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 Мукатова А.К.

Москва, 2023

Содержание

Введение

Композиционные материалы — это искусственно созданные материалы, состоящие из нескольких других с четкой границей между ними. Композиты обладают теми свойствами, которые не наблюдаются у компонентов по отдельности. При этом композиты являются монолитным материалом, т. е. компоненты материала неотделимы друг от друга без разрушения конструкции в целом. Яркий пример композита - железобетон. Бетон прекрасно сопротивляется сжатию, но плохо растяжению. Стальная арматура внутри бетона компенсирует его неспособность сопротивляться сжатию, формируя тем самым новые, уникальные свойства. 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).

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.). **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.

Датасет со свойствами композитов находится по ссылке: https://drive.google.com/file/d/1B1s5gBlvgU81H9GGolLQVw_SOi-vyNf2/view?usp=sharing

Характеристика первого датасета X_bp представлена на рисунке 1:

	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	Прочность при растяжении, МПа	Потребление смолы, г/м2
0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000	210.000000	70.000000	3000.000000	220.000000
1	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	210.000000	70.000000	3000.000000	220.000000
2	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385	210.000000	70.000000	3000.000000	220.000000
3	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	210.000000	70.000000	3000.000000	220.000000
4	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385	210.000000	70.000000	3000.000000	220.000000
...	...	...	...	...	...	...	...	...	...	...
1018	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576	209.198700	73.090961	2387.292495	125.007669
1019	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401	350.660830	72.920827	2360.392784	117.730099
1020	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047	740.142791	74.734344	2662.906040	236.606764
1021	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840	641.468152	74.042708	2071.715856	197.126067
1022	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708	758.747882	74.309704	2856.328932	194.754342

1023 rows × 10 columns

Рисунок 1 - Датасет X_bp

Характеристика второго датасета X_nup представлена на рисунке 2:

	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	0	4.000000	57.000000
1	0	4.000000	60.000000
2	0	4.000000	70.000000
3	0	5.000000	47.000000
4	0	5.000000	57.000000
...	...	...	...
1035	90	8.088111	47.759177
1036	90	7.619138	66.931932
1037	90	9.800926	72.858286
1038	90	10.079859	65.519479
1039	90	9.021043	66.920143

1040 rows × 3 columns

Рисунок 2 - Датасет X_nup

Объединение двух датасетов делаем по индексу тип объединения INNER и выводим первые 5 строк нашего датасета с помощью функции head (рисунок 3):

	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	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0	0	4.0	57.0
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0	0	4.0	60.0
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0	0	4.0	70.0
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0	0	5.0	47.0
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0	0	5.0	57.0

Рисунок 3 - Объединенный датасет inner

1.2 Описание используемых методов

Обычно рекомендуется начинать с простых, интерпретируемых моделей, таких как линейная регрессия, и если результаты будут неудовлетворительными, то переходить к более сложным, но обычно более точным методам. На рисунке 4 показана взаимосвязь точности и интерпретируемости некоторых алгоритмов:

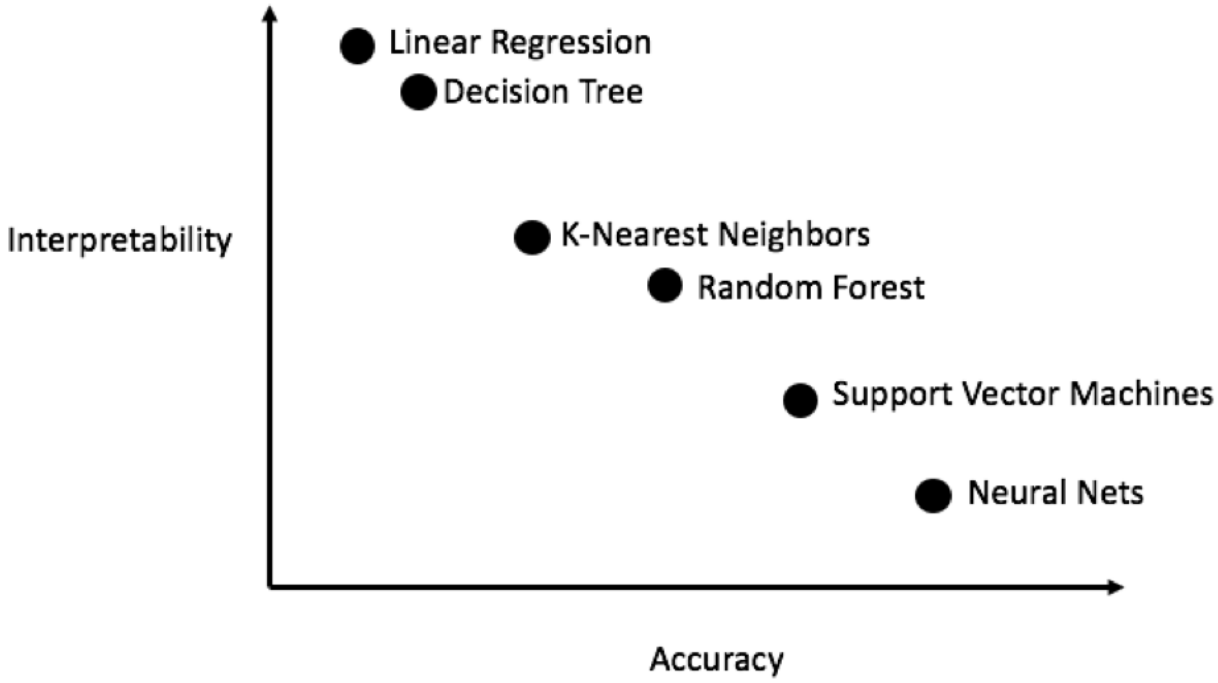

Рисунок 4 - Интерпретируемость и точность моделей

Мы будем оценивать три модели разной степени сложности:

- Линейная регрессия
- Дерево решений
- Метод k-ближайших соседей

Линейная регрессия — это метод машинного обучения с учителем, который используется для предсказания непрерывной целевой переменной от

одного или нескольких независимых признаков. В основе метода лежит предположение предполагает о том, что существует линейная связь между признаками и целевой переменной. Эта связь моделируется с помощью линейной функции. Модель линейной регрессии пытается найти лучшую прямую, которая может описывать зависимость между независимыми признаками и зависимой переменной. Это делается с помощью поиска оптимальных коэффициентов, которые могут быть использованы для описания линейной функции. Эта модель может быть использована как для предсказания, так и для анализа влияния признаков на целевую переменную.

Линейная регрессия может использоваться для решения различных задач. Примерами успешного применения этого метода могут служить такие задачи как: прогнозирование продаж, прогнозирование цены на недвижимость, анализ влияния факторов на уровень заболеваемости и т.д. Однако, она не эффективна для решения задач, где не существует линейной связи между признаками и целевой переменной.

Преимущества линейной регрессии:

- Простота и удобство в использовании: линейная регрессия является одним из самых простых методов машинного обучения, который может быть легко использован и интерпретирован.
- Эффективность при линейных зависимостях: линейная регрессия может предсказывать значения зависимой переменной с высокой точностью, если между независимыми и зависимыми переменными существует линейная связь.
- Интерпретируемость: в линейной регрессии каждый коэффициент регрессии может быть использован для определения влияния каждой независимой переменной на зависимую переменную.

Недостатки линейной регрессии:

- Ограниченная эффективность при нелинейных зависимостях: если между независимыми и зависимыми переменными существует нелинейная зависимость, линейная регрессия может давать неточные предсказания.
- Необходимость проведения предварительной подготовки данных: линейная регрессия чувствительна к выбросам и мультиколлинеарности, так что необходимо выполнить предварительную подготовку данных.

Дерево решений (DecisionTree) — это модель машинного обучения, которая представляет собой дерево с узлами и листьями. Узлы дерева представляют собой решения, которые необходимо принимать, а листья — конечные результаты. В задачах классификации каждый лист дерева соответствует определенному классу, а в задачах регрессии — числовому значению.

Основная идея алгоритма заключается в построении бинарного дерева, в котором каждый внутренний узел представляет собой условие на признаках, а листья — конечный результат работы алгоритма (например, принадлежность к определенному классу).

DecisionTree — это один из наиболее простых и интерпретируемых алгоритмов машинного обучения, который может быть использован для решения различных задач, например, прогнозирования кредитного скоринга, диагностики заболеваний, определения температуры на улице и т.д.

Кроме простоты и интерпретируемости, DecisionTree также имеет ряд **преимуществ** перед другими алгоритмами машинного обучения:

- Может работать с данными различных типов (категориальные, бинарные, числовые)

- Может работать с несбалансированными данными
- Может использоваться для отбора признаков

Однако у DecisionTree также есть некоторые **недостатки**:

- Склонен к переобучению (overfitting) при большой глубине дерева или при отсутствии ограничений на минимальное количество объектов в листе
- Неустойчив к шуму в данных
- Не гарантирует наилучшее решение, так как выбор оптимального разделения происходит на каждом шаге по отдельности, а не глобально

Алгоритм **k ближайших соседей** основан на принципе близости объектов в пространстве признаков. Он состоит в следующем: для каждого объекта из тестовой выборки находим k ближайших соседей из обучающей выборки, и классифицируем объект на основе классов его соседей. Класс, который наиболее часто встречается среди соседей, и будет классом, к которому относится исходный объект.

Преимущества алгоритма k ближайших соседей в задачах классификации заключаются в его простоте и интуитивности.

Однако, он может быть **неэффективен** в случаях, когда обучающая выборка имеет большое количество атрибутов или объектов, поскольку поиск ближайших соседей может быть очень ресурсоемким.

Кроме того, выбор оптимального значения k может оказаться нетривиальной задачей. Слишком маленькое значение k может привести к переобучению модели, тогда как слишком большое значение k может привести к недообучению модели.

Для решения этих проблем существуют различные техники, такие как перекрестная проверка и оптимизация параметров, которые позволяют настроить параметры модели для достижения наилучшего качества классификации.

1.3 Разведочный анализ данных

Разведочный анализ данных (Exploratory Data Analysis или сокращённо EDA) — это предварительное исследование данных с целью выявления основных характеристик данных, типов распределений отдельных переменных, а также зависимостей между переменными.

Основные шаги в EDA:

- Проверка на наличие **дубликатов**. Если дубликаты имеются, их удаляют.
- Проверка на наличие **пропусков** или неинформативных значений. Необходимо принять решение о способе обработки пропусков или удалить такие строки.
- Приведение значений к нужному **типу данных**.
- Проверка на **выбросы** или аномальные значения и их корректировка.
- Проверка на наличие зависимостей между отдельными колонками, выявление **корреляции** с целевой переменной.
- **Визуализация** данных при помощи графиков.

```
#Чтобы проверить наличие дубликатов в данных, мы можем использовать функцию duplicated
full_df.duplicated().sum()
```

```
0
```

Дубликаты и пропуски представлены на рисунках 5 и 6:

Рисунок 5 - Дубликаты

```
#Чтобы проверить наличие пропущенных значений в данных, мы можем использовать функцию isnull
full_df.isnull().sum()
```

```
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
Угол нашивки, град                      0
Шаг нашивки                             0
Плотность нашивки                       0
dtype: int64
```

Рисунок 6 - Пропуски

Типы данных представлены на рисунке 7:

```
#Чтобы узнать общую информацию о нашем датасете, мы можем использовать функцию info
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
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
dtypes: float64(12), int64(1)
memory usage: 111.9 KB
```

Рисунок 7 – Типы данных

Выбросы или аномальные значения — точки данных, которые отклоняются от нормы набора данных, то есть сильно отличаются от основного набора значений.

Существует две техники нахождения выбросов: правило трех сигм и межквартильный (интерквартильный) размах. В данной работе будем использовать вторую технику, изображенную на рисунке 8:

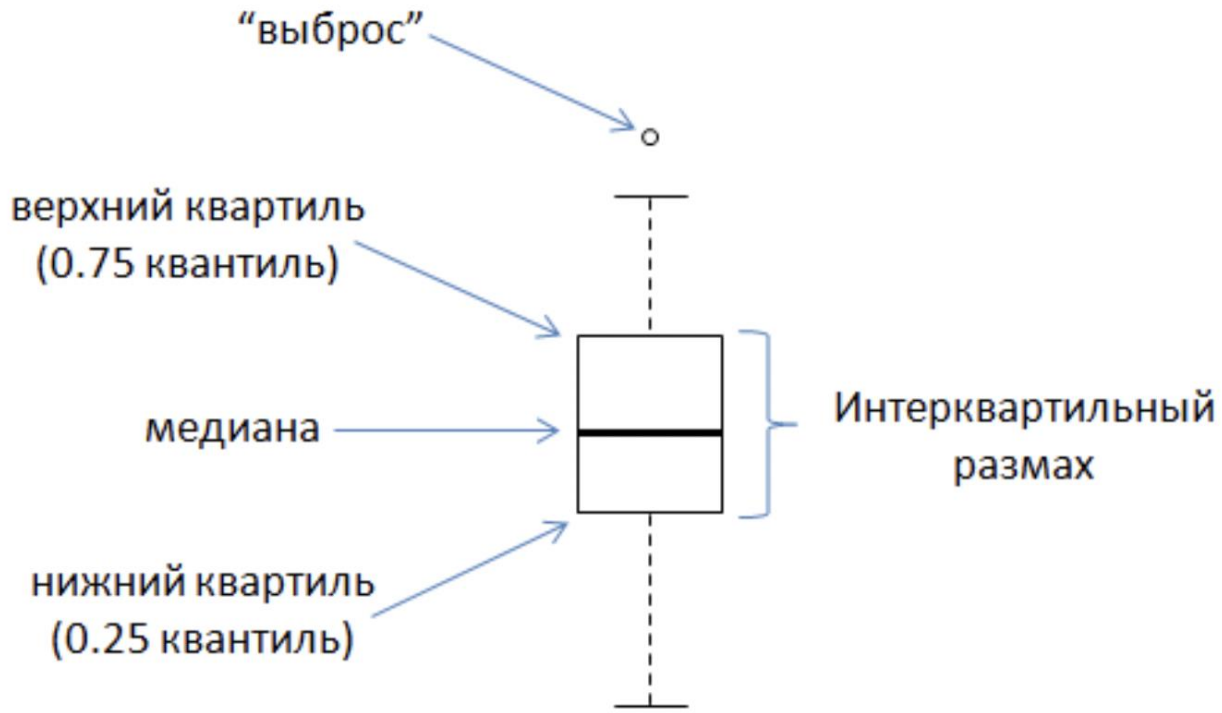

Рисунок 8 - Межквартильный (интерквартильный) размах

Межквартильный размах полезен для оценки разброса случайной величины и представляет собой разницу между 0,75-м и 0,25-м квантилями. Квантиль же в свою очередь отвечает на вопрос о том, какое значение случайная величина не превысит с заданной вероятностью. Квантили 0,25, 0,5 и 0,75 делят область значений на четыре части, поэтому называются квартилями. Пороги для выбросов определяются на основе межквартильного размаха и квартилей.

Корреляция — это статистическая взаимосвязь двух или более случайных величин.

При прямой или положительной зависимости большему значению одной величины соответствует большее значение другой, меньшему значению — меньшее.

При обратной или отрицательной зависимости меньшему значению одной величины соответствует большее значение другой и наоборот.

Корреляция отражается в **группировке** точек одного цвета в одном месте.

Коэффициент корреляции Пирсона — это мера линейной связи между величинами. Нужно помнить: если зависимость не является линейной, а задана полиномом или другой функцией, то коэффициент корреляции не сработает. В таком случае необходимо дополнительно проверить зависимости с помощью визуализации и других способов.

При расчёте коэффициента корреляции измеряется теснота связи. Чем теснее взаимосвязь переменных, тем точки ближе к предполагаемой прямой на графике. Коэффициент корреляции находится в интервале от –1 до 1, где –1 означает отрицательную зависимость, единица — положительную, а ноль говорит о том, что линейной зависимости нет. Чем ближе коэффициент к нулю, тем меньше зависимость; чем ближе к модулю единицы, тем больше зависимость.

Слабая корреляция начинается с 0,5, сильная — с 0,7, высокая — с 0,9.

Общая корреляция по датасету представлена на рисунке 9:

```
#Чтобы посмотреть на корреляцию между столбцами, мы можем использовать функцию corr
full_df.corr()
```

	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	Прочность при растяжении, МПа	Потребление смолы, г/м2
Соотношение матрица-наполнитель	1.000000	0.003841	0.031700	-0.006445	0.019766	-0.004776	-0.006272	-0.008411	0.024148	0.072531
Плотность, кг/м3	0.003841	1.000000	-0.009647	-0.035911	-0.008278	-0.020695	0.044930	-0.017602	-0.069981	-0.015937
модуль упругости, ГПа	0.031700	-0.009647	1.000000	0.024049	-0.006804	0.031174	-0.005306	0.023267	0.041868	0.001840
Количество отвердителя, м.%	-0.006445	-0.035911	0.024049	1.000000	-0.000684	0.095193	0.055198	-0.065929	-0.075375	0.007446
Содержание эпоксидных групп,%_2	0.019766	-0.008278	-0.006804	-0.000684	1.000000	-0.009769	-0.012940	0.056828	-0.023899	0.015165
Температура вспышки, С_2	-0.004776	-0.020695	0.031174	0.095193	-0.009769	1.000000	0.020121	0.028414	-0.031763	0.059954

Рисунок 9 – Корреляция

Визуализация данных — это иллюстрация данных в виде графиков для анализа и обработки. Она позволяет выявить закономерности или аномалии в данных.

Библиотека Matplotlib — одна из самых известных и часто используемых Python-библиотек для создания визуализаций.

Для выявления аномалий наиболее полезны гистограмма и «ящик с усами» (box plot).

Построенные гистограммы представлены на рисунке 10:

```
#Чтобы посмотреть на распределение значений в каждом столбце, мы можем использовать функцию hist
full_df.hist(figsize = (20,20) )
plt.show()
```

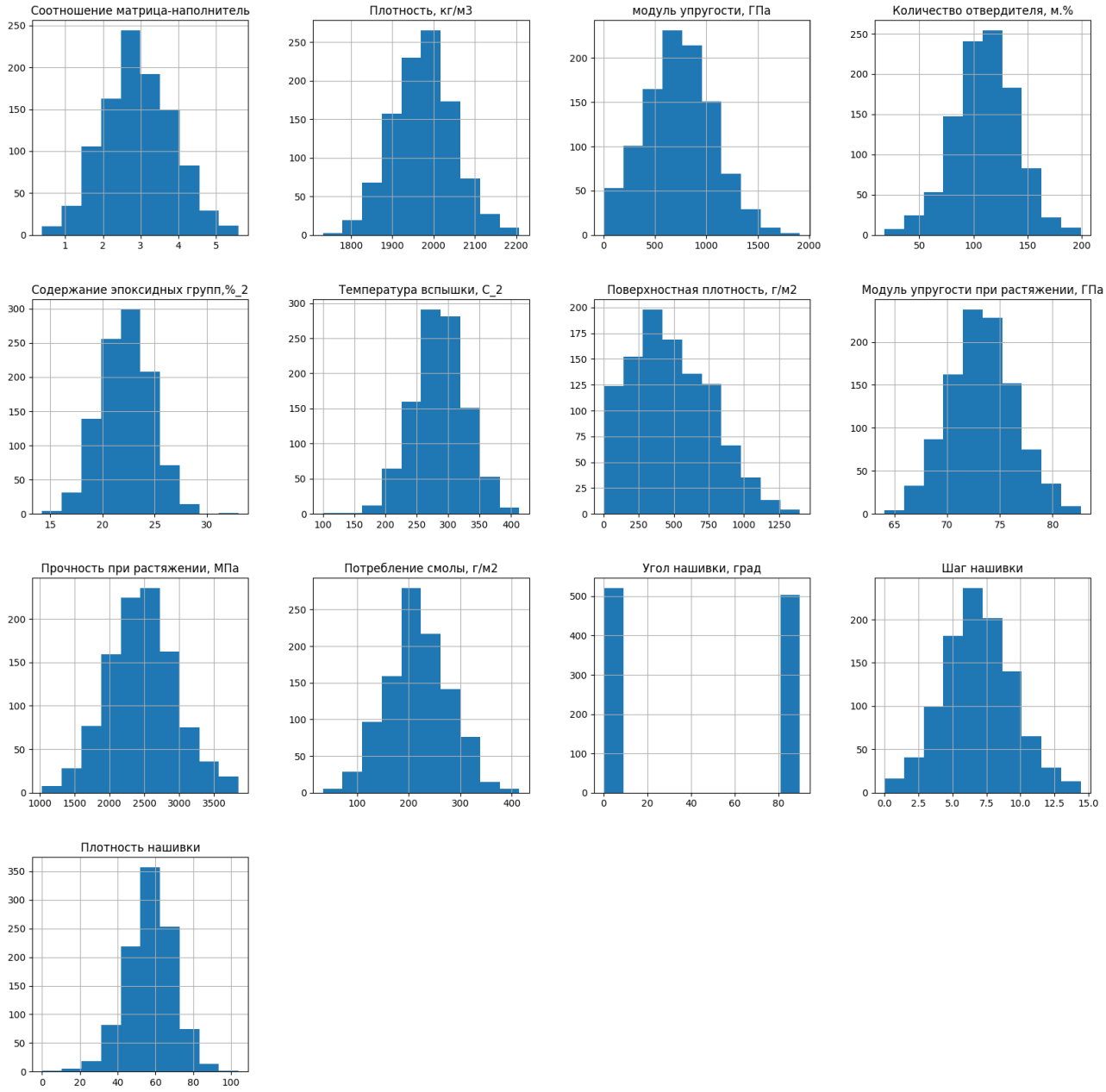

Рисунок 10 – Гистограммы

Box plot («ящик с усами») — это график, который позволяет выявить аномалии в данных. Прямоугольник или «ящик» на графике содержит значения между 0,25-м и 0,75-м квартилем, а «усы» и горизонтальные отсечки показывают значения от минимального до максимального и сами значения. Линия внутри «ящика» отображает медианное значение выборки.

```
#Для построения диаграмм ящика с усами используем метод boxplot() из библиотеки seaborn
fig = plt.figure(figsize=(15,50))
for number, column in enumerate(full_df.columns):
    plt.subplot(13, 2, number+1)
    plt.xlabel(column)
    sns.boxplot(data=full_df[column], width=0.3, palette='Set2', orient='h')
```

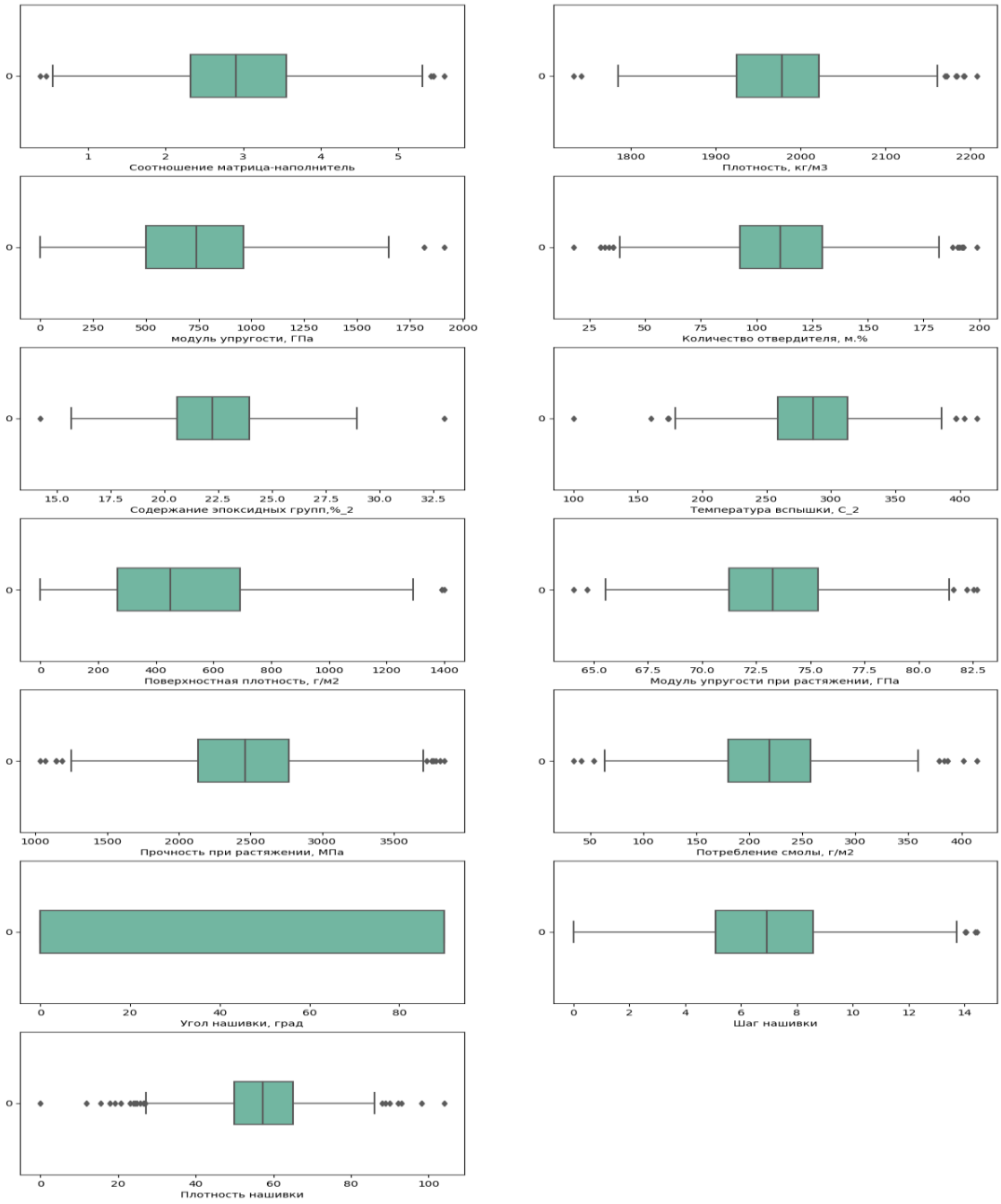

Рисунок 11 – «Ящик с усами»

2 Практическая часть

2.1 Предобработка данных

Пакет sklearn.preprocessing обеспечивает несколько функций общей полезности и трансформаторные классы для изменения необработанных векторов характеристик в представление , которое является более подходящим для нисходящих потока оценок.

В целом алгоритмы обучения выигрывают от стандартизации набора данных. Если в наборе присутствуют какие-то выбросы, более подходящими являются надежные скейлеры или трансформаторы.

Для начала необходимо провести удаление пропусков (рисунок 12):

```
#Удаляем пропуски
full_df_drp = full_df.dropna(axis=0)
full_df_drp
```

	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	Прочность при растяжении, МПа	Потребление смолы, г/м2
1	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	210.000000	70.000000	3000.000000	220.000000
3	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	210.000000	70.000000	3000.000000	220.000000
4	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385	210.000000	70.000000	3000.000000	220.000000
5	2.767918	2000.000000	748.000000	111.860000	22.267857	284.615385	210.000000	70.000000	3000.000000	220.000000
6	2.569620	1910.000000	807.000000	111.860000	22.267857	284.615385	210.000000	70.000000	3000.000000	220.000000
...	...	...	...	...	...	...	...	...	...	...
1018	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576	209.198700	73.090961	2387.292495	125.007669
1019	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401	350.660830	72.920827	2360.392784	117.730099
1020	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047	740.142791	74.734344	2662.906040	236.606764
1021	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840	641.468152	74.042708	2071.715856	197.126067
1022	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708	758.747882	74.309704	2856.328932	194.754342

936 rows × 13 columns

Рисунок 12 – Удаление пропусков

Альтернативная стандартизация — это масштабирование функций таким образом, чтобы они находились между заданным минимальным и максимальным значением, часто между нулем и единицей, или так, чтобы максимальное абсолютное значение каждой функции масштабировалось до размера единицы. Этого можно добиться с помощью MinMaxScaler (рисунок 13):

```
#Нормализуем данные с помощью класса MinMaxScaler
scaler = MinMaxScaler()
full_df_scaler = pd.DataFrame(scaler.fit_transform(full_df_drp), columns = full_df_drp.columns, index=full_df_drp.index)
full_df_scaler.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	Прочность при растяжении, МПа	Потребление смолы, г/м2
count	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000	936.000000
mean	0.498933	0.502695	0.446764	0.504664	0.491216	0.516059	0.373733	0.488647	0.495706	0.521141
std	0.187489	0.187779	0.199583	0.188865	0.180620	0.190624	0.217078	0.191466	0.188915	0.195781
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.372274	0.368517	0.301243	0.376190	0.367716	0.386128	0.205619	0.359024	0.365149	0.392067
50%	0.494538	0.511229	0.447061	0.506040	0.489382	0.515980	0.354161	0.485754	0.491825	0.523766
75%	0.629204	0.624999	0.580446	0.637978	0.623410	0.646450	0.538683	0.615077	0.612874	0.652447
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

Рисунок 13 – Нормализация

Мотивация к использованию этого масштабирования включает устойчивость к очень небольшим стандартным отклонениям функций и сохранение нулевых записей в разреженных данных.

Строим гистограммы после нормализации (рисунок 14):

```
#Гистограммы после нормализации
full_df_scaler.hist(figsize = (20,20) )
plt.show()
```

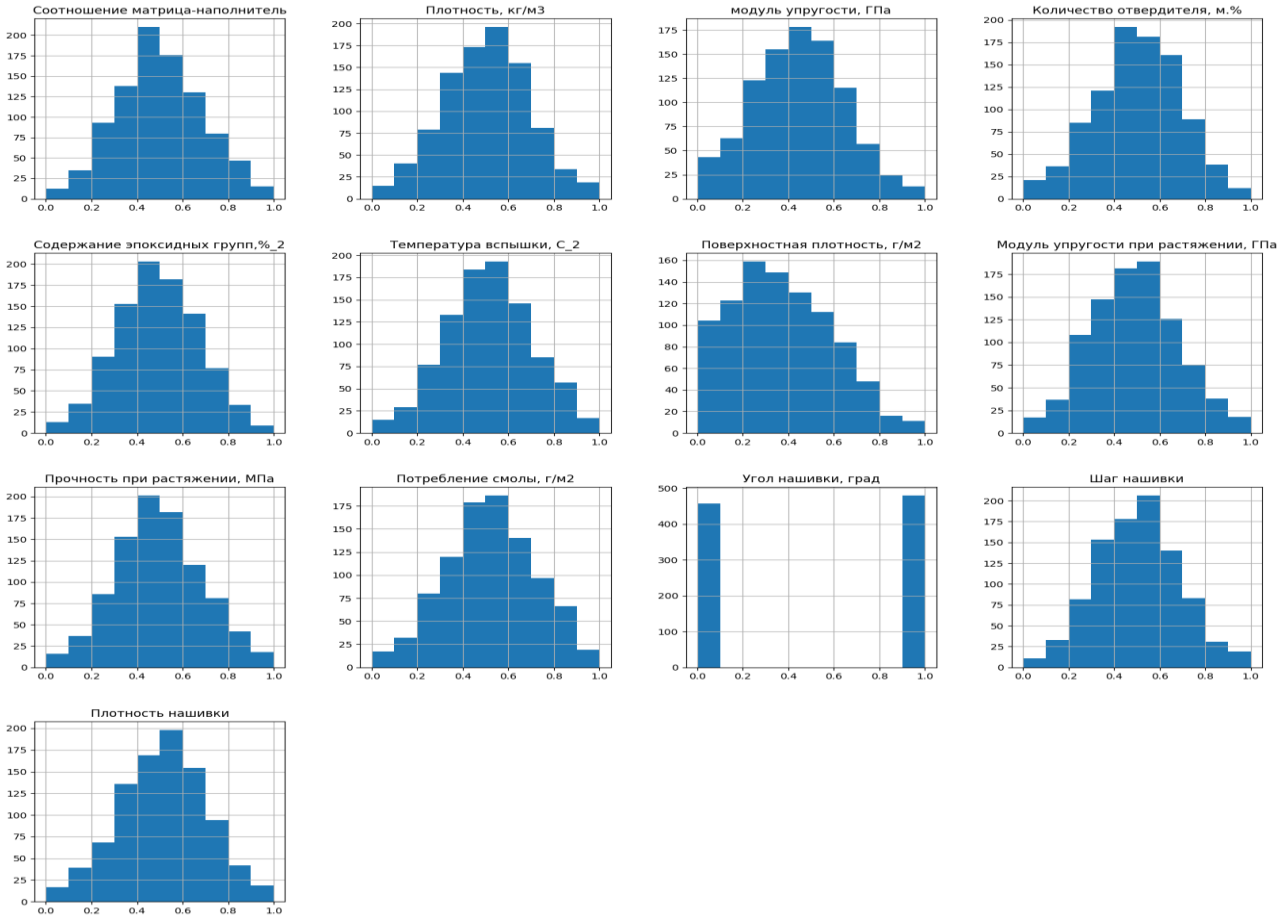

Рисунок 14 – Гистограммы после нормализации

Далее находим минимальные и максимальные значения (рисунок 15):

	Минимальные значения	Максимальные значения
Соотношение матрица-наполнитель	0.55	5.31
Плотность, кг/м3	1784.48	2161.57
модуль упругости, ГПа	2.44	1649.42
Количество отвердителя, м.%	38.67	181.83
Содержание эпоксидных групп,%_2	15.70	28.96
Температура вспышки, С_2	179.37	386.07
Поверхностная плотность, г/м2	0.60	1291.34
Модуль упругости при растяжении, ГПа	65.55	81.42
Прочность при растяжении, МПа	1250.39	3705.67
Потребление смолы, г/м2	63.69	359.05
Угол нашивки, град	0.00	90.00
Шаг нашивки	0.04	13.73
Плотность нашивки	27.27	86.01

Рисунок 15 – Минимальные и максимальные значения

2.2 Разработка и обучение модели

В данной части приводится список моделей, которые будут использоваться для прогноза модуля упругости при растяжении и прочности при растяжении.

При построении моделей необходимо 30% данных оставить на тестирование модели, на остальных происходит обучение моделей (рисунок 16):

```
#Убираем из прогноза столбцы модуля упругости X_dex и прочности при растяжении X_str и присваиваем значения y_dex и y_str
X_dex = full_df_drp.drop(['Модуль упругости при растяжении, ГПа'], axis=1)
y_dex = full_df_drp.loc[:,['Модуль упругости при растяжении, ГПа']]

X_str = full_df_drp.drop(['Прочность при растяжении, МПа'], axis=1)
y_str = full_df_drp.loc[:,['Прочность при растяжении, МПа']]

#30% данных оставляем на тестирование модели, на остальных происходит обучение моделей
X_train_dex, X_test_dex, y_train_dex, y_test_dex = train_test_split(X_dex, y_dex, test_size=0.3, random_state=42, shuffle = True)
X_train_str, X_test_str, y_train_str, y_test_str = train_test_split(X_str, y_str, test_size=0.3, random_state=31, shuffle = True)
```

```
print (X_train_dex.shape)
print (X_test_dex.shape)
print (y_train_dex.shape)
print (y_test_dex.shape)
```

```
(655, 12)
(281, 12)
(655, 1)
(281, 1)
```

Рисунок 16 – Разбивка на тестовую и тренировочную выборки

Все модели строились по следующему шаблону (рисунок 17):

```
#Определяем параметры для поиска лучшей модели
param_grid = [{'fit_intercept': [True, False]}]
#Создаем модель
lin_reg = LinearRegression()
#Обучаем модель
grid_search = GridSearchCV(lin_reg, param_grid, cv = 10, scoring= 'neg_mean_squared_error')
grid_search.fit(X_train_dex, y_train_dex)
#Выводим результат
print(grid_search.best_params_)
print(grid_search.best_score_)
print(sqrt(abs(grid_search.best_score_)))
```

```
{'fit_intercept': True}
-9.713199406381275
3.116600617079653
```

```
#Создаем модель с лучшими параметрами
lin_reg = LinearRegression(fit_intercept = True)
#Обучаем модель на тренировочных данных
lin_reg.fit(X_train_dex, y_train_dex)
#Выводим метрики качества модели на тестовых данных
print(mean_absolute_error(y_test_dex, lin_reg.predict(X_test_dex)))
print(mean_squared_error(y_test_dex, lin_reg.predict(X_test_dex)))
print(sqrt(mean_squared_error(y_test_dex, lin_reg.predict(X_test_dex))))
print(r2_score(y_test_dex, lin_reg.predict(X_test_dex)))
```

Рисунок 17 – Шаблон построения модели линейной регрессии

При построении моделей провести поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10.

2.3 Тестирование модели

В данном разделе показывается ошибка каждой модели на тренировочной и тестирующей части выборки (рисунок 18):

	Модель	MAE	MSE	RMSE	R2
0	Decision Tree	3.514720	18.998326	4.358707	-1.180151
1	Linear Regression	2.414573	8.759731	2.959684	-0.005222
2	KNeighbors	2.595879	10.858807	3.295270	-0.246101

Рисунок 18 – Ошибки моделей

Можно сделать вывод, что лучше всего справилась модель линейной регрессии, так как у нее наименьшие значения MAE, MSE и RMSE, а также наибольшее значение R2.

Модель KNeighbors также показала неплохие результаты, но все же уступает линейной регрессии.

Модель Decision Tree показала наихудшие результаты, так как у нее наибольшее значение MAE, MSE и RMSE, а также отрицательное значение R2, что говорит о том, что модель не смогла объяснить вариативность данных.

Тем не менее ни одна из моделей не показала хороших результатов при предсказании. все значения коэффициента детерминации (R2) отрицательны, что указывает на то, что модели не могут объяснить дисперсию данных. Также значения среднеквадратической ошибки (MSE) высоки, что говорит о том, что модели не могут достаточно точно предсказывать значения целевых переменных.

2.4 Нейронная сеть, рекомендующрая соотношение матрица-наполнитель

```
#Архитектура 3 модели
model_ns = Sequential()
model_ns.add(Dense(8, activation = 'relu', input_shape = (X_train_ns.shape[1],)))
model_ns.add(Dropout(0.2))
model_ns.add(Dense(8, activation='relu'))
model_ns.add(Dropout(0.2))
model_ns.add(BatchNormalization())
model_ns.add(Dense(1, activation = 'sigmoid'))

#Компиляция
model_ns.compile(optimizer = 'adam', loss = 'mean_absolute_error', metrics = ['mse'])
```

Описывается выбранная архитектура нейронной сети и ее результаты.

Рисунок 19 – Архитектура выбранной нейросети

Данная модель представляет собой простую нейронную сеть, состоящую из трех полносвязных слоев. Входной слой содержит 8 нейронов, функция активации ReLU. Далее следует слой Dropout, который помогает избежать переобучения модели. Затем идет второй полносвязный слой с 8 нейронами и функцией активации ReLU. Снова используется слой Dropout, а затем добавлен слой BatchNormalization, который нормализует данные перед передачей в следующий слой. Выходной слой содержит 1 нейрон и функцию активации sigmoid.

Модель скомпилирована с оптимизатором Adam и функцией потерь mean_absolute_error. В качестве метрики используется среднеквадратичная ошибка (mse).

Результаты модели на тестовой выборке показывают среднеквадратичную ошибку 0.184, что является достаточно хорошим результатом для данной задачи.

2.5 Разработка приложения

Для начала мы импортируем библиотеку для работы с командной строкой argparse. Затем создаем парсер аргументов командной строки и добавляем аргументы для ввода пользователем.

Далее загружаем ранее сохраненную модель нейросети, парсим аргументы командной строки, проверяем, что все аргументы были введены пользователем и выводим результат.

Пример кода для разработки приложения с интерфейсом командной строки приведен на рисунке 20.

```
import argparse
import keras

#Создаем парсер аргументов командной строки
parser = argparse.ArgumentParser(description='Приложение для прогнозирования соотношения матрица-наполнитель

#Добавляем аргументы для ввода пользователем
parser.add_argument('density', type=float, help='Плотность, кг/м3')
parser.add_argument('elasticity', type=float, help='Модуль упругости, ГПа')
parser.add_argument('hardener', type=float, help='Количество отвердителя, м.%')
parser.add_argument('epoxy', type=float, help='Содержание эпоксидных групп,%_2')
parser.add_argument('flash', type=float, help='Температура вспышки, С_2')
parser.add_argument('surface_density', type=float, help='Поверхностная плотность, г/м2')
parser.add_argument('tensile_elasticity', type=float, help='Модуль упругости при растяжении, ГПа')
parser.add_argument('tensile_strength', type=float, help='Прочность при растяжении, МПа')
parser.add_argument('resin_consumption', type=float, help='Потребление смолы, г/м2')
parser.add_argument('weaving_angle', type=float, help='Угол нашивки, град')
parser.add_argument('weaving_step', type=float, help='Шаг нашивки')
parser.add_argument('weaving_density', type=float, help='Плотность нашивки')

# Загружаем сохраненную модель
model = keras.models.load_model('model_ns.h5')
#Парсим аргументы командной строки
args = parser.parse_args()
density = args.density
elasticity = args.elasticity
hardener = args.hardener
epoxy = args.epoxy
flash = args.flash
surface_density = args.surface_density
tensile_elasticity = args.tensile_elasticity
tensile_strength = args.tensile_strength
resin_consumption = args.resin_consumption
weaving_angle = args.weaving_angle
weaving_step = args.weaving_step
weaving_density = args.weaving_density
result = model.predict([[density, elasticity, hardener, epoxy, flash, surface_density,
                         tensile_elasticity, tensile_strength, resin_consumption,
                         weaving_angle, weaving_step, weaving_density]])

#Проверяем, что все аргументы были введены пользователем
if not all(vars(args).values()):
    print('Ошибка: не все аргументы были введены')
else: print('Соотношение матрица-наполнитель:', result[0]) #Выводим результат
```

Рисунок 20 – Приложение

2.6 Создание удаленного репозитория и загрузка работы на него

Созданный репозиторий находится по адресу: https://github.com/traintestds/vkr

По ссылке можно найти код исследования, исходные датасеты, объединенный по заданию датасет, сохраненную модель, пояснительную записку и презентацию.

Заключение

В данной работе были проанализированы данные о композитных материалах на основе представленных датасетов. Распределение данных в объединенном датасете оказалось близко к нормальному, но корреляция между признаками практически отсутствует. Были использованы различные модели регрессии для прогнозирования характеристик композитов, но ни одна из них не показала эффективности. Нейросеть также не помогла в решении данного вопроса.

Можно сделать вывод, что текущий набор алгоритмов не позволяет решить задачу прогнозирования свойств композитных материалов на основе имеющихся данных. Для улучшения прогнозов необходимы дополнительные данные, новые признаки, консультации с экспертами и дополнительные исследования.

Библиографический список

1 scikit-learn Машинное обучение в Python: - Режим доступа: https://scikit-learn.ru/ (дата обращения: 25.04.2023)

2 Python School: – Режим доступа: https://python-school.ru/blog/ (дата обращения: 25.04.2023)

3 Medium: – Режим доступа: https://medium.com/nuances-of-programming/ (дата обращения: 25.04.2023)

4 New Tech Audit: - Режим доступа: https://newtechaudit.ru/pandas-merge-join-concatenate/ (дата обращения: 25.04.2023)

5 Habr: - Режим доступа: https://habr.com/ru/companies/nix/articles/425907/ (дата обращения: 25.04.2023)