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»

Тема: «Прогнозирование конечных свойств новых материалов (композиционных материалов)»

Слушатель Пономарева Мария Александровна

Москва 2025

Содержание

1. Аттестационное задание

Композиционные материалы — это материалы, состоящие из двух или более компонентов, нерастворимых друг с другом, с чётко обозначенной границей раздела и сильным взаимодействием по всей зоне контакта. Одним из компонентов композитных материалов является непрерывная фаза, он называется матрица, в которой нерастворимые материалы помещаются в другую природу, называемую арматурой или наполнителем.

Внедрение композиционных материалов обусловлено стремлением использовать их преимущества по сравнению с традиционно используемыми металлами и сплавами. Примеры композита – железобетон (сочетание стали арматуры и камня бетона), древесноволокнистая плита ДВП (сочетание древесной основы – щепы и полимерного связующего).

В настоящее время активно развивается использование композитных материалов на основе базальта.

Базальтопластик - современный композитный материал на основе базальтовых волокон и органического связующего вещества. В настоящее время базальтопластик успешно конкурирует с металлическими изделиями, превосходя их по коррозионной, щелочной, кислотоустойчивости и некоторым другим свойствам. Целью данной работы является прогнозирование конечных свойств новых материалов на основе базальтопластика (композиционных материалов).

Расширение разнообразия материалов, используемых при проектировании нового композиционного материала, увеличивает необходимость определения свойств нового композита при минимальных финансовых затратах. Для решения этой проблемы обычно используются два способа: физические тесты образцов материалов или оценка свойств, в том числе на основе физико-математических моделей. Традиционно разработка

композитных материалов является долгосрочным процессом, так как из свойств отдельных компонентов невозможно рассчитать конечные свойства композита. Для достижения определенных характеристик требуется большое количество различных комбинированных тестов, что делает насущной задачу прогнозирования успешного решения, снижающего затраты на разработку новых материалов и затраты на рабочую силу. Суть прогнозирования заключается в моделировании репрезентативного элемента композитного объёма на основе данных о свойствах входящих компонентов (связующего и армирующего компонента).

Для выполнения данной работы были предоставлены таблицы со свойствами композиционных материалов. На выходе необходимо спрогнозировать конечные свойства получаемых композиционных материалов.

Работа проходила в несколько основных этапов:

- первичная обработка данных,

- разведочный анализ,

- построение регрессионных моделей с помощью Sklearn,

- построение модели с помощью нейросети с помощью keras,

- создание веб-приложения для обученных моделей с помощью Flask,

2. Постановка задачи. Загрузка данных.

Цель работы заключается в разработке моделей для прогнозирования конечных свойств композитов: модуля упругости при растяжении, прочности при растяжении и модели для рекомендации соотношения «матрица-наполнитель».

Задачи: обучить несколько моделей регрессии различными методами, выбрать наилучшие, подготовить модель к дальнейшему использованию.

В качестве исходных данных для выполнения работы был предоставлен архив с двумя файлами (датасеты со свойствами композитов):

df_xbp.xlsx (данные о параметрах базальтопластика, состоящий из 1024 строки и 11 столбцов);

и df_xnup.xlsx (данные по нашивкам углепластика, состоящий из 1041 строки и 4 столбцов).

Согласно заданию датасеты были объединены, тип объединения INNER. В дальнейшей работе использовалась объединённая таблица размерностью 1023 x 13 (переменная df_all)

```
#Выведем первые и последние стоки dataseta
df_all
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	
0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000	
1	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	
2	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385	
3	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	
4	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385	
...	...	...	...	...	...	...	
1018	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576	
1019	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401	
1020	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047	
1021	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840	
1022	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708	

1023 rows × 13 columns

Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
210.000000	70.000000	3000.000000	220.000000	0	4.000000	57.000000
210.000000	70.000000	3000.000000	220.000000	0	4.000000	60.000000
210.000000	70.000000	3000.000000	220.000000	0	4.000000	70.000000
210.000000	70.000000	3000.000000	220.000000	0	5.000000	47.000000
210.000000	70.000000	3000.000000	220.000000	0	5.000000	57.000000
...	...	...	...	...	...	...
209.198700	73.090961	2387.292495	125.007669	90	9.076380	47.019770
350.660830	72.920827	2360.392784	117.730099	90	10.565614	53.750790
740.142791	74.734344	2662.906040	236.606764	90	4.161154	67.629684
641.468152	74.042708	2071.715856	197.126067	90	6.313201	58.261074
758.747882	74.309704	2856.328932	194.754342	90	6.078902	77.434468

Итоговый датасет состоит из 13 столбцов с различными характеристиками, включая как исходные параметры компонентов, так и конечные свойства композитов.

3. Разведочный анализ данных

Разведочный анализ данных - очень важный этап для любого проекта, связанного с анализом данных. Он помогает получить понимание данных и определить, какие методы и алгоритмы будут применяться для моделирования и анализа данных.

Разведочный анализ данных - это оценка качества исходных данных (наличие пропусков, выбросов, дубликатов), получение первоначальных представлений о характерах распределений переменных, выявление характера взаимосвязи между переменными.

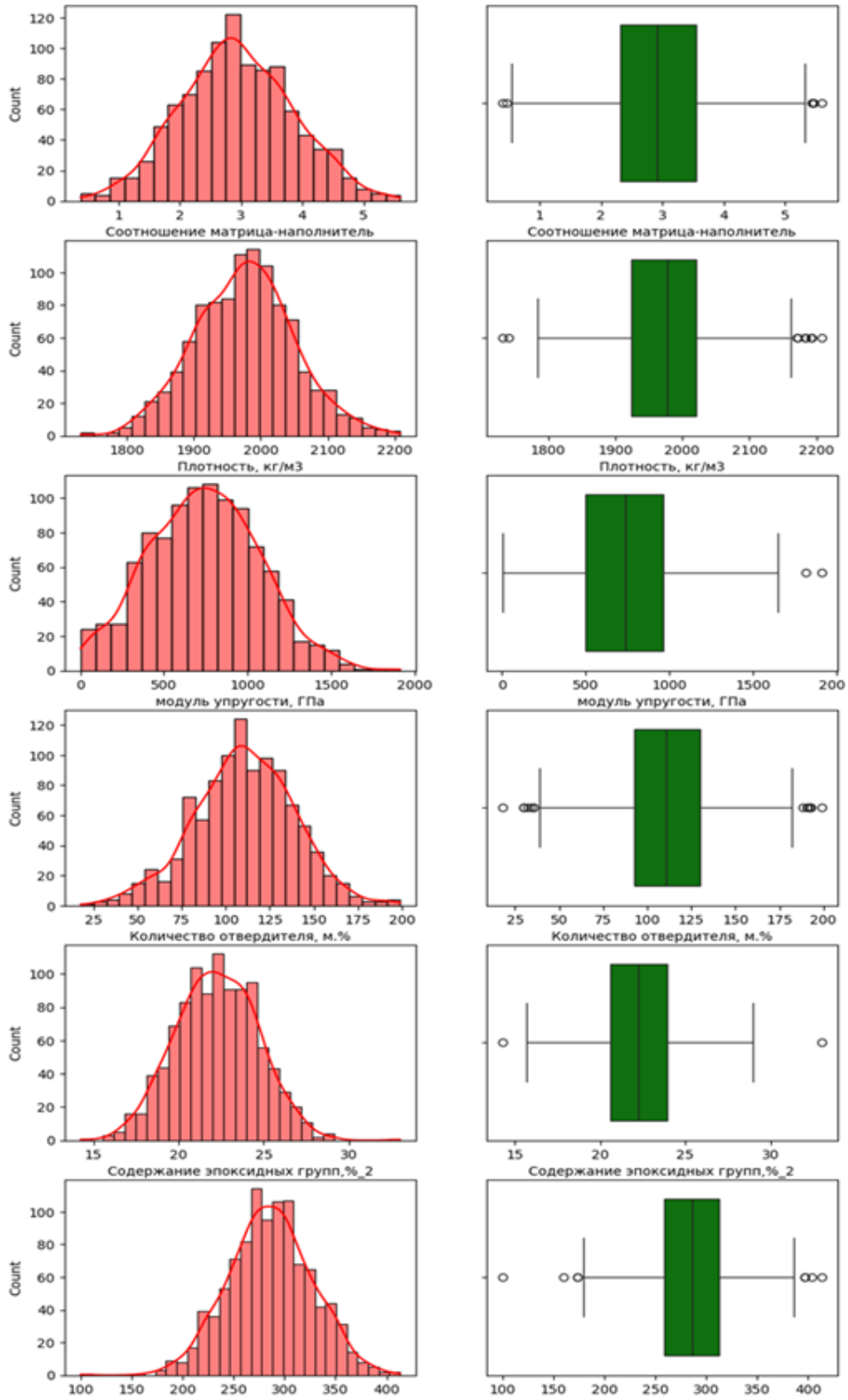
Для первоначального анализа данных используем следующие команды Python:

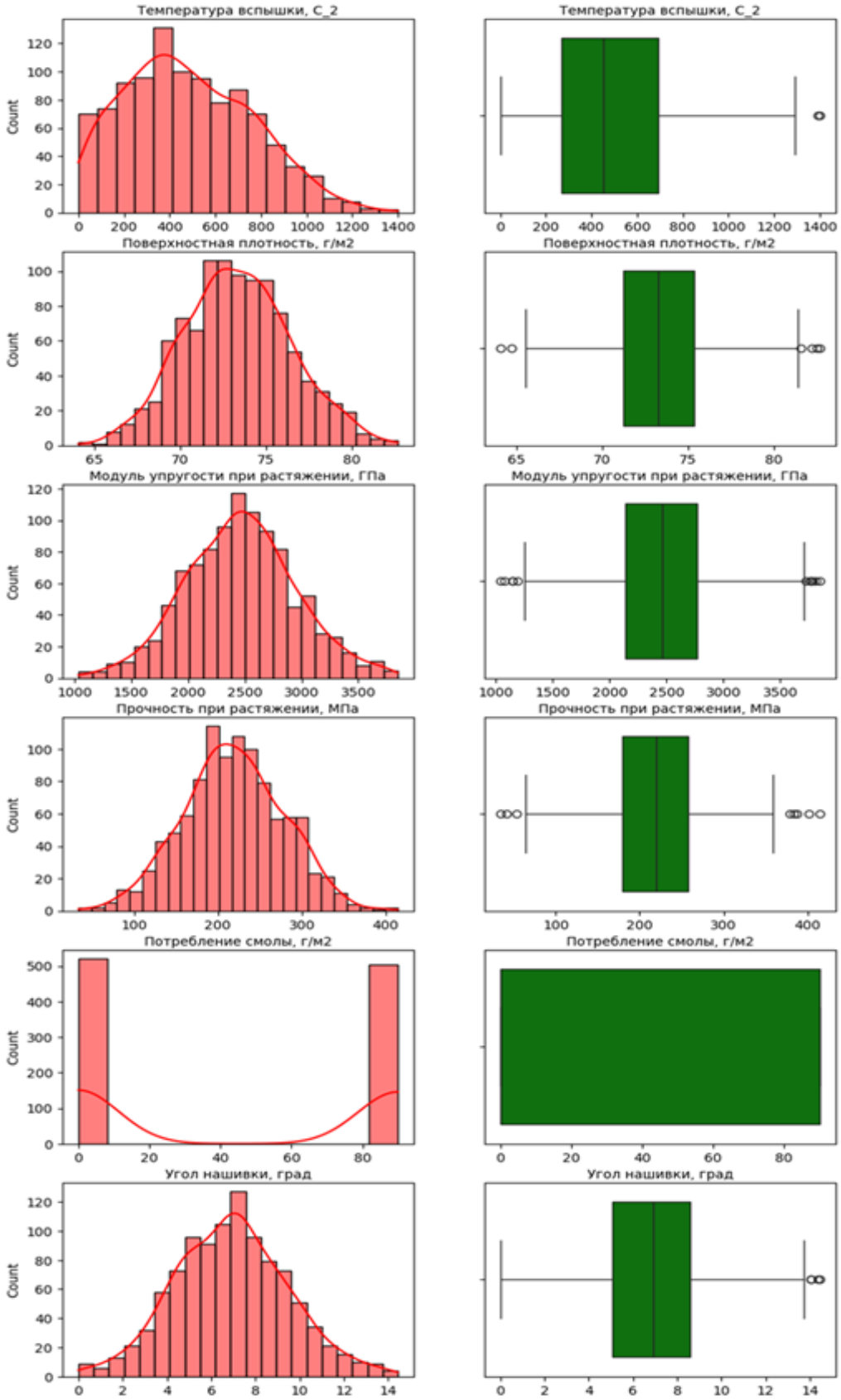
df.info() - проверяем наличие пустых ячеек и тип данных;

```
#Выведем информацию о нем
df_all.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1023 entries, 0 to 1022
Data columns (total 13 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   Соотношение матрица-наполнитель       1023 non-null   float64
 1   Плотность, кг/м3                      1023 non-null   float64
 2   модуль упругости, ГПа                 1023 non-null   float64
 3   Количество отвердителя, м.%           1023 non-null   float64
 4   Содержание эпоксидных групп,%_2       1023 non-null   float64
 5   Температура вспышки, С_2              1023 non-null   float64
 6   Поверхностная плотность, г/м2         1023 non-null   float64
 7   Модуль упругости при растяжении, ГПа  1023 non-null   float64
 8   Прочность при растяжении, МПа         1023 non-null   float64
 9   Потребление смолы, г/м2               1023 non-null   float64
 10  Угол нашивки, град                    1023 non-null   int64  
 11  Шаг нашивки                           1023 non-null   float64
 12  Плотность нашивки                     1023 non-null   float64
dtypes: float64(12), int64(1)
memory usage: 111.9 KB
```

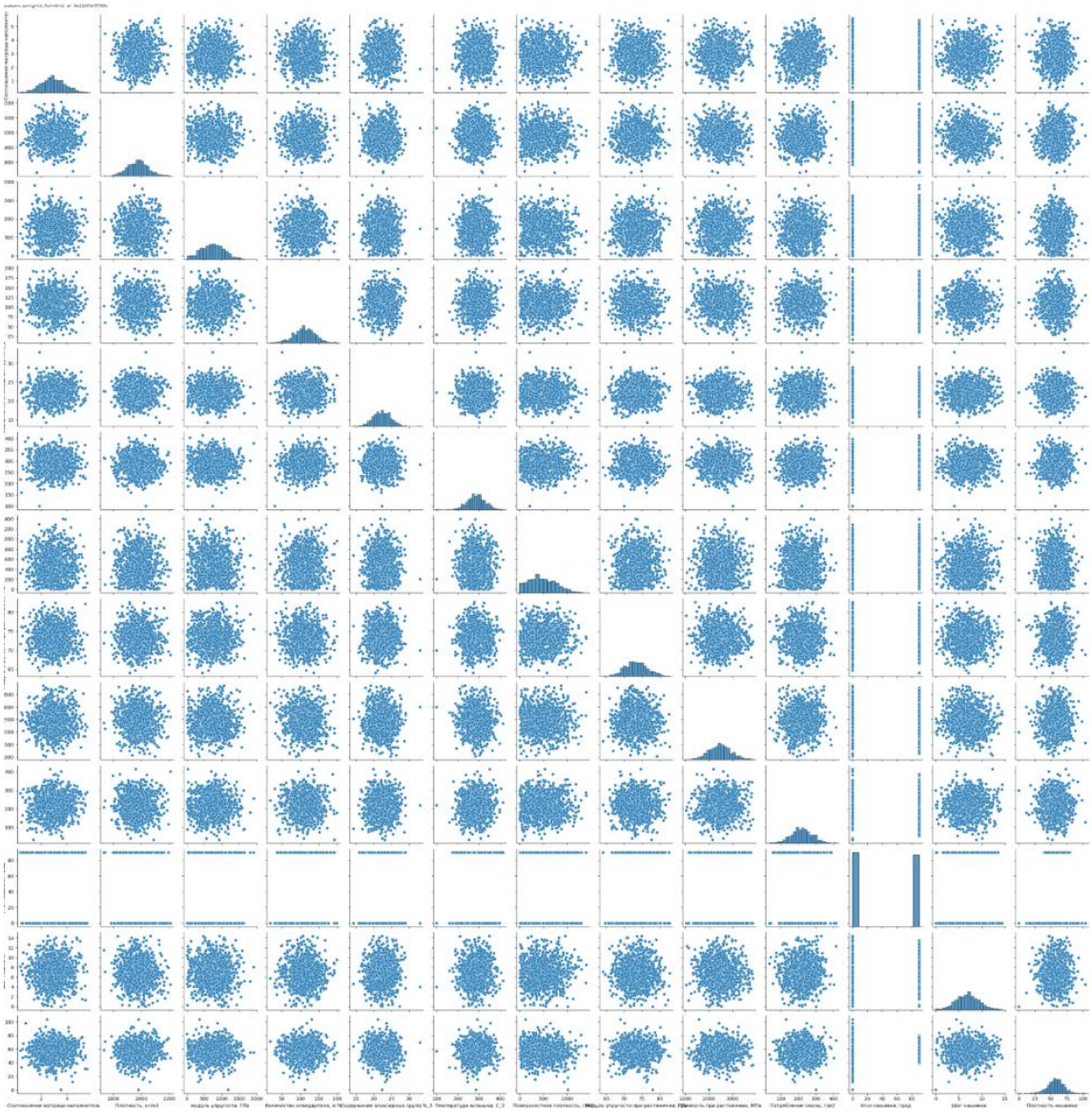
Гистограммы распределения переменных и диаграммы «ящик с усами». По ним видно, что все признаки, кроме «Угол нашивки», имеют нормальное распределение и принимают неотрицательные значения. «Угол нашивки» принимает значения: 0, 90.





df.duplicated().sum() - проверяем наличие дубликатов в данных;

sns.pairplot(df) - обзор гистограмм и графиков рассеяния



По графикам рассеяния мы видим, что некоторые точки отстоят далеко от общего облака. Так визуально выглядят выбросы — аномальные, некорректные значения данных, выходящие за пределы допустимых значений признака.

Существуют такие методы выявления выбросов для признаков с нормальным распределением:

− метод 3-х сигм;

− метод межквартильных расстояний.

Применив эти методы на нашем датасете было найдено:

− методом 3-х сигм — 24 выброса;

− методом межквартильных расстояний — 93 выброса

df.describe() - смотрим описательную статистику (средние, медианы, минимальные, максимальные значения…)

```
#Выведем данные описательной статистики при помощи функции describe
df_all.describe()
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа
count	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000
mean	2.930366	1975.734888	739.923233	110.570769	22.244390	285.882151	482.731833	73.328571
std	0.913222	73.729231	330.231581	28.295911	2.406301	40.943260	281.314690	3.118983
min	0.389403	1731.764635	2.436909	17.740275	14.254985	100.000000	0.603740	64.054061
25%	2.317887	1924.155467	500.047452	92.443497	20.608034	259.066528	266.816645	71.245018
50%	2.906878	1977.621657	739.664328	110.564840	22.230744	285.896812	451.864365	73.268805
75%	3.552660	2021.374375	961.812526	129.730366	23.961934	313.002106	693.225017	75.356612
max	5.591742	2207.773481	1911.536477	198.953207	33.000000	413.273418	1399.542362	82.682051

	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
count	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000
mean	2466.922843	218.423144	44.252199	6.899222	57.153929
std	485.628006	59.735931	45.015793	2.563467	12.350969
min	1036.856605	33.803026	0.000000	0.000000	0.000000
25%	2135.850448	179.627520	0.000000	5.080033	49.799212
50%	2459.524526	219.198882	0.000000	6.916144	57.341920
75%	2767.193119	257.481724	90.000000	8.586293	64.944961
max	3848.436732	414.590628	90.000000	14.440522	103.988901

sns.heatmap(df.corr() - строим матрицу корреляции

```
# Построим матрицу корреляции
corr = df_all.corr()
fig, ax = plt.subplots(figsize=(10, 7))
sns.heatmap(corr, vmin=-1, vmax=1, annot=True, fmt='.2f', cmap='BrBG')
plt.show()
```

Гистограммы используются для изучения распределений частот значений переменных. Видим, что почти все переменные имеют близкое к нормальному распределение.

Видим очень слабую корреляцию между переменными. Корреляция между всеми параметрами близка к 0, корреляционные связи между переменными не наблюдаются.

Затем более детально изучаем распределение и описательную статистику каждой переменной, а также корреляцию переменных между собой.

Далее смотрим количество уникальных значений по каждому параметру, строим графики «ящики с усами», которые отражают характер распределения каждого параметра. Используются следующие команды:

df.nunique() - подсчёт уникальных значений каждой переменной;

```
#Поиск уникальных значений
df_all.nunique()
```

```
Соотношение матрица-наполнитель         1014
Плотность, кг/м3                        1013
модуль упругости, ГПа                   1020
Количество отвердителя, м.%             1005
Содержание эпоксидных групп,%_2         1004
Температура вспышки, С_2                1003
Поверхностная плотность, г/м2           1004
Модуль упругости при растяжении, ГПа    1004
Прочность при растяжении, МПа           1004
Потребление смолы, г/м2                 1003
Угол нашивки, град                         2
Шаг нашивки                              989
Плотность нашивки                        988
dtype: int64
```

sns.boxplot() - отрисовка «ящиков с усами»

Общие первичные выводы по результатам разведочного анализа:

- распределение почти всех признаков близко к нормальному, кроме 2 параметров:

- "Угол нашивки" (здесь только два значения),

- "Поверхностная плотность" (ассиметрия);

- корреляции нет ни по одному пересечению переменных.

На этапе обработки данных и разведочного анализа применялись стандартные методы Python и методы библиотек Pandas, Numpy, Seaborn, Matplotlib, Sklearn. Эти методы позволяют вывести статистические показатели, выявить данные с пустыми ячейками, количество уникальных

значений и другое. Также с помощью этих методов можно визуализировать данные в виде различных графиков.

4. Ход решения задачи

Ход решения каждой из задач и построения оптимальной модели будет следующим:

1. разделить данные на тренировочную и тестовую выборки. В задании ука зано, что на тестирование оставить 30% данных;
2. выполнить препроцессинг, то есть подготовку исходных данных;
3. взять несколько моделей с гиперпараметрами по умолчанию, а некоторые модели взять с гиперпараметрами, подобранными с помощью GridSearchCV, и используя перекрестную проверку, посмотреть их метрики на тренировочной выборке;
4. сравнить метрики моделей после подбора гиперпараметров и выбрать лучшую;
5. получить предсказания моделей на тестовой выборке, сделать выводы;

Препроцессинг

Цель препроцессинга или предварительной обработки данных — обеспечить корректную работу моделей.

Его необходимо выполнять после разделения на тренировочную и тестовую выборку, как будто мы не знаем параметров тестовой выборки (минимум, максимум, матожидание, стандартное отклонение).

Препроцессинг для категориальных и количественных признаков выполняется по-разному.

У признака 'Угол нашивки, град' два значения: 0 и 90. Т.к. у этого признака всего два значения, то при нормализации они примут значения 0 и 1, поэтому можно не использовать LabelEncoder или OrdinalEncoder.

Количественных признаков у нас большинство. Проблема вещественных признаков в том, что их значения лежат в разных диапазонах, в разных масштабах. Это видно в таблице 2. Т.к. наши данные распределены нормально, то можно использовать любой метод преобразований:

− нормализацию — приведение в диапазон от 0 до 1 с помощью MinMaxScaler;
− стандартизацию — приведение к матожиданию 0, стандартному отклонению 1 с помощью StandartScaler.

Использую нормализацию MinMaxScaler.

Выходные переменные никак не изменяем.

6. Методы регрессионного анализа

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

Имеющиеся данные можно разделить на свойства исходных компонентов и на свойства получаемых композитов. В качестве входных параметров логично использовать свойства исходных компонентов и их соотношение. Но по имеющимся данным однозначно непонятно какие параметры относятся к исходным, а какие к полученным. Дополнительной информации по предметной области не предоставлено.

Далее в работе были использованы методы регрессионного анализа:

Линейная регрессия

Простая линейная регрессия имеет место, если рассматривается зависимость между одной входной и одной выходной переменными. Для этого определяется уравнение регрессии (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).

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

(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 бибилиотеке sklearn.linear_model.LinearRegression.

1.2.2 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 ме тода 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, евклидово 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, по павших 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 некото рое 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 классифи кации — 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 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 промыш- ленных 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.

Поиск гиперпараметров по сетке

Поиск гиперпараметров по сетке реализует класс GridSearchCV из sklearn. Он получает модель и набор гиперпараметров, поочередно передает их в модель, выполняет обучение и определяет лучшие комбинации гиперпараметров.

Метрики качества моделей

Существует множество различных метрик качества, применимых для регрессии. В этой работе я использую:

1. 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;
2. MAE (Mean Absolute Error) - средняя абсолютная ошибка так же принимает значениях в тех же единицах, что и целевая переменная;
3. MSE – показатель среднеквадратичной ошибки (Mean Squared Error).

MAE, MSE, RMSE эти метрики надо минимизировать.

Результаты применения данных моделей для задач определения «Модуля упругости»:

Linear Regression Results Train:
Ttain score: 0.02
Linear Regression Results:
lr_MAE: 396
lr_MAPE: 0.18
lr_MSE: 240591.32
lr_RMSE: 490.50
Test score: 0.02

K Neighbors Regressor Results Train:
Train score: 0.24
K Neighbors Regressor Results:
KNN_MAE: 3

KNN_MAPE: 0.04
KNN_MSE: 12.45
KNN_RMSE: 3.53
Test score: -0.30

```
Gradient Boosting Regressor Results Train:
Train score: 0.49
Gradient Boosting Regressor Results:
GBR_MAE:  3
GBR_MAPE: 0.04
GBR_MSE: 10.34
GBR_RMSE: 3.22
Test score: -0.08

Support Vector Regression Results Train:
Train score: 0.87
Support Vector Regression Results:
SVR_MAE: 3
SVR_MAPE: 0.04
SVR_MSE: 15.88
SVR_RMSE: 3.98
Test score: -0.66


Decision Tree Regressor Results Train:
Train score: 1.00
Decision Tree Regressor Results:
DTR_MAE:  4
DTR_MSE: 20.24
DTR_RMSE: 4.50
DTR_MAPE: 0.05
Test score: -1.12

Random Forest Regressor Results Train:
Train score: 0.40
Random Forest Regressor Results:
RF_MAE:  3
RF_MAPE: 0.03
RF_MSE: 10.19
RF_RMSE: 3.19
Test score: -0.07
```

Результаты применения данных моделей для задач определения «Прочности при растяжении»:

Random Forest Regressor Results Train:
Train score: -24.93
Random Forest Regressor Results:
RF_MAE: 401
RF_MAPE: 0.18
RF_MSE: 250450.54
RF_RMSE: 500.45
Test score: -0.02

Linear Regression Results Train:
Ttain score: 0.02
Linear Regression Results:

lr_MAE: 396
lr_MAPE: 0.18
lr_MSE: 240591.32
lr_RMSE: 490.50
Test score: 0.02

```
K Neighbors Regressor  Results Train:
Train score: -24.93
K Neighbors Regressor  Results:
KNN_MAE:  424
KNN_MAPE: 0.19
KNN_MSE: 284049.93
KNN_RMSE: 532.96
Test score: -23.08
```

```
Support Vector Regression Results Train:
Train score: 0.30
Support Vector Regression Results:
SVR_MAE: 405
SVR_MAPE: 0.18
SVR_MSE: 257808.70
SVR_RMSE: 507.75
Test score: -0.05
```

```
Gradient Boosting Regressor Results Train:
Train score: 0.49
Gradient Boosting Regressor Results:
GBR_MAE:  412
GBR_MAPE: 0.18
GBR_MSE: 261216.18
GBR_RMSE: 511.09
Test score: -0.07
```

```
Decision Tree Regressor Results Train:
Train score: -24.93
Decision Tree Regressor Results:
DTR_MAE:  562
DTR_MSE: 500285.68
DTR_RMSE: 707.31
DTR_MAPE: 0.25
Test score: -1.05
```

Ни одна из выбранных мной моделей не соответствует данным.

Результат исследования отрицательный. Не удалось получить модели, которая могла бы оказать помощь в принятии решений специалисту предметной области.

7. **Разработка нейронной сети для прогнозирования соотношения матрица-наполнитель**

По заданию для соотношения матрица-наполнитель необходимо построить нейросеть.

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.

Строим нейронную сеть с помощью класса MLPRegressor следующей архитектуры:

− слоев: 8;

− нейронов на каждом слое: 24;

− активационная функция: relu;

− оптимизатор: adam;

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

− ранняя остановка, если метрики на валидационной выборке не улучшаются;

− количество итераций: 4000.

Нейросеть обучилась за 826 мс и 105 итерации. График обучения приведен на рисунке 17.

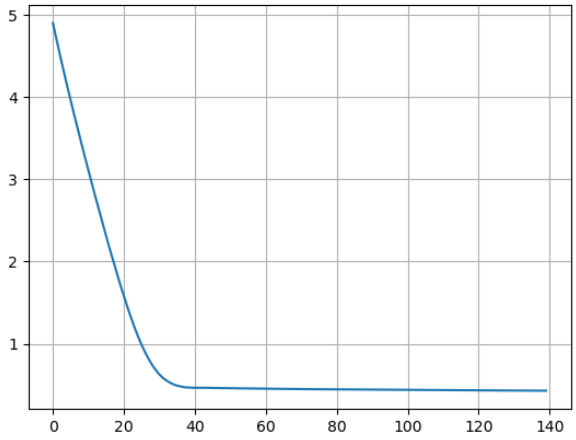


График ошибки при обучении MLPRegressor

Итоги работы модели:

MAE3: 0.7994235690741177
MSE3: 0.974118032555233
RMSE3: 0.9869741802880322

График тестовых/прогнозных значений для данной модели:

Строим другую нейросеть

Строю нейронную сеть с помощью класса keras.Sequential со следующими параметрами:

− входной слой для 12 признаков;

− пакетная нормализация BatchNormalization

− оптимизатор: Adam;

− loss-функция: MeanAbsoluteError.

− Архитектура нейросети:

Model: "sequential_16"

Layer (type)	Output Shape	Param #
dense_64 (Dense)	(None, 16)	208
batch_normalization_16 (BatchNormalization)	(None, 16)	64
dense_65 (Dense)	(None, 8)	136
dropout_11 (Dropout)	(None, 8)	0
dense_66 (Dense)	(None, 8)	72
dense_67 (Dense)	(None, 8)	72
dense_68 (Dense)	(None, 1)	9

Total params: 561 (2.19 KB)
Trainable params: 529 (2.07 KB)
Non-trainable params: 32 (128.00 B)
CPU times: total: 0 ns
Wall time: 0 ns
Epoch 1/25

График обучения:

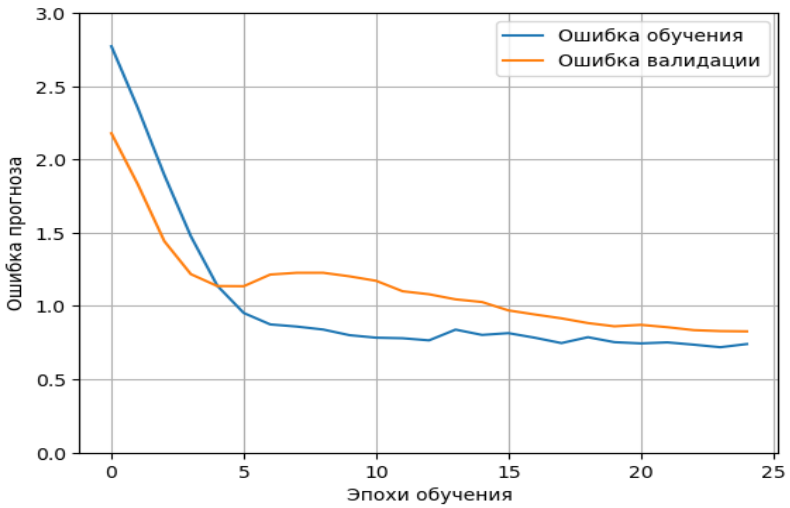


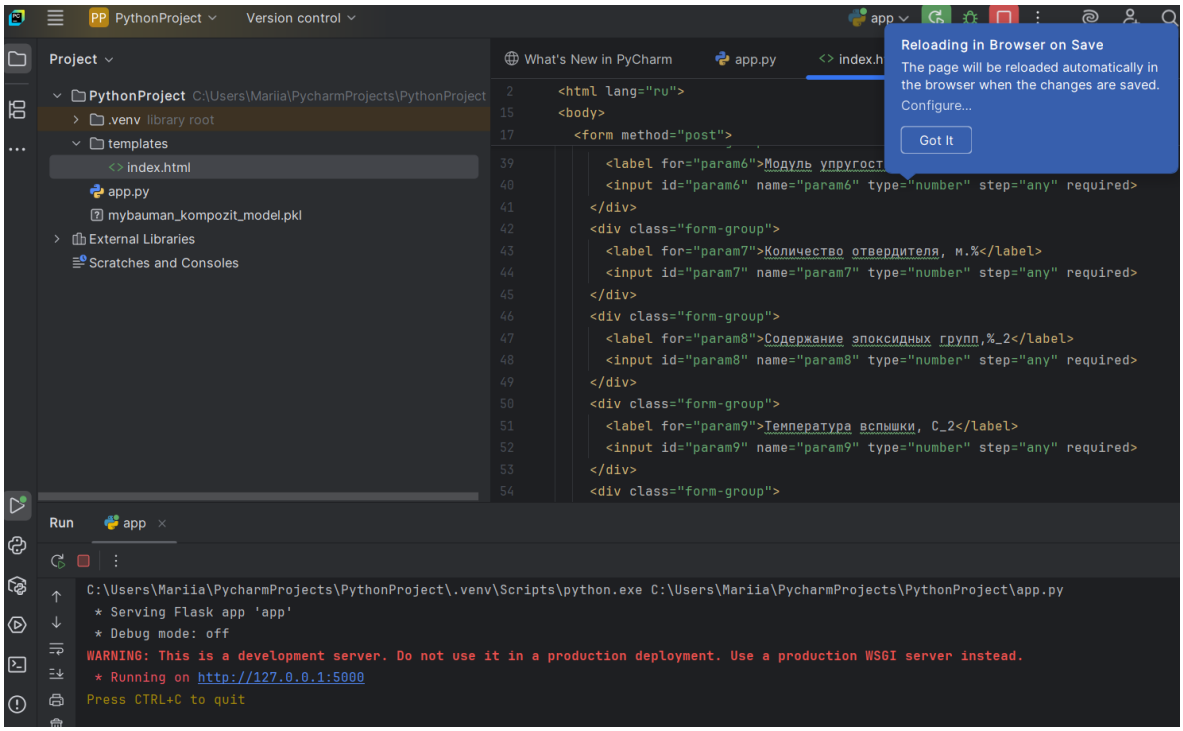
График тестовых/прогнозных значений для данной модели:

У нейросети показатели для тестовой выборки сильнее отличаются в худшую сторону от показателей тренировочной. Это говорит о том, что она не нашла закономерностей. Возможно, требуется более тщательное, грамотное изучение данных, построение другой архитектуры нейронной сети, чтобы получить лучший результат. Но сейчас задача далека от решения и модель не имеет пользы в реальных условиях.

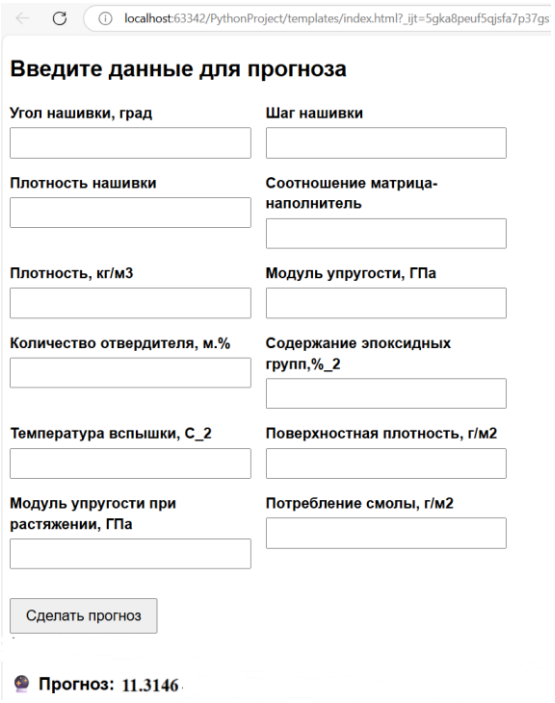
8. **Разработка приложения**

На основе сохраненной модели создадим приложение.

Для создания приложения была использована среда разработки PyCharm.



На основе введенных данных рассчитана Прочность при растяжении:



Заключение

В ходе выполнения данной работы мы рассмотрели большую часть операций и задач, которые приходится выполнять специалисту по работе с данными.

К сожалению, с практической стороны результаты работы оказались неудовлетворительными, и не могут быть применены на практике.

На основе проведенной работы можно сделать следующие предположения:

- исходные данные представлены не полностью;

- исходные данные некорректны, содержат ошибки;

- в данных не учтен какой-то важный, решающий фактор, или несколько факторов.

При дальнейшей работе с этими данными можно дать следующие рекомендации.

В части исходных данных можно рекомендовать:

- необходима консультация со специалистом в предметной области;

- необходимо получить дополнительную информацию по исходным данным;

- разобраться какие параметры – исходные свойства компонентов, а какие – свойства полученных композитов;

В части анализа данных можно рекомендовать провести дополнительные эксперименты (которые могут занять значительное время):

- применить другие модели нейронных сетей;

- применить преобразование исходных данных (логарифмирование, возведение в степень, «взвешивание» и др.);

- изучить вопрос разделения данных на отдельные кластеры.

Список используемых источников

1. Andre Ye. 5 алгоритмов регрессии в машинном обучении, о которых вам следует знать: – Режим доступа: https://habr.com/ru/company/vk/blog/513842/ (дата обращения: 26.03.2023).
2. Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru. 2020. - 412 с.: ил.
3. Абросимов Н.А.: Методика построения разрешающей системы уравнений динамического деформирования композитных элементов конструкций (Учебно-методическое пособие), ННГУ, 2010
4. Сара Бослаф. Статистика для всех. O'REILLY, 2015
5. Гафаров, Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учеб. пособие. – Казань: Издательство Казанского университета, 2018. – 121 с.
6. Грас Д. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.
7. Документация по библиотеке matplotlib: – Режим доступа: https://matplotlib.org/stable/users/index.html (дата обращения: 26.03.2023)
8. Документация по библиотеке numpy: – Режим доступа: https://numpy.org/doc/1.22/ (дата обращения: 26.03.2023).
9. Документация по библиотеке pandas: – Режим доступа: https://pandas.pydata.org/docs/user_guide (дата обращения: 26.03.2023).
10. Документация по библиотеке scikit-learn: – Режим доступа: https://scikit-learn.org/stable/user_guide.html. (дата обращения: 26.03.2023).
11. Документация по библиотеке seaborn: – Режим доступа: https://seaborn.pydata.org/tutorial.html. (дата обращения: 26.03.2023).
12. Документация по библиотеке Tensorflow: – Режим доступа: https://www.tensorflow.org/overview (дата обращения: 26.03.2023).

13. Документация по языку программирования python: – Режим доступа: https://docs.python.org/3.8/index.html. (дата обращения: 26.03.2023).
14. Иванов Д.А., Ситников А.И., Шляпин С.Д – Композиционные материалы: учебное пособие для вузов, 2019. 13 с.
15. Краткий обзор алгоритма машинного обучения Метод Опорных Векторов (SVM) – Режим доступа: https://habr.com/ru/post/428503/ (дата обращения 26.03.2023)
16. Ларин А. А., Способы оценки работоспособности изделий из композиционных материалов методом компьютерной томографии, Москва, 2013, 148 с.
17. Материалы конференции: V Всероссийская научно-техническая конференция «Полимерные композиционные материалы и производственные технологии нового поколения», 19 ноября 2021 г.
18. Миронов А.А. Машинное обучение часть I ст.9 – Режим доступа: http://is.ifmo.ru/verification/machine-learning-mironov.pdf. (дата обращения 26.03.2023)
19. Роббинс, Дженнифер. HTML5: карманный справочник, 5-е издание.: Пер. с англ. - М.: ООО «И.Д. Вильямс»: 2015. - 192 с.: ил.
20. Руководство по быстрому старту в flask: – Режим доступа: https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html. (дата обращения: 26.03.2023)
21. Русскоязычная документация Keras: – Режим доступа: https://ru-keras.com/. (дата обращения: 26.03.2023).
22. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.
23. Справочник по композиционным материалам: в 2 - х кн. Кн. 2 / Под ред. Дж. Любина; Пер. с англ. Ф. Б. Геллера, М. М. Гельмонта; Под ред. Б. Э. Геллера - М.: Машиностроение, 1988. - 488 с. : ил.
24. Учебник по машинному обучению. Академия Яндекс. Режим доступа: https://academy.yandex.ru/handbook/ml (дата обращения: 16.03.2023).