунок 11 – Метрики polynomial_regression_model

Итог: Лучшей моделью является DecisionTreeRegressor(max_depth=3, max_features=2, min_samples_leaf=7, random_state=42, splitter='random').

2.2.2 Модели для прогнозирования прочности при растяжении

	R2	RMSE	MAE
SVR	-0.017829	-462.124894	-368.182686
Ridge	-0.035964	-465.952762	-372.252565
LinearRegression	-0.037942	-466.380420	-372.716892
GradientBoostingRegressor	-0.104503	-481.199704	-387.761514
DecisionTreeRegressor	-1.141230	-666.306515	-539.118955

Рисунок 12 – Метрики на обучении моделей с параметрами по умолчанию

	R2	RMSE	MAE
GradientBoostingRegressor(max_depth=1, max_feat...	-0.008436	-459.878249	-367.513068
DecisionTreeRegressor(criterion='poisson', max_...	-0.011705	-460.674928	-366.320557
SVR(C=0.5, kernel='sigmoid')	-0.017800	-462.117096	-368.159793
Ridge(alpha=610, positive=True, solver='lbfgs')	-0.019710	-462.490354	-368.661097

Рисунок 13 – Метрики на обучении моделей с подобранными гиперпараметрами (LR не участвовала)

Вывод: в случае с Прочностью при растяжении так же подбором гиперпараметров не удалось достичь положительного коэффициента детерминации R2 при обучении моделей для предсказаний, тем не менее подбор улучшил показатели моделей, но, к сожалению, R2 у всех отрицательный, а значит они работают даже хуже простого усреднения.

Лучшими моделями себя показали GradientBoostingRegressor, DecisionTreeRegressor и SVR с подобранными параметрами:

```
SVR(C=0.5, kernel='sigmoid')
DecisionTreeRegressor(criterion='poisson', max_depth=1, max_features=6,
                      min_samples_leaf=3, random_state=42)
GradientBoostingRegressor(max_depth=1, max_features=1, n_estimators=50,
```

```
random_state=42).
```

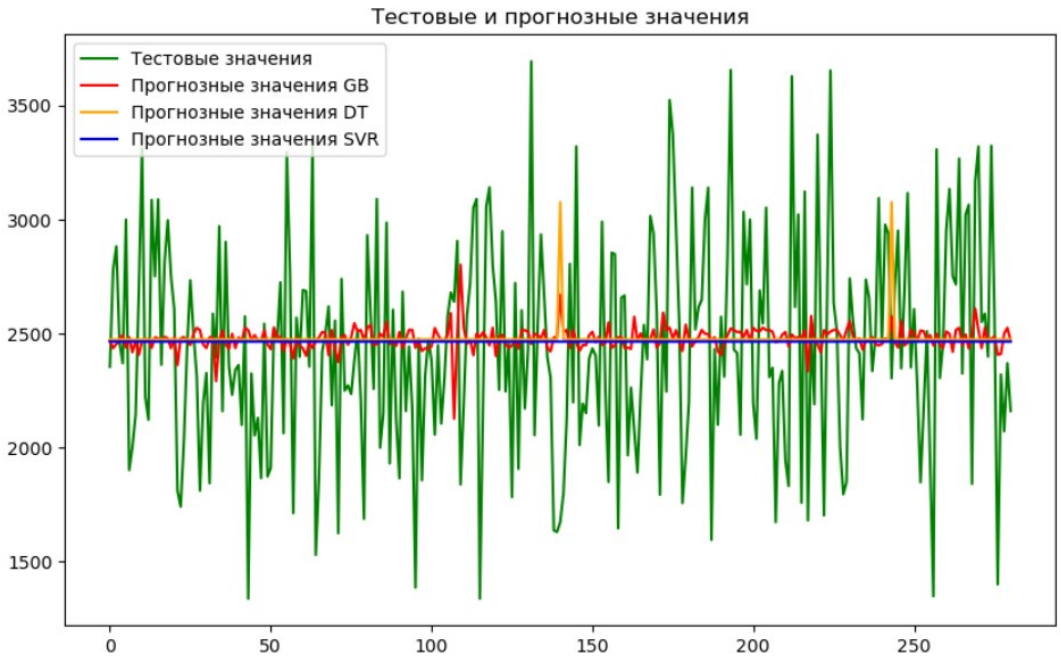

Рисунок 14 – Визуализация предсказаний вместе с тестовыми значениями

	R2	RMSE	MAE
GradientBoosting	-0.011825	-468.558170	-373.390534
DecisionTree	-0.035476	-474.002696	-375.698009
SVR	-0.002050	-466.289316	-370.630697

Рисунок 15 – Метрики на тесте моделей с подобранными гиперпараметрами

```
Метрики на обучении
-------------------------------------------
RMSE 425.14453540723554
R2 0.15416224154768499


Метрики на тесте
-------------------------------------------
RMSE 508.8288573483319
R2 -0.1932235848169599
```

Рисунок 16 – Метрики polynomial_regression_model

Итог: Лучшей моделью является SVR(C=0.5, kernel='sigmoid').

2.2.3 Разработка нейронной сети для прогнозирования соотношения матрица-наполнитель

В случае с соотношением матрица-наполнитель датасет нормализовался полностью до разделения на вход (X) и выход (y), таким образом предсказания требуется вернуть к исходному масштабу.

Нейронная сеть 1 создана с помощью класса keras.Sequential со следующими параметрами:

– входной слой для 12 признаков;

– выходной слой для 1 признака;

– скрытых слоев: 2;

– нейронов на каждом скрытом слое: 8;

– активационная функция скрытых слоев: relu;

– оптимизатор: Adam;

– loss-функция: MeanAbsoluteError.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                650

 dense_1 (Dense)             (None, 8)                 408

 dense_2 (Dense)             (None, 8)                 72

 dense_3 (Dense)             (None, 1)                 9

=================================================================
Total params: 1,139
Trainable params: 1,139
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 17 — Архитектура нейросети 1

Обучение запускалось со следующими параметрами:

– пропорция разбиения данных на тестовые и валидационные: 30%;

– количество эпох: 50.

– раннюю остановку не использую.

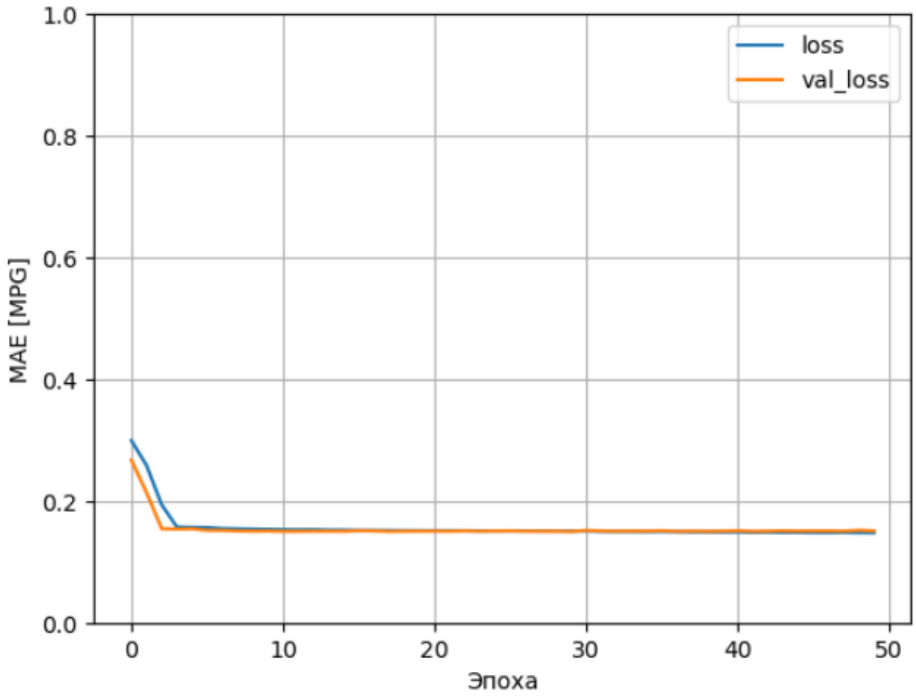

Рисунок 18 — График обучения нейросети 1

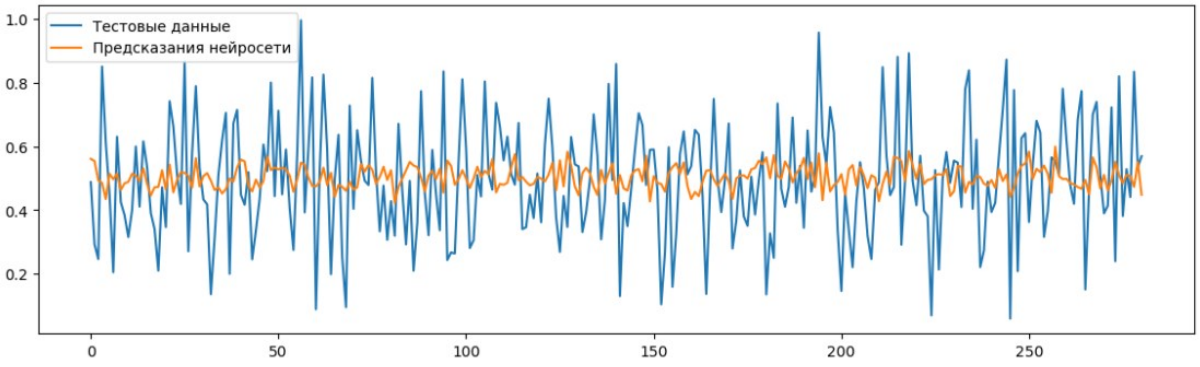

Рисунок 19 — Визуализация предсказаний вместе с тестовыми значениями нейросети 1

Нейронная сеть 2 создана с помощью класса keras.Sequential со следующими параметрами:

– входной слой для 12 признаков;

– выходной слой для 1 признака;

– скрытых слоев: 3;

– слоев Dropout с параметром 0.12: 3

– нейронов на каждом скрытом слое: 8;

– активационная функция скрытых слоев: relu;

– оптимизатор: Adam;

– loss-функция: MeanAbsoluteError.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 24)                312

 dropout (Dropout)           (None, 24)                0

 dense_5 (Dense)             (None, 128)               3200

 dropout_1 (Dropout)         (None, 128)               0

 dense_6 (Dense)             (None, 64)                8256

 dropout_2 (Dropout)         (None, 64)                0

 dense_7 (Dense)             (None, 32)                2080

 dense_8 (Dense)             (None, 1)                 33

=================================================================
Total params: 13,881
Trainable params: 13,881
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 20 — Архитектура нейросети 2

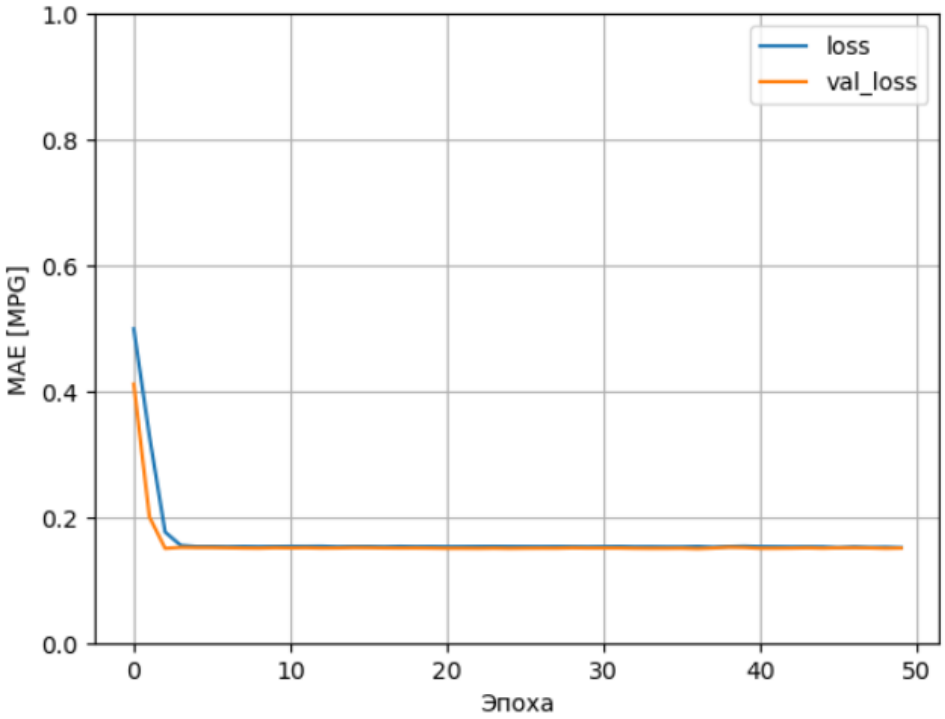

Рисунок 21 — График обучения нейросети 2

Обучение запускалось со следующими параметрами:

– пропорция разбиения данных на тестовые и валидационные: 30%;

– количество эпох: 50.

– раннюю остановку не использую.

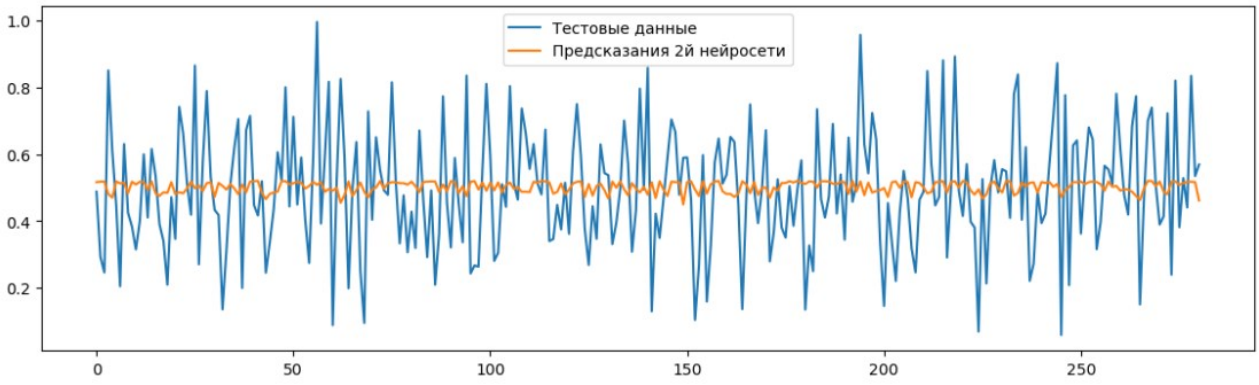

Рисунок 22 — Визуализация предсказаний вместе с тестовыми значениями нейросети 2

	R2	RMSE	MAE
Нейросеть 1	-0.033967	-0.186174	-0.150316
Нейросеть 2	-0.021892	-0.185084	-0.147983

Рисунок 23 – Метрики нейросетей на тесте

Итог: Вторая нейронная сеть отработала немного лучше. Хотя обе дали предсказания хуже усреднения (отрицательный R2).

В завершение предсказанию лучшей модели возвращён реальный масштаб таким путём:

```
min = np.min(ds['Соотношение матрица-наполнитель'].values)
max = np.max(ds['Соотношение матрица-наполнитель'].values)
y3_pred_2_res = y3_pred_2 * (max - min) + min
```

2.3. Разработка приложения

Эту задачу не получилось решить.

2.4. Создание удаленного репозитория

Для данного исследования был создан удаленный репозиторий на GitHub, который находится по адресу https://github.com.... На него были загружены результаты работы: исследовательский notebook, код приложения.

Заключение

В ходе выполнения данной работы были применены методы, освоенные на курсе. К сожалению, не удалось разработать достоверные прогнозные модели.

Библиографический список

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

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

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

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

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

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

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

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

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

10 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/.

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