

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

Агаметова А.С.

Москва, 2022

Содержание:

Введение	3
1. Аналитическая часть	4
1.1. Постановка задачи.....	4
1.2. Описание используемых методов.....	5
1.3. Разведочный анализ данных	9
2. Практическая часть	13
2.1. Предобработка данных	13
2.2. Разработка и обучение модели.....	16
2.3. Оценка точности работы моделей.	22
2.5 Создание удаленного репозитория.	25
Заключение.....	26
Список используемой литературы.	27

Введение

В настоящее время большое внимание уделяется вопросу использования веществ с повышенными физико-химическими свойствами, в сравнении с общепринятыми материалами; такими свойствами обладают композиты. Композитный материал – компонентный материал, в структуру которого входит пластичная основа (матрица), дополненная сочетанием нескольких химически разнородных компонентов с границей раздела между ними. Данная взаимосвязь приводит к образованию нового вещества, свойства которого отличаются от свойств каждого из его компонентов. Варьируя комбинации основы, армирующих компонентов и наполнителей, можно подвести к широкому диапазону материалов с нужным количеством свойств, что и делает большинство композитов по своим характеристикам лучше, чем общепринятые материалы и сплавы. Возможность получения материалов с уникальными свойствами не присущими отдельным компонентам обуславливает широкое применение композиционных материалов в различных областях. Примеры композита – железобетон (сочетание стальной арматуры и бетона) или его потенциальная альтернатива – композитный бетон (замена стали на базальтовую или стеклопластиковую арматуру).

Разработка новых видов композитных материалов дает дополнительное развитие и оптимизацию в различных индустриях, одновременно снижая затраты на производство.

Однако, как и любое исследование, разработка композитных материалов – это длительный и ресурсозатратный процесс, так как по характеристикам отдельных компонентов невозможно рассчитать итоговые свойства композита. Для получения предварительного результата требуется сравнительный анализ большого количества данных с последующим испытаниями, что и делает актуальной задачу прогнозирования при разработке новых материалов.

1. Аналитическая часть

1.1. Постановка задачи.

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

Описание:

Современные композиты изготавливаются из других материалов: полимеры, керамика, стеклянные и углеродные волокна, но данный принцип сохраняется. У такого подхода есть и недостаток: даже если мы знаем характеристики исходных компонентов, определить характеристики композита, состоящего из этих компонентов, достаточно проблематично. Для решения этой проблемы есть два пути: физические испытания образцов материалов, или прогнозирование характеристик. Суть прогнозирования заключается в симуляции представительного элемента объема композита, на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

Основной целью данной работы является создание системы прогнозирования параметров композитного материала на основе методов машинного обучения. Для достижения ее достижения необходимо определить следующие задачи:

- 1) провести разведочный анализ данных;
- 2) провести предобработку обучающих и тестовых выборок;
- 3) создать и обучить несколько моделей для прогноза модуля упругости при растяжении и прочности при растяжении;
- 4) разработать нейронную сеть, для рекомендации (прогноза) соотношения матрица-наполнитель;
- 5) оценить точность модели на тренировочном и тестовом наборах данных.

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

спрогнозировать ряд конечных свойств получаемых композиционных материалов. Кейс основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

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

1.2. Описание используемых методов

Регрессионный анализ - статистический аналитический метод, позволяющий вычислить предполагаемые отношения между зависимой переменной одной или несколькими независимыми переменными. Используя регрессионный анализ, можно моделировать отношения между выбранным переменными, а также прогнозируемыми значениями на основе модели.

Для решения поставленной задачи применим следующие методы:

- линейная регрессия;
- метод ближайших соседей;
- дерево решений;
- градиентный бустинг;
- метод опорных векторов;
- «Случайный лес».

Линейная регрессия (Linear regression) — модель зависимости переменной X от одной или нескольких других переменных (факторов, регрессоров, независимых переменных) с линейной функцией зависимости.

Линейная регрессия относится к задаче определения «линии наилучшего соответствия» через набор точек данных и стала простым предшественником нелинейных методов, которые используют для обучения нейронных сетей.

Преимущества: чрезвычайно простой, легкий и интуитивно понятный в использовании метод.

Недостатки: линейная регрессия только моделирует отношения между зависимыми и независимыми переменными, которые являются линейными. Предполагается, что между ними существует прямая связь, которая иногда неверна. Линейная регрессия очень чувствительна к аномалиям в данных (или выбросам). Другим недостатком является то, что если у нас есть несколько параметров, а не количество доступных выборок, то модель начинает моделировать шум, а не взаимосвязь между переменными.

Метод ближайших соседей (k Nearest Neighbours) — метрический алгоритм для автоматической классификации объектов или регрессии. В случае использования метода для классификации объект присваивается тому классу, который является наиболее распространённым среди k соседей данного элемента, классы которых уже известны.

Преимущества: высокая точность, нечувствительность к выбросам и отсутствие допущений о вводе данных.

Недостатки: высокая временная сложность и большая пространственная сложность. Когда выборка несбалансирована, например, размер выборки одного класса очень велик, а размер выборки других классов очень мал. При вводе выборки большинство K соседних значений являются классом с большим размером выборки. Вызывает ошибки классификации. Метод улучшения состоит в том, чтобы взвесить K соседних точек, то есть точки, которые находятся ближе, имеют больший вес, а точки, которые находятся дальше, имеют меньший вес. Количество вычислений велико. Каждая классифицируемая выборка должна рассчитываться по ее расстоянию до всех точек. В соответствии с сортировкой по расстоянию можно получить K соседних точек.

Дерево решений (Decision Tree) — метод, который позволяет последовательно представить альтернативные варианты решений с их выходными данными и соответствующей неопределенностью. Как и при

выполнении анализа дерева событий, построение следует начинать с начального события или принятого решения. Далее необходимо построить пути развития событий, определить результаты, которые могут быть получены при реализации событий, и различные решения, которые могут быть приняты.

Преимущества: метод обеспечивает точное графическое представление всех деталей решения проблемы и позволяет рассчитать лучшие пути решения проблемы.

Недостатки: большие деревья решений слишком сложны для обмена информацией с заинтересованными сторонами. Применение диаграммы дерева решений может привести к излишнему упрощению ситуации.

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

Преимущества: на сегодняшний день является одним из самых мощных алгоритмов распознавания и предоставляет множество возможностей для вариаций - можно рассматривать различные функции потерь, возможно рассмотрение любого семейства базовых алгоритмов, в каждой конкретной вариации бустинга не сложно провести некоторые математические и алгоритмические оптимизации, которые заметно ускорят работу алгоритма.

Недостатки: бустинг – трудоемкий метод, и работает он достаточно медленно. Без дополнительных модификаций он имеет свойство полностью подстраиваться под данные, в том числе под ошибки и выбросы в них. Идея бустинга обычно плохо применима к построению композиции из достаточно сложных и мощных алгоритмов, также результаты работы бустинга сложно интерпретируемы, особенно если в композицию входят десятки алгоритмов.

Метод опорных векторов (Support vector machines) - семейство алгоритмов бинарной классификации, основанных на обучении с учителем,

использующих линейное разделение пространства признаков с помощью гиперплоскости.

Преимущества: метод сводится к решению задачи квадратичного программирования в выпуклой области и является наиболее быстрым нахождения решающих функций; он также находит разделяющую полосу максимальной ширины, что позволяет в дальнейшем осуществлять более уверенную классификацию.

Недостатки: метод чувствителен к шумам и стандартизации данных, и в нем не существует общего подхода к автоматическому выбору ядра (и построению спрямляющего подпространства в целом) в случае линейной неразделимости классов.

Случайный лес (Random forest, RF) — это алгоритм обучения с учителем. Его можно применять как для классификации, так и для регрессии. Также это наиболее гибкий и простой в использовании алгоритм. RF создает деревья решений для случайно выбранных семплов данных, получает прогноз от каждого дерева и выбирает наилучшее решение посредством голосования. Он также предоставляет довольно эффективный критерий важности показателей (признаков).

Благодаря своей гибкости Random Forest применяется для решения практически любых проблем в области машинного обучения. Сюда относятся классификации (RandomForestClassifier) и регрессии (RandomForestRegressor), а также более сложные задачи, вроде отбора признаков, поиска выбросов/аномалий и кластеризации.

Преимущества: случайный лес считается высокоточным и надежным методом, не страдает проблемой переобучения, также может работать с отсутствующими значениями.

Недостатки: метод довольно медленный, так как для работы алгоритм использует множество слоев (деревьев). Модель случайного леса сложнее интерпретировать по сравнению с деревом решений, где вы легко определяете результат, следуя по пути в дереве.

1.3.Разведочный анализ данных

Для анализа предоставлено два файла с входными данными о параметрах базальтопластика X_br.xlsx и нашивки углепластика X_nup.xlsx, на основании которых необходимо подготовить модели для прогноза модуля упругости при растяжении, прочности при растяжении и соотношения «матрица-наполнитель».

Анализ данных, их предобработка, построение моделей выполнены при помощи языка программирования Python с использованием библиотек: Numpy, Pandas и Matplotlib. После объединения двух файлов при первичном анализе данных можно отметить, что в набор содержится 13 признаков, каждый из которых содержит 1023 значения, пропуски отсутствуют, все признаки числовые в соответствии с рисунком 1.

```
[6]: df.info()

<class 'pandas.core.frame.DataFrame'>
Float64Index: 1023 entries, 0.0 to 1022.0
Data columns (total 13 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Угол нашивки, град                       1023 non-null   float64
1   Шаг нашивки                             1023 non-null   float64
2   Плотность нашивки                       1023 non-null   float64
3   Соотношение матрица-наполнитель         1023 non-null   float64
4   Плотность, кг/м3                        1023 non-null   float64
5   модуль упругости, ГПа                   1023 non-null   float64
6   Количество отвердителя, м.%             1023 non-null   float64
7   Содержание эпоксидных групп,%_2        1023 non-null   float64
8   Температура вспышки, C_2               1023 non-null   float64
9   Поверхностная плотность, г/м2          1023 non-null   float64
10  Модуль упругости при растяжении, ГПа    1023 non-null   float64
11  Прочность при растяжении, МПа           1023 non-null   float64
12  Потребление смолы, г/м2                 1023 non-null   float64
dtypes: float64(13)
memory usage: 111.9 KB
```

Рисунок 1 - Первичный анализ данных

Описательная статистика, представленная на рисунке 2, содержит следующие данные:

- общее количество;
- среднее значение;
- стандартное отклонение;
- минимальное и максимальное значение;
- квантили.

Описательная статистика полученных данных									
[8]: df.describe().T									
[8]:		count	mean	std	min	25%	50%	75%	max
	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
	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

Рисунок 2 - Описательная статистика

Отсутствие каких-либо данных в квартилях 25% и 50% может указывать на минимальное количество значений для признака «Угол нашивки, град».

Для визуализации описательной статистики воспользуемся гистограммами и диаграммами размаха.

Гистограмма позволит наглядно представить тенденции изменения измеряемых параметров качества объекта и зрительно оценить их распределение. Кроме того, гистограмма дает возможность быстро определить центр, разброс и форму распределения случайной величины.

По форме распределения можно судить о природе исследуемой переменной (например, бимодальное распределение позволяет предположить, что выборка не является однородной и содержит наблюдения, принадлежащие двум различным множествам, которые в свою очередь нормально распределены).

На рисунке 3 можно видеть, что все признаки имеют распределение близкое к нормальному, кроме «Угол нашивки» (принимает два значения: 0 и 90), и принимают только неотрицательные значения. Хотелось бы отметить асимметрию влево «Поверхностной плоскости», но тут скорее всего речь не о нарушениях, а о сортировке результатов, которые выпадают за пределы верхней границы допуска.

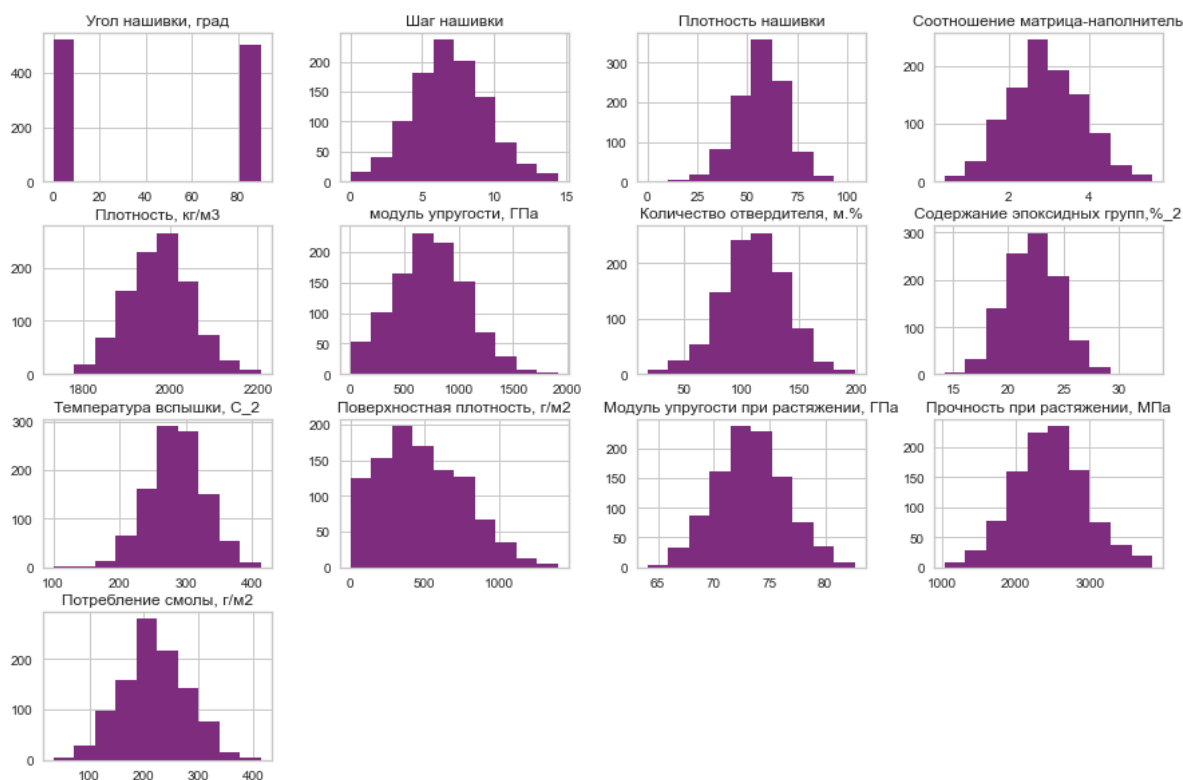


Рисунок 3 - Гистограммы распределения переменных.

Для определения наличия выбросов в ненормализованных данных построим для каждого признака диаграмму размаха в соответствии с рисунком 4; выбросы для дальнейшей работы будут удалены. Такой вид диаграммы в удобной форме показывает медиану (или, если нужно, среднее), нижний и верхний квартили, минимальное и максимальное значение выборки и выбросы

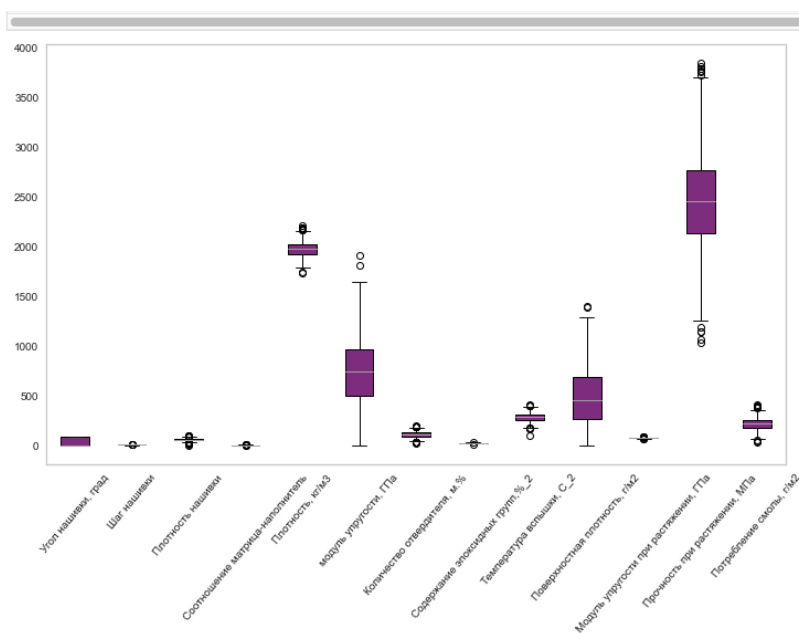


Рисунок 4 - Диаграммы размаха (выбросы).

Матрица корреляции и матрица рассеивания позволяют визуализировать попарную взаимосвязь между признаками. Достаточно будет оценить данные рисунке 5 и отметить отсутствие попарной корреляции, что указывает на нелинейный характер связей между ними.

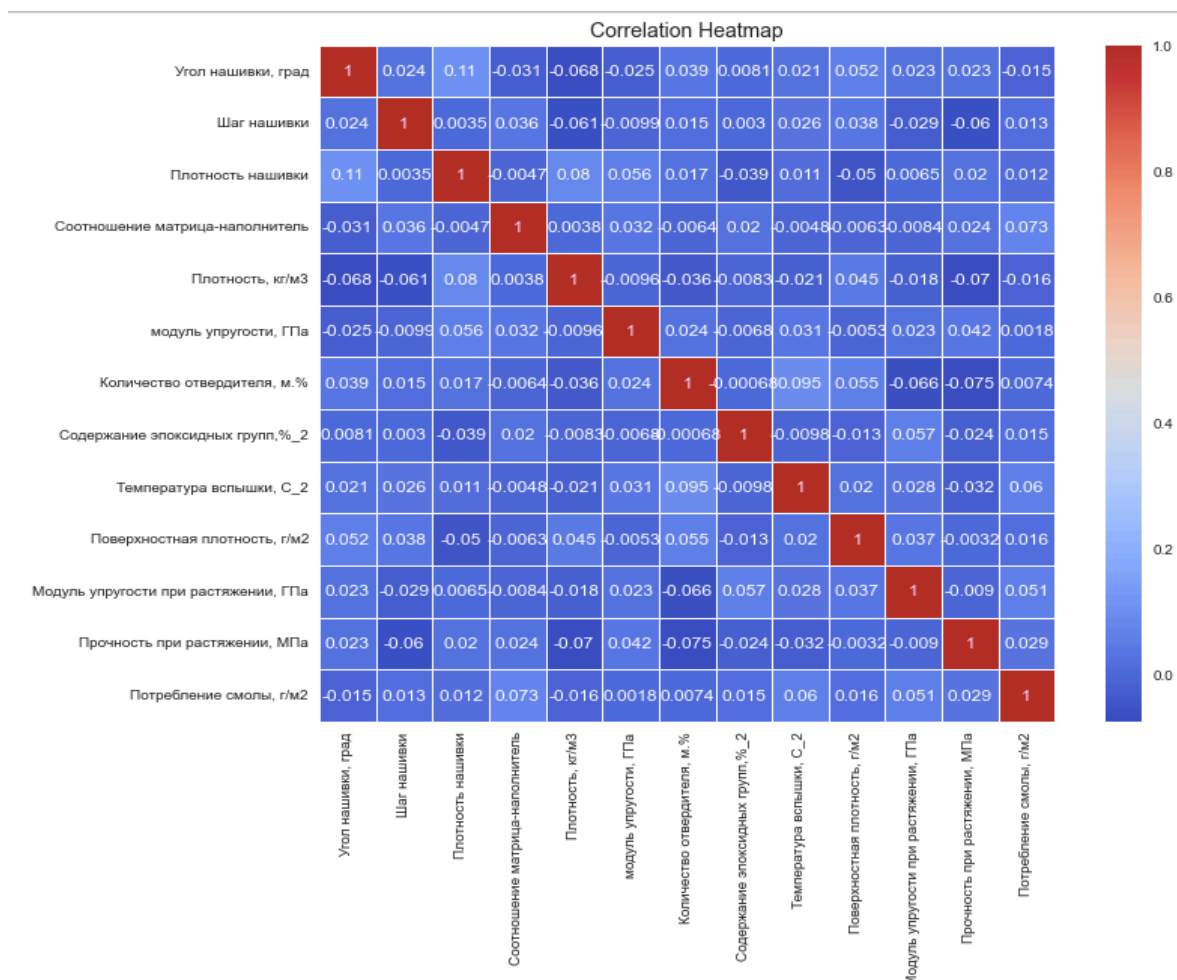


Рисунок 5 - Матрица корреляций

2. Практическая часть

2.1. Предобработка данных

Первичные данные покрывают разные диапазоны, поэтому для получения более сглаженного вида и во избежание искажения результатов, в таких алгоритмах как метод опорных векторов и ближайших соседей, проведем нормализацию данных – перекодируем величины в диапазоне $[0,1]$ с помощью `MinMaxScaler()`. На рисунке 6 можно оценить описательную статистику уже нормализованных данных – все данные приведены в соответствующий диапазон.

norm_df.describe().T								
	count	mean	std	min	25%	50%	75%	max
Угол нашивки, град	1023.0	0.491691	0.500175	0.0	0.0000	0.000	1.0000	1.0
Шаг нашивки	1023.0	0.477763	0.177518	0.0	0.3520	0.479	0.5950	1.0
Плотность нашивки	1023.0	0.549614	0.118772	0.0	0.4785	0.551	0.6245	1.0
Соотношение матрица-наполнитель	1023.0	0.488422	0.175554	0.0	0.3705	0.484	0.6080	1.0
Плотность, кг/м3	1023.0	0.512526	0.154903	0.0	0.4040	0.516	0.6080	1.0
модуль упругости, ГПа	1023.0	0.386300	0.172971	0.0	0.2605	0.386	0.5025	1.0
Количество отвердителя, м.%	1023.0	0.512272	0.156130	0.0	0.4120	0.512	0.6180	1.0
Содержание эпоксидных групп,%_2	1023.0	0.426212	0.128384	0.0	0.3390	0.425	0.5180	1.0
Температура вспышки, С_2	1023.0	0.593348	0.130698	0.0	0.5080	0.593	0.6800	1.0
Поверхностная плотность, г/м2	1023.0	0.344641	0.201093	0.0	0.1905	0.323	0.4950	1.0
Модуль упругости при растяжении, ГПа	1023.0	0.497891	0.167451	0.0	0.3860	0.495	0.6070	1.0
Прочность при растяжении, МПа	1023.0	0.508635	0.172734	0.0	0.3910	0.506	0.6155	1.0
Потребление смолы, г/м2	1023.0	0.484839	0.156879	0.0	0.3830	0.487	0.5870	1.0

Рисунок 6 - Описательная статистика (нормализованные данные).

После нормализации диаграммы размаха по каждому из признаков показывают некоторые значения за пределами полутора межквартильных расстояний от первого и третьего квартилей. Но принимая во внимание источник данных и целевое назначение – получение композита с уникальными свойствами, такие значения, возможно, не стоит трактовать, как выбросы, до

тех пор, пока их наличие в обучающей и тестовой выборке не будет негативно сказываться на точности предсказания модели.

Воспользуемся правилом «трех сигм» и удалим все выбросы из исходных нормализованных данных.

```
[16]: Угол нашивки, град      0
      Шаг нашивки            4
      Плотность нашивки      20
      Соотношение матрица-наполнитель  6
      Плотность, кг/м3       9
      модуль упругости, ГПа   2
      Количество отвердителя, м.%  14
      Содержание эпоксидных групп,%_2  2
      Температура вспышки, С_2    8
      Поверхностная плотность, г/м2  2
      Модуль упругости при растяжении, ГПа  6
      Прочность при растяжении, МПа  11
      Потребление смолы, г/м2     8
      dtype: int64
```

```
[17]: norm_df = norm_df.dropna(axis = 0)
      norm_df.isnull().sum()
```

```
[17]: Угол нашивки, град      0
      Шаг нашивки            0
      Плотность нашивки      0
      Соотношение матрица-наполнитель  0
      Плотность, кг/м3       0
      модуль упругости, ГПа   0
      Количество отвердителя, м.%  0
      Содержание эпоксидных групп,%_2  0
      Температура вспышки, С_2    0
      Поверхностная плотность, г/м2  0
      Модуль упругости при растяжении, ГПа  0
      Прочность при растяжении, МПа  0
      Потребление смолы, г/м2     0
      dtype: int64
```

Рисунок 7 - Удаление выбросов

На рисунке 8 с диаграммами размаха можно отметить, что медианы выровнялись с выборочным средним, кроме «Угол нашивки», представленного только двумя значениями (0 и 90).

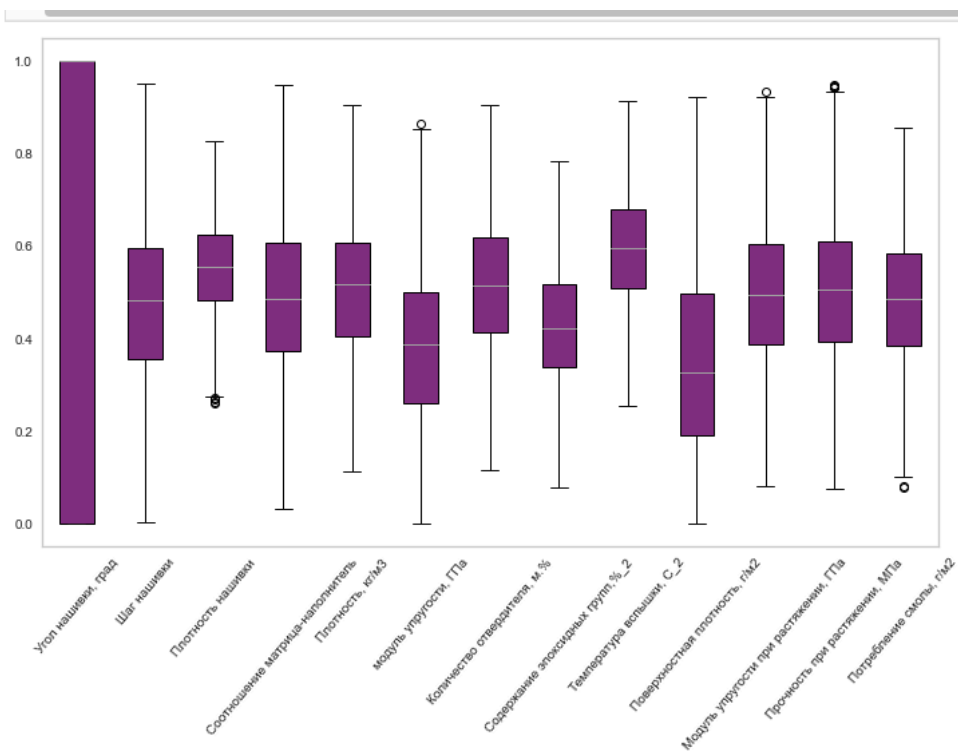


Рисунок 8 - Диаграммы размаха после нормализации и удаления выбросов.

После нормализации данных и удаления выбросов можно перейти к разработкам и обучению моделей.

2.2. Разработка и обучение модели

Разработка и обучение моделей машинного обучения в данной работе формируется для двух признаков - «Прочность при растяжении» и «Модуль упругости при растяжении». Для каждого признака будут построены две отдельные модели.

Предварительно необходимо выбрать исходный базовый уровень – некое предположение, с которым мы будем сравнивать результаты работы модели. Для регрессионных задач в качестве базового уровня разумно угадывать медианное значение цели на обучающем наборе для всех примеров в тестовом наборе.

За метрику возьмём среднюю абсолютную ошибку (MAE) в прогнозах – ее легко вычислить и удобно интерпретировать в соответствии с рисунком 9.

Выберем одну метрику, среднюю абсолютную ошибку (mae) в прогнозах, и с её помощью будем оценивать модели. Среднюю абсолютную ошибку легко вычислить и интерпретировать. Вычислим показатель для исходного базового уровня

```
def mae(y_true, y_pred):  
    y_true, predictions = np.array(y_true), np.array(y_pred)  
    return np.mean(np.abs(y_true - predictions))  
  
baseline_guess = np.median(y1)  
print('The baseline guess is a score of %0.2f' % baseline_guess)  
print("Baseline Performance for test data: MAE = %0.4f" % mae(y1_test, baseline_guess))  
  
The baseline guess is a score of 0.49  
Baseline Performance for test data: MAE = 0.1382
```

Рисунок 9 - MAE для базового уровня.

Данные разделим на обучающую и тестовую выборку в соотношении 70% и 30%. Обучающий набор признаков — то, что мы предоставляем нашей модели вместе с ответами в ходе обучения. Тестовый набор признаков используется для оценки обученной модели.

Модуль упругости при растяжении

```
[19]: x1 = norm_df.copy()
      y1 = x1.pop('Модуль упругости при растяжении, ГПа')
      y1

[19]: 1.0      0.319
      3.0      0.319
      4.0      0.319
      5.0      0.319
      6.0      0.319
      ...
     1018.0    0.485
     1019.0    0.476
     1020.0    0.573
     1021.0    0.536
     1022.0    0.551
      Name: Модуль упругости при растяжении, ГПа, Length: 937, dtype: float64

[20]: x1_train, x1_test, y1_train, y1_test = train_test_split(x1, y1,
                                                             test_size = 0.3,
                                                             random_state = 42)
```

Рисунок 12 - Выборки для построения модели «Модуль упругости при растяжении».

Для исследования вышеизложенной задачи воспользуемся следующими моделями машинного обучения:

- линейная регрессия;
- градиентный бустинг;
- к-ближайших соседей;
- дерево решений;
- метод опорных векторов;
- случайный лес.

Примеры использованных моделей к-ближайших соседей и дерево решений в соответствии с рисунком 13.

```

KNeighborsRegressor

[24]: KNN_1 = KNeighborsRegressor()
      KNN_1.fit(x1_train, y1_train)
      prediction_KNN = KNN_1.predict(x1_test)

      KNN_1_mae = np.mean(abs(prediction_KNN - y1_test))

      print('KNeighborsRegressor test data MAE is = %.4f' % KNN_1_mae)

      KNeighborsRegressor test data MAE is = 0.1503

[25]: KNN_1_model_result = pd.DataFrame({
      'MAE': np.mean(abs(prediction_KNN - y1_test)),
      }, index=['KNeighborsRegressor'])
      models = models.append(KNN_1_model_result)

/var/folders/zw/h5dlsnx954bds_97b36f830w0000gn/T/ipykernel_2454/848309772.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
      models = models.append(KNN_1_model_result)

DecisionTreeRegressor

[26]: from sklearn import tree

      DTR_1 = tree.DecisionTreeRegressor()
      DTR_1.fit(x1_train, y1_train)

      DTR_1_prediction = DTR_1.predict(x1_test)

      DTR_1_mae = np.mean(abs(DTR_1_prediction - y1_test))

      print('Decision Tree test data MAE is = %.4f' % DTR_1_mae)

      Decision Tree test data MAE is = 0.1898

[27]: DTR_1_model_result = pd.DataFrame({
      'MAE': np.mean(abs(DTR_1_prediction - y1_test)),
      }, index=['Decision Tree'])
      models = models.append(DTR_1_model_result)

/var/folders/zw/h5dlsnx954bds_97b36f830w0000gn/T/ipykernel_2454/3501961834.py:4: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat instead.
      models = models.append(DTR_1_model_result)

```

Рисунок 13 - Примеры использованных регрессий (KNN и дерево решений)

Во всех вышеперечисленных моделях для машинного обучения для сравнения была рассчитана средняя абсолютная ошибка. Для базового уровня MAE составила 0.1386. Результаты моделей представлены на рисунке 14.

	MAE
Linear Regression	0.138974
KNeighborsRegressor	0.150284
Decision Tree	0.188833
GradientBoosting	0.141703
Support Vector Machine (SVM)	0.148111
Random Forest	0.139369

Рисунок 14 - Результат работы моделей

Ни одна из моделей не показала улучшение, несмотря на тестирование и обучение. Наилучшей в списке с итогом 0,138974 оказалась линейная регрессия, но это скорее всего связано с тем, что остальные модели работают не совсем корректно, либо нужен другой подход, а модель просто подстраивается.

Регуляризацию проведем настройкой гиперпараметров с помощью поиска по сетке с перекрёстной проверкой. Для линейных моделей регуляризация обычно достигается путем ограничения веса модели. Обычные методы ограничения весов: регрессия гребня (Ridge), регрессия лассо и упругая регрессия.

```

Ridge_1 = linear_model.Ridge()
param_grid = {'alpha': [0.0001, 0.0002, 0.005, 0.05, 0.1, 1, 2, 5, 10, 100], "fit_intercept": [True, False]}
GSCV = GridSearchCV(estimator=Ridge_1, param_grid=param_grid, cv=10, verbose=2)
GSCV.fit(x1_train, y1_train)
GSCV.best_params_

```

```

Fitting 10 folds for each of 20 candidates, totalling 200 fits
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=True; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s
[CV] END .....alpha=0.0001, fit_intercept=False; total time= 0.0s

```

Рисунок 15 - Настройка гиперпараметров для регрессии гребня.

Модель после регуляризации улучшила результат: MAE is = 0.138294, но опытным наблюдением было замечено, что при изменении показателя alpha (от 10 до 100), модель всегда стремится к самому высокому показателю, тогда веса просто стремятся к нулю, и в результате получается просто горизонтальная линия, пересекающая среднее значение данных.

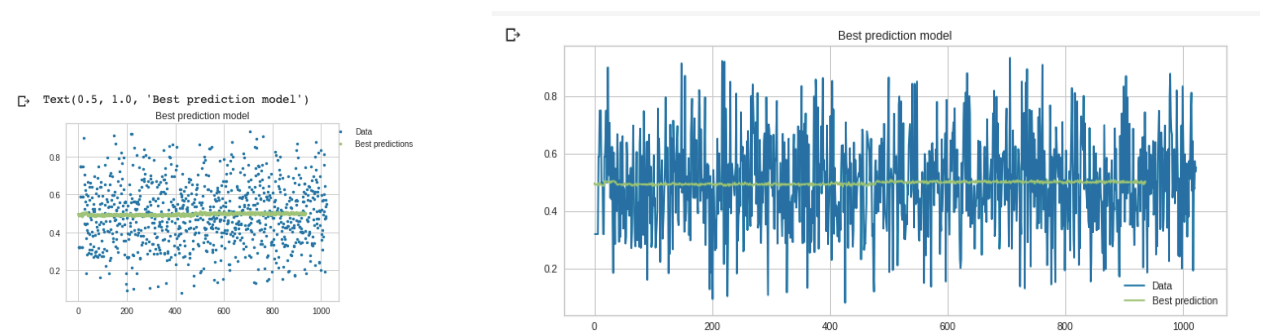


Рисунок 16 - Результаты настройки линейной регрессии.

Весь алгоритм работы повторим для признака «Прочность при растяжении».

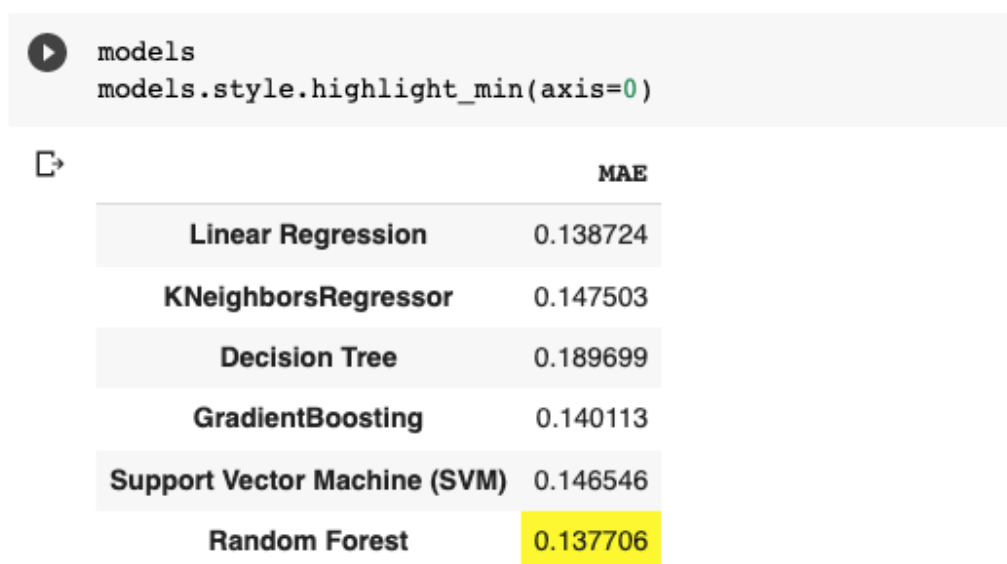
Ниже на рисунке 17 представлена отработка базовой модели для прочности при растяжении.

```
[ ] def mae(y_true, y_pred):  
    y_true, predictions = np.array(y_true), np.array(y_pred)  
    return np.mean(np.abs(y_true - predictions))  
  
baseline_guess = np.median(y2)  
  
print('The baseline guess is a score of %0.2f' % baseline_guess)  
print("Baseline Performance for test data: MAE = %0.4f" % mae(y2_test, baseline_guess))  
print('Accuracy = {:0.2f}%'.format(accuracy))
```

The baseline guess is a score of 0.51
Baseline Performance for test data: MAE = 0.1386
Accuracy = 86.17%.

Рисунок 17 - Базовая модель.

По результатам можно отметить, для второго признака модели работают чуть лучше в соответствии с рисунком 18.



The screenshot shows a Jupyter Notebook interface. The top part is a code cell with the following code:

```
models  
models.style.highlight_min(axis=0)
```

Below the code cell is a table with two columns: the model names and their corresponding MAE values. The table is styled with alternating light and dark gray rows. The 'Random Forest' row is highlighted in yellow.

	MAE
Linear Regression	0.138724
KNeighborsRegressor	0.147503
Decision Tree	0.189699
GradientBoosting	0.140113
Support Vector Machine (SVM)	0.146546
Random Forest	0.137706

Рисунок 18 - MAE моделей для плотности при растяжении.

Лучший показатель при отработке показала модель случайного леса, для него была проведена гиперпараметрическая оптимизация модели.

Лучший результат показала модель на основе RandomForestRegressor - для нее оптимизируем модель

```
param_grid = {
    'bootstrap': [True],
    'max_depth': [80, 90, 100, 110],
    'max_features': ['auto'],
    'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]
}

RF = RandomForestRegressor()

GSCV_RF = GridSearchCV(estimator = RF, param_grid = param_grid,
                       cv = 3, n_jobs = -1, verbose = 2)

GSCV_RF.fit(x2_train, y2_train)
GSCV_RF.best_params_

[ ] Fitting 3 folds for each of 40 candidates, totalling 120 fits
{'bootstrap': True,
 'max_depth': 80,
 'max_features': 'auto',
 'n_estimators': 1000}
```

Рисунок 19 - Регуляризация модели случайного леса.

При регуляризации модель чуть ухудшила показатель ошибки (числовой и графические результаты на рисунках 20 и 21), хотя случайный лес имеет преимущества перед линейными регрессиями – у него выше точность предсказания, меньше тенденции к переобучению и более широкий выбор гиперпараметров, которые можно настраивать.

```
[ ] model2=GSCV_RF.best_estimator_
    model2.fit(x2_train, y2_train)
    GSCV_RF_prediction2 = GSCV_RF.predict(x2_test)
    mae = np.mean(abs(GSCV_RF_prediction2 - y2_test))
    accuracy = (1 - mae)*100

    print('Model Performance')
    print('Random Forest test data MAE is = %0.6f' % mae)
    print('Accuracy = {:.2f}%'.format(accuracy))

Model Performance
Random Forest test data MAE is = 0.138045
Accuracy = 86.20%.
```

Рисунок 20 - Результаты настроенной модели случайного леса

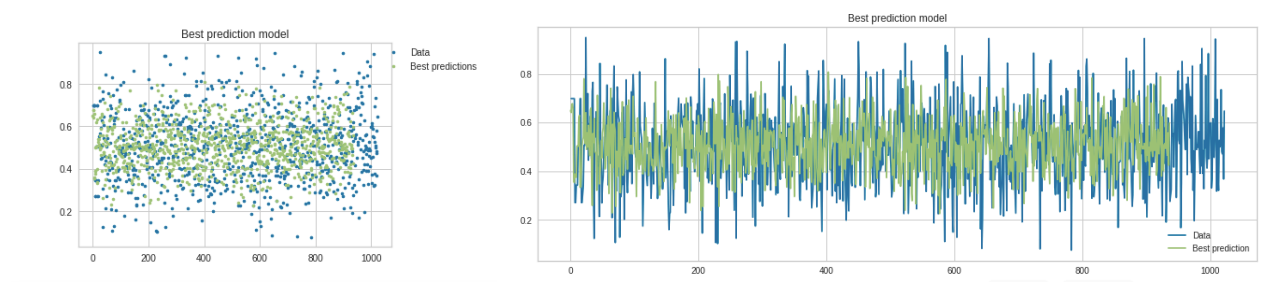


Рисунок 21 - Визуализация работы модели.

Несмотря на дополнительные возможности модели случайного леса, все использованные в работе модели не справились со своей задачей.

2.3. Оценка точности работы моделей.

Сравнительная оценка точности работы моделей опять же лучше себя отработала для модели случайного леса, что опять же показывает его преимущество перед линейными регрессиями.

Оценка точности модели на тренировочном и тестовом датасете

```
[ ] BM_1=GSCV.best_estimator_
    BM_1.fit(x1_train, y1_train)
    prediction_train = BM_1.predict(x1_train)
    prediction_test = BM_1.predict(x1_test)
    mae_train = np.mean(abs(prediction_train - y1_train))
    accuracy_train = (1-mae_train)*100
    mae_test = np.mean(abs(prediction_test - y1_test))
    accuracy_test = (1-mae_test)*100
    print('Ridge trian data MAE is = %0.6f, ' % mae_train,
          'Accuracy train data = {:0.2f}%'.format(accuracy_train))
    print('Ridge test data MAE is = %0.6f, ' % mae_test,
          'Accuracy test data = {:0.2f}%'.format(accuracy_test))
```

Ridge trian data MAE is = 0.128319, Accuracy train data = 87.17%.
Ridge test data MAE is = 0.138294, Accuracy test data = 86.17%.

```
[ ] BM_2=GSCV_RF.best_estimator_
    BM_2.fit(x2_train, y2_train)
    prediction_train = BM_2.predict(x2_train)
    prediction_test = BM_2.predict(x2_test)
    mae_train = np.mean(abs(prediction_train - y2_train))
    accuracy_train = (1-mae_train)*100
    mae_test = np.mean(abs(prediction_test - y2_test))
    accuracy_test = (1-mae_test)*100
    print('Random Forest train data MAE is = %0.6f, ' % mae_train,
          'Accuracy train data = {:0.2f}%'.format(accuracy_train))
    print('Random Forest test data MAE is = %0.6f, ' % mae_test,
          'Accuracy test data = {:0.2f}%'.format(accuracy_test))
```

Random Forest train data MAE is = 0.049103, Accuracy train data = 95.09%.
Random Forest test data MAE is = 0.138312, Accuracy test data = 86.17%.

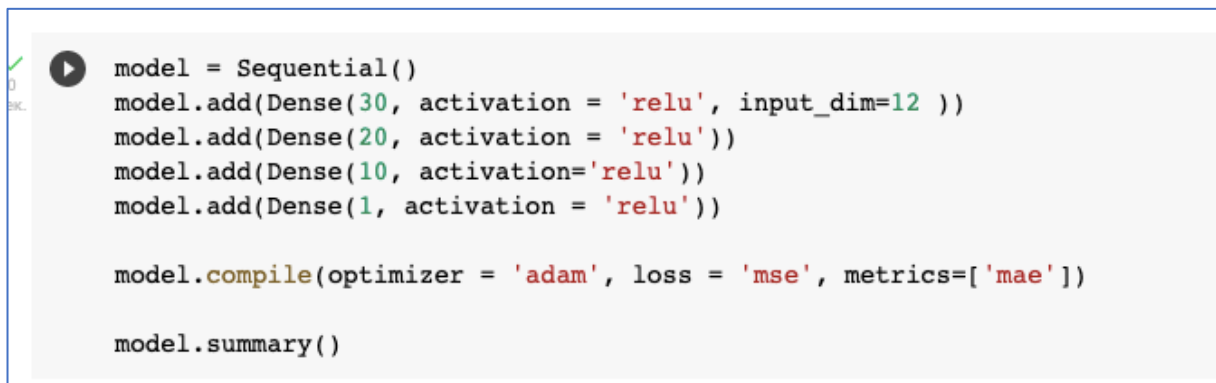
Рисунок 22 - Оценка точности моделей.

2.4. Написание нейронной сети, которая будет рекомендовать соотношение матрица-наполнитель.

Для рекомендации соотношения «матрица-наполнитель» была разработана простая модель глубокого обучения.

Архитектура нейронной сети может быть описана следующим образом.

Модель состоит из четырех скрытых уровней. Первый содержит 30 нейронов, что немного превышает объем входных данных. Последующие скрытые уровни – они содержат 20, 10 и 1 нейрона. Снижение числа нейронов на каждом уровне сжимает информацию, которую сеть обработала на предыдущих уровнях. В качестве активационной функции была выбрана Relu, так как она является хорошим аппроксиматором.



```
model = Sequential()  
model.add(Dense(30, activation = 'relu', input_dim=12 ))  
model.add(Dense(20, activation = 'relu'))  
model.add(Dense(10, activation='relu'))  
model.add(Dense(1, activation = 'relu'))  
  
model.compile(optimizer = 'adam', loss = 'mse', metrics=['mae'])  
  
model.summary()
```

Рисунок 23 - Архитектура нейронной сети

Перед обучением модели был настроен процесс помощью методом `compile()` с тремя аргументами:

- оптимизатора «adam», который является одним из самых эффективных алгоритмов оптимизации в обучении нейронных сетей. Он сочетает в себе идеи RMSProp и оптимизатора импульса;

- функция ошибки – `mse`: значение, которое модель пытается минимизировать;

- метрика `['mae']`.

После обучения модели для была определена средняя абсолютная ошибка: 0.13865895634424602, она оказалась чуть хуже рассчитанной для базового уровня, которая составила 0.135312.

На рисунках 24 и 25 представлена визуализация обучения модели и разброса предсказаний.

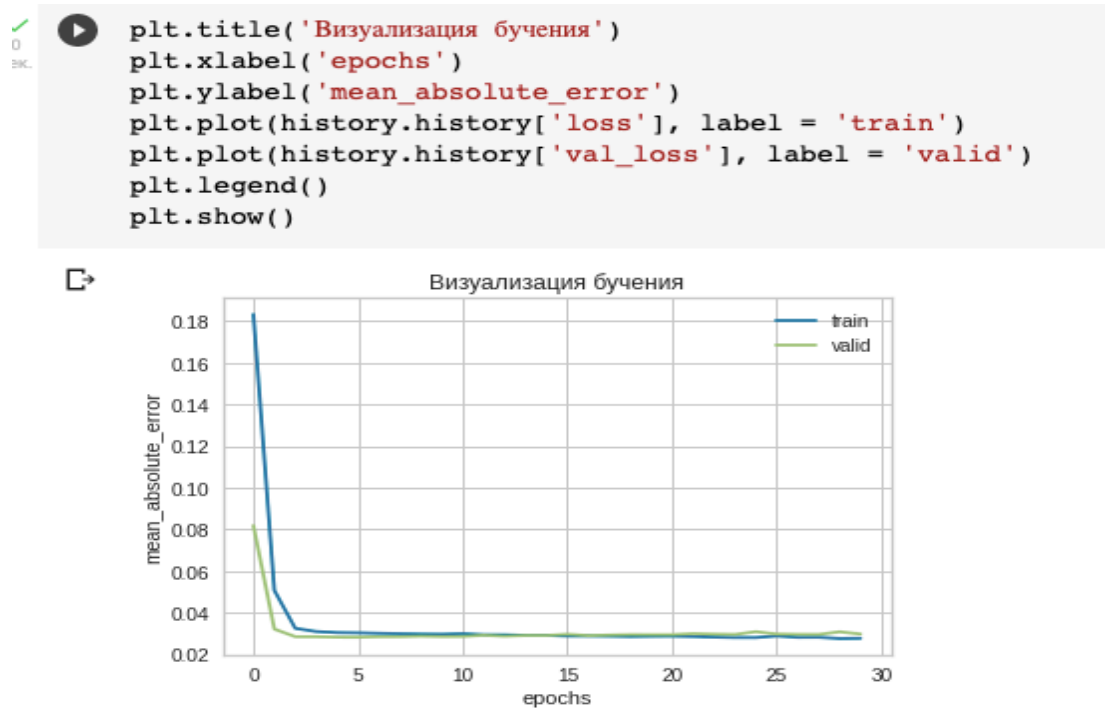


Рисунок 24 - Визуализация работы модели.

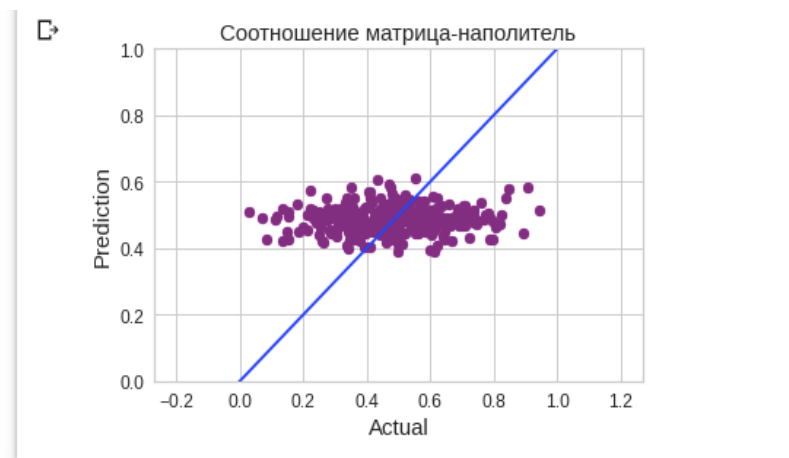


Рисунок 25 - Разброс предсказаний для соотношения матрица-наполнитель.

2.5 Создание удаленного репозитория.

Создана страница на GitHub: https://github.com/Albina-Agametova/VKR_Agametova_AS

В репозиторий загружены файлы: пояснительная записка, презентация, страница с кодом и создан файл ReadMe.

Заключение.

Удобство теоретических исследований состоит в формировании неограниченного количества гипотез, на основании которых в свою очередь строится модель. В исследуемой работе ни одна из моделей не показала желаемый результат. Возможно, для такого исследования нужен более многослойный подход, так как свойства композиционных материалов зависят от структуры и свойств матрицы, от агрегатного состояния второй фазы композиционного материала (твердый или нет), от дисперсности и силы взаимодействия фаз с матрицей.

Список используемой литературы.

1 ГОСТ Р 57970-2017. Национальный стандарт Российской Федерации. Композиты углеродные. Углеродные композиты, армированные углеродным волокном [Текст]. – Введ. 2018–06–01. – М. : Стандартиформ, 2018. – 20 с.

2 Субботина, С. А., Шлыкова, И. Д., Авдеева А. А., Одиноква, Г. В, Соколова, Н. В., Виды композитных материалов: стеклопластик, углепластик, базальтопластик [Текст] : Синергия наук. – 2017. – № 18. – 5 с.

3 Д.А. Иванов А.И., Ситников, С.Д Шляпин – Композиционные материалы: учебное пособие для вузов, 2019. 13 с.

4 А.А. Ларин, Способы оценки работоспособности изделий из композиционных материалов методом компьютерной томографии, Москва, 2013, 148 с.

5 А.А. Миронов. Машинное обучение часть I ст.9 – Режим доступа: <http://is.ifmo.ru/verification/machine-learning-mironov.pdf>. (дата обращения 12.06.2022)

6 Шитиков В.К., Мастицкий С.Э. (2017) Классификация, регрессия и другие алгоритмы Data Mining с использованием R. 351 с. – Режим доступа: <https://github.com/ranalytics/data-mining> (дата обращения 13.06.2022)

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

8 Билл Любанович. Простой Python. Современный стиль программирования. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O'Reilly»)

9 Язык программирования Python: - Режим доступа: <https://www.python.org/> (дата обращения 13.06.2022)

10 Библиотека Pandas – Режим доступа: <https://pandas.pydata.org/> (дата обращения 15.06.2022)

11 Библиотека Matplotlib – Режим доступа: [https:// matplotlib.org/](https://matplotlib.org/) (дата обращения 15.06.2022)

12 Библиотека Sklearn – Режим доступа: <https://scikit-learn.org/stable/> (дата обращения 15.06.2022)

13 Краткий обзор алгоритма машинного обучения Метод Опорных Векторов (SVM) – Режим доступа: <https://habr.com/ru/post/428503/> (дата обращения 15.06.2022)

14 Random Forest in Python with scikit-learn - Режим доступа: <https://www.datacareer.de/blog/random-forest-in-python-with-scikit-learn/> (дата обращения 15.06.2022)

15 Florian Muller. Hyperparameter Tuning a Random Forest Classifier using Grid Search in Python – Режим доступа: <https://www.relataly.com/hyperparameter-tuning-with-grid-search/2261/>

16 How to choose a machine learning model in Python? – Режим доступа: <https://www.codeastar.com/choose-machine-learning-models-python/> (дата обращения 14.06.2022)