

**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

Решетников Александр Анатольевич

Москва, 2022

## Содержание

Введение.....	3
1. Теоретическая часть .....	4
1.1. Постановка задачи .....	4
1.2 Описание используемых методов .....	6
1.2.1 Линейная регрессия.....	6
1.2.2 Дерево принятия решений.....	7
1.2.3 Случайный лес .....	9
1.2.4 Нейронная сеть .....	10
1.3 Описание используемых библиотек .....	11
1.3 Методы разведочного анализа данных.....	12
2 Практическая часть .....	14
2.1 Разведочный анализ данных.....	14
2.2 Предобработка данных .....	22
2.3 Разработка и обучение регрессионных моделей .....	26
2.3.1 Модели для прогноза модуля упругости при растяжении .....	27
2.3.2 Модели для прогноза прочности при растяжении .....	28
2.4 Разработка модели нейронной сети .....	30
2.4 Сериализация модели нейронной сети .....	36
2.5 Разработка приложения.....	37
2.6 Репозиторий и результаты .....	38
Заключение .....	39
Библиографический список .....	40

## **Введение**

Пояснительная записка подготовлена в рамках выпускной квалификационной работы по курсу «Data Science» на тему «Прогнозирование конечных свойств новых материалов (композиционных материалов)».

Для решения поставленной задачи изучены теоретические основы предметной области, изучены материалы, предоставленные МГТУ им. Н.Э. Баумана.

Проведены исследования и анализ предоставленных МГТУ им. Н.Э. Баумана данных с использованием методов, изученных на курсе «Data Science».

## 1. Теоретическая часть

### 1.1. Постановка задачи

Постановка задачи предоставлена МГТУ им. Н.Э. Баумана:

#### **Описание:**

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).

**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.

**Датасеты:** Данные для решения поставленных задач представлены в виде двух датасетов разной длины.

Число входных и выходных переменных:

**Входные переменные:**

1. Угол нашивки, град;
2. Шаг нашивки;
3. Плотность;
4. Соотношение матрица-наполнитель;
5. Плотность, кг/м<sup>3</sup>;
6. Модуль упругости, ГПа;
7. Количество отвердителя, м;
8. Содержание эпоксидных групп, %<sub>2</sub>;
9. Температура вспышки, С<sub>2</sub>;
10. Поверхностная плотность, г/м<sup>2</sup>;
11. Модуль упругости при растяжении, ГПа;
12. Прочность при растяжении, МПа;
13. Потребление смолы, г/м<sup>2</sup>.

**Выходные переменные:**

В зависимости от решаемой задачи выходными переменными становятся:

- Соотношение матрица-наполнитель;
- модуль упругости при растяжении, ГПа;
- прочность при растяжении, МПа.

При этом, становясь выходной переменной, соответствующая переменная исключается из входных данных.

## 1.2 Описание используемых методов

Для решения поставленной задачи были использованы методы машинного обучения.

**Машинное обучение** (англ. *machine learning*, ML) — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, математического анализа, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.

Постановка задачи по прогнозированию ряда конечных свойств получаемых композиционных материалов решается задачей регрессии.

**Задача регрессии в машинном обучении** - это предсказание одного параметра (Y) по известному параметру X, где X — набор параметров, характеризующий наблюдение.

Для решения задач были использованы следующие регрессионные методы машинного обучения:

1. Линейная регрессия;
2. Дерево принятия решений;
3. Случайный лес;
4. Многослойный персептрон.

### 1.2.1 Линейная регрессия

Линейная регрессия - 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.

Модель линейной регрессии является часто используемой и наиболее изученной в эконометрике. А именно изучены свойства оценок параметров, получаемых различными методами при предположениях о вероятностных характеристиках факторов, и случайных ошибок модели. Предельные (асимптотические) свойства оценок нелинейных моделей также выводятся исходя из аппроксимации последних линейными моделями. С эконометрической точки зрения более важное значение имеет линейность по параметрам, чем линейность по факторам модели.

Достоинства:

1. Хорошо изучены;
2. Очень быстрые, могут работать на очень больших выборках;
3. Практически вне конкуренции, когда признаков очень много (от сотен тысяч и более), и они разреженные (хотя есть еще факторизационные машины);
4. Коэффициенты перед признаками могут интерпретироваться (при условии, что признаки масштабированы).

Недостатки:

1. Плохо работают в задачах, в которых зависимость ответов от признаков сложная, нелинейная.

### **1.2.2 Дерево принятия решений**

Дерево принятия решений (также называют деревом классификации или регрессионным деревом) - средство поддержки принятия решений, использующееся в машинном обучении, анализе данных и статистике. Структура дерева представляет собой «листья» и «ветки». На рёбрах («ветках») дерева решения записаны признаки, от которых зависит целевая функция, в «листьях» записаны значения целевой функции, а в остальных узлах - признаки, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение.

Достоинства метода:

1. Прост в понимании и интерпретации.

2. Не требует специальной подготовки данных, как например: нормализации данных, добавления фиктивных переменных, а также удаления пропущенных данных.

3. Способен работать как с категориальными, так и с интервальными переменными. (Прочие методы работают лишь с теми данными, где присутствует лишь один тип переменных. Например, метод отношений может быть применён только на номинальных переменных, а метод нейронных сетей только на переменных, измеренных по интервальной шкале.)

4. Использует модель «белого ящика», то есть если определённая ситуация наблюдается в модели, то её можно объяснить при помощи булевой логики. Примером «чёрного ящика» может быть искусственная нейронная сеть, так как полученные результаты сложно объяснить.

5. Позволяет оценить модель при помощи статистических тестов. Это даёт возможность оценить надёжность модели.

6. Метод хорошо работает даже в том случае, если были нарушены первоначальные предположения, включённые в модель.

7. Позволяет работать с большим объёмом информации без специальных подготовительных процедур. Данный метод не требует специального оборудования для работы с большими базами данных.

#### Недостатки:

1. Проблема получения оптимального дерева решений является NP-полной задачей, с точки зрения некоторых аспектов оптимальности даже для простых задач. Таким образом, практическое применение алгоритма деревьев решений основано на эвристических алгоритмах, таких как алгоритм «жадности», где единственно оптимальное решение выбирается локально в каждом узле. Такие алгоритмы не могут обеспечить оптимальность всего дерева в целом.

2. В процессе построения дерева решений могут создаваться слишком сложные конструкции, которые недостаточно полно представляют данные. Данную проблему называют переобучением. Для того, чтобы её избежать, необходимо использовать метод «регулирования глубины дерева».



3. Существуют понятия, которые сложно понять из модели, так как модель описывает их сложным путём. Данное явление может быть вызвано проблемами XOR, чётности или мультиплексарности. В этом случае мы имеем дело с непомерно большими деревьями. Существует несколько подходов решения данной проблемы, например, попытка изменить репрезентацию концепта в модели (составление новых суждений), или использование алгоритмов, которые более полно описывают и репрезентируют концепт (например, метод статистических отношений, индуктивная логика программирования).

4. Для данных, которые включают категориальные переменные с большим набором уровней (закрытий), большой информационный вес присваивается тем признакам, которые имеют большее количество уровней.

### **1.2.3 Случайный лес**

Случайный лес - алгоритм машинного обучения, предложенный Лео Брейманом и Адель Катлер, заключающийся в использовании комитета (ансамбля) решающих деревьев. Алгоритм сочетает в себе две основные идеи: метод бэггинга Бреймана, и метод случайных подпространств, предложенный Тин Кам Хо. Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого ансамбля решающих деревьев, каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим.

Преимущества:

1. Способность эффективно обрабатывать данные с большим числом признаков и классов.

2. Нечувствительность к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков.

3. Одинаково хорошо обрабатываются как непрерывные, так и дискретные признаки. Существуют методы построения деревьев по данным с пропущенными значениями признаков.

4. Существуют методы оценивания значимости отдельных признаков в модели.
5. Внутренняя оценка способности модели к обобщению (тест по неотобраннным образцам).
6. Высокая параллелизуемость и масштабируемость.

Недостатки:

1. Большой размер получающихся моделей. Требуется большой объем памяти для хранения модели.

#### **1.2.4 Нейронная сеть**

Нейронная сеть (НС) - математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети У. Маккалока и У. Питтса. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

НС представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие по отдельности простые процессоры вместе способны выполнять довольно сложные задачи.

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения — одно из главных преимуществ нейронных

сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искажённых данных.

Преимущества:

1. Высокая точность классификации;
2. Сильная способность параллельной распределенной обработки, сильная распределенная память и способность к обучению;
3. Он обладает высокой устойчивостью и отказоустойчивостью к шумовым нервам и может полностью аппроксимировать сложные нелинейные отношения;
4. С функцией ассоциативной памяти.

Недостатки:

1. Нейронные сети требуют большого количества параметров, таких как топология сети, начальные значения весов и пороговых значений;
2. Невозможно наблюдать процесс обучения между ними, а полученные результаты трудно объяснить, что повлияет на достоверность и приемлемость результатов;
3. Время обучения слишком велико и может даже не достичь цели обучения.

### **1.3 Описание используемых библиотек**

Задачи решались с использованием следующих библиотек Python:

1. NumPy. Общеизвестный модуль для Python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций, обеспечивает функционал, который можно сравнить с функционалом MatLab. NumPy (Numeric Python) предоставляет базовые методы для манипуляции с большими массивами и матрицами.

2. Matplotlib. Библиотека на языке программирования Python для визуализации данных двумерной (2D) графикой (3D графика также

поддерживается). Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях.

3. Pandas. Данная библиотека делает Python мощным инструментом для анализа данных. Пакет дает возможность строить сводные таблицы, выполнять группировки, предоставляет удобный доступ к табличным данным.

4. Scikit-learn. Один из наиболее широко используемых пакетов Python для Data Science и Machine Learning. Он содержит функции и алгоритмы для машинного обучения: классификации, прогнозирования или разбивки данных на группы.

5. Seaborn. Библиотека для создания статистических графиков на Python. Она основывается на matplotlib и тесно взаимодействует со структурами данных pandas.

6. TensorFlow. Открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия.

7. Math является самым базовым математическим модулем Python. Охватывает основные математические операции, такие как сумма, экспонента, модуль и так далее. Эта библиотека не используется при работе со сложными математическими операциями, такими как умножение матриц.

8. Pickle реализует мощный алгоритм сериализации и десериализации объектов Python. Широко применяется для сохранения и загрузки сложных объектов в Python как поток байтов.

### **1.3 Методы разведочного анализа данных**

Для разведочного анализа данных используются графические методы – построение гистограмм, ящичковых диаграмм, графики рассеяния, а также аналитические, например, корреляционный анализ для установления статистических зависимостей между переменными.

Корреляционный анализ - статистический метод изучения взаимосвязи между двумя и более случайными величинами. В качестве случайных величин в

эмпирических исследованиях выступают значения переменных, измеряемые свойства исследуемых объектов наблюдения. Суть корреляционного анализа заключается в расчете коэффициентов корреляции. Коэффициенты корреляции могут принимать, как правило, положительные и отрицательные значения. Знак коэффициента корреляции позволяет интерпретировать направление связи, а абсолютное значение – силу связи.

Способ расчета коэффициентов корреляции зависит от шкал измерения переменных, между которыми исследуется взаимосвязь. Для переменных, измеряемых в количественной шкале (интервальной шкале или шкале отношений), рассчитывают ковариацию или корреляционный момент, а на его основе линейный коэффициент корреляции (коэффициент корреляции Пирсона).

Для оценки силы и направления связи между переменными, измеренными в порядковой шкале, используются непараметрические ранговые коэффициенты корреляции: коэффициент ранговой корреляции Кендалла и коэффициент корреляции Спирмена. Также часто используют коэффициент корреляции знаков Фехнера, коэффициент множественной ранговой корреляции (коэффициент Конкордации). Существуют меры оценки связи и между дихотомическими переменными.

## 2 Практическая часть

### 2.1 Разведочный анализ данных

Для работы были предоставлены 2 датасета: X\_bp.xlsx, X\_nup.xlsx. Проверим количество записей командами:

```
df_bp = pd.read_excel(
    '/content/drive/MyDrive/Datasets/MGTU-DS-VKR-2022/X_bp.xlsx', index_col=0)
df_nup = pd.read_excel(
    '/content/drive/MyDrive/Datasets/MGTU-DS-VKR-2022/X_nup.xlsx', index_col=0)
print(df_bp.shape)
print(df_nup.shape)
```

```
(1023, 10)
(1040, 3)
```

Датасет «X\_nup.xlsx» имеет на 17 записей больше, чем «X\_bp.xlsx».

Проверим типы данных командами:

```
print(df_bp.dtypes)
print()
print(df_nup.dtypes)
```

Соотношение матрица-наполнитель	float64
Плотность, кг/м3	float64
модуль упругости, ГПа	float64
Количество отвердителя, м.%	float64
Содержание эпоксидных групп, %_2	float64
Температура вспышки, C_2	float64
Поверхностная плотность, г/м2	float64
Модуль упругости при растяжении, ГПа	float64
Прочность при растяжении, МПа	float64
Потребление смолы, г/м2	float64
dtype: object	
Угол нашивки, град	float64
Шаг нашивки	float64
Плотность нашивки	float64
dtype: object	

Не смотря на то, что в данных присутствуют целочисленные значения, все типы данных по умолчанию выбраны как числа с плавающей запятой двойной точности. Строковых данных нет.

Проверим наличие пропусков командами:

```
print("df_bp количество пропущенных значений :", df_bp.isna().sum().sum())
```

---

```
print("df_nup количество пропущенных значений :", df_nup.isna().sum().sum())
print()
print("df_bp количество NULL значений :", df_bp.isnull().sum().sum())
print("df_nup количество NULL значений :", df_nup.isnull().sum().sum())
```

---

```
df_bp количество пропущенных значений : 0
df_nup количество пропущенных значений : 0
```

---

```
df_bp количество NULL значений : 0
df_nup количество NULL значений : 0
```

---

Пропуски и нулевые значения отсутствуют.

Проверим значения на уникальность командами:

---

```
print(df_bp.nunique())
print()
print(df_nup.nunique())
```

---

```
Соотношение матрица-наполнитель      1014
Плотность, кг/м3                       1013
модуль упругости, ГПа                  1020
Количество отвердителя, м.%            1005
Содержание эпоксидных групп,%_2       1004
Температура вспышки, C_2               1003
Поверхностная плотность, г/м2         1004
Модуль упругости при растяжении, ГПа   1004
Прочность при растяжении, МПа          1004
Потребление смолы, г/м2                1003
dtype: int64

Угол нашивки, град      2
Шаг нашивки             1006
Плотность нашивки       1005
dtype: int64
```

---

Датасеты имеют не уникальные значения. Угол нашивки с виду представляет бинарный параметр, но т.к. предоставлен усеченный набор данных, то считаем, что параметр может и иметь другие значения угла нашивки, не попавшие в данный датасет.

Проведем анализ дублирующихся значений и нулевых строк командами:

---

```
print('Соотношение матрица-наполнитель: ',
      df_bp.duplicated(subset='Соотношение матрица-наполнитель').sum())
print('Плотность, кг/м3: ',
      df_bp.duplicated(subset='Плотность, кг/м3').sum())
print('модуль упругости, ГПа: ',
      df_bp.duplicated(subset='модуль упругости, ГПа').sum())
print('Количество отвердителя, м.:%: ',
      df_bp.duplicated(subset='Количество отвердителя, м.:%').sum())
```

---

---

```

print('Содержание эпоксидных групп,%_2: ',
      df_bp.duplicated(subset='Содержание эпоксидных групп,%_2').sum())
print('Температура вспышки, C_2: ',
      df_bp.duplicated(subset='Температура вспышки, C_2').sum())
print('Поверхностная плотность, г/м2: ',
      df_bp.duplicated(subset='Поверхностная плотность, г/м2').sum())
print('Модуль упругости при растяжении, ГПа: ',
      df_bp.duplicated(subset='Модуль упругости при растяжении, ГПа').sum())
print('Прочность при растяжении, МПа: ',
      df_bp.duplicated(subset='Прочность при растяжении, МПа').sum())
print('Потребление смолы, г/м2: ',
      df_bp.duplicated(subset='Потребление смолы, г/м2').sum())
print('df_bp: ',
      df_bp.duplicated(subset=['Соотношение матрица-наполнитель',
                              'Плотность, кг/м3',
                              'модуль упругости, ГПа',
                              'Количество отвердителя, м.%',
                              'Содержание эпоксидных групп,%_2',
                              'Температура вспышки, C_2',
                              'Поверхностная плотность, г/м2',
                              'Модуль упругости при растяжении, ГПа',
                              'Прочность при растяжении, МПа',
                              'Потребление смолы, г/м2']).sum())

print()
print('Шаг нашивки: ',
      df_nup.duplicated(subset=['Шаг нашивки']).sum())
print('Плотность нашивки: ',
      df_nup.duplicated(subset=['Плотность нашивки']).sum())
print('df_nup: ',
      df_nup.duplicated(subset=['Шаг нашивки', 'Плотность нашивки']).sum())
print()
print('Количество нулевых строк df_bp: ',
      (df_bp[df_bp['Соотношение матрица-наполнитель'] == 0]
       [df_bp['Плотность, кг/м3'] == 0]
       [df_bp['модуль упругости, ГПа'] == 0]
       [df_bp['Количество отвердителя, м.%'] == 0]
       [df_bp['Содержание эпоксидных групп,%_2'] == 0]
       [df_bp['Температура вспышки, C_2'] == 0]
       [df_bp['Поверхностная плотность, г/м2'] == 0]
       [df_bp['Модуль упругости при растяжении, ГПа'] == 0]
       [df_bp['Прочность при растяжении, МПа'] == 0]
       [df_bp['Потребление смолы, г/м2'] == 0]).shape[0])
print()
print('Количество нулевых строк df_nup: ',
      (df_nup[(df_nup['Шаг нашивки'] == 0)
               [(df_nup['Плотность нашивки'] == 0))].shape[0])

```

---



```

Соотношение матрица-наполнитель: 9
Плотность, кг/м3: 10
модуль упругости, ГПа: 3
Количество отвердителя, м.:%: 18
Содержание эпоксидных групп,%_2: 19
Температура вспышки, C_2: 20

```

---



```
Поверхностная плотность, г/м2: 19
Модуль упругости при растяжении, ГПа: 19
Прочность при растяжении, МПа: 19
Потребление смолы, г/м2: 20
df_bp: 0
```

```
Шаг нашивки: 34
Плотность нашивки: 35
df_nup: 19
```

```
Количество нулевых строк df_bp: 0
```

```
Количество нулевых строк df_nup: 1
```

Датасеты имеют дублирующиеся значения и нулевые строки. На следующих этапах будут исключены из датасетов дублирующиеся значения и нулевые строки, а также произведено объединение датасетов, что будет описано в разделе 2.2 Предобработка данных.

Получим первичное представление о статистических характеристиках объединенного датасета командами:



```
df_describe = df.describe()
df_describe.T
```



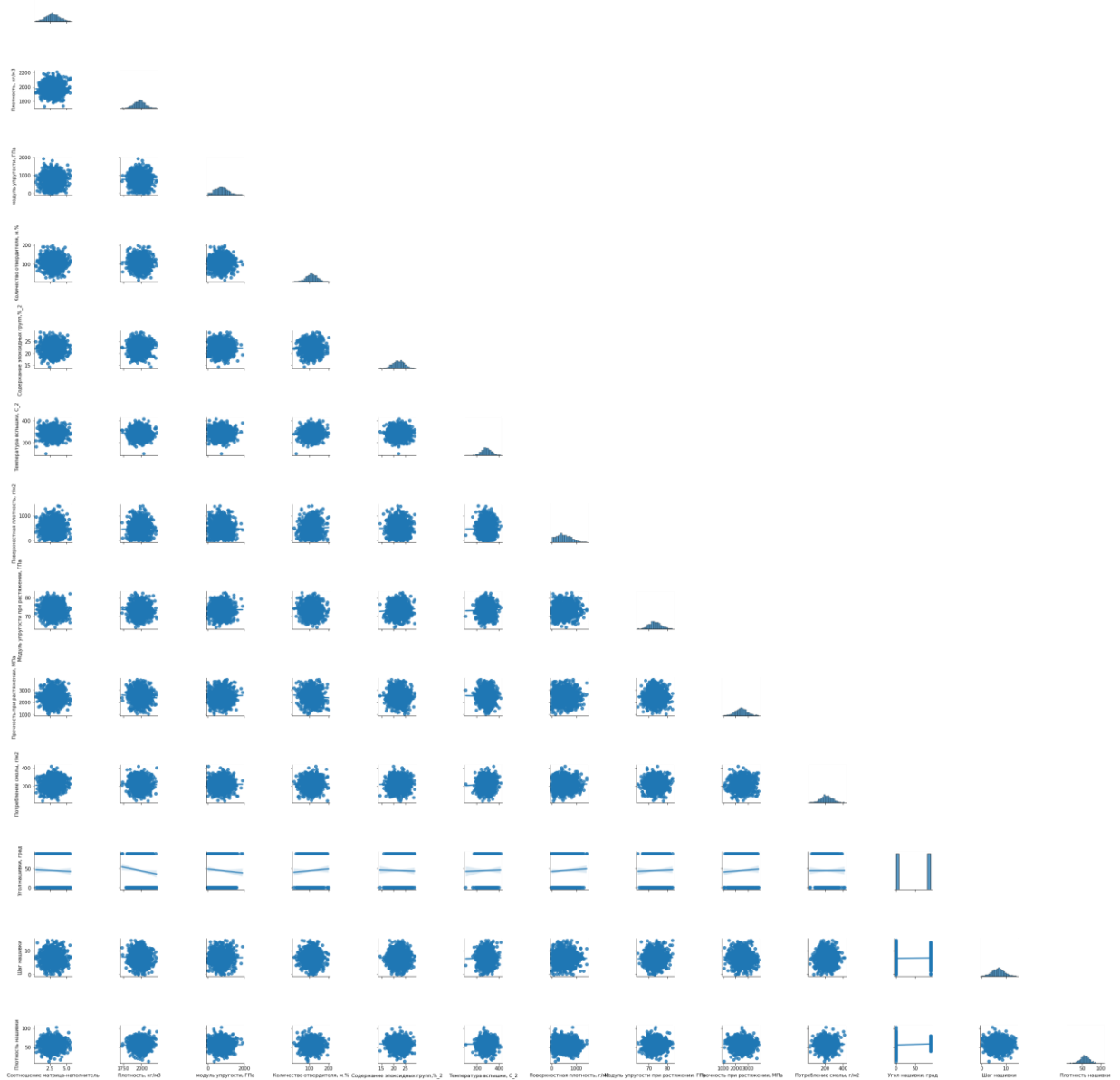
	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1002.0	2.930165	0.913871	0.389403	2.317247	2.907832	3.552781	5.591742
Плотность, кг/м3	1002.0	1975.675440	73.757180	1731.764635	1924.370115	1977.574305	2021.186675	2207.773481
модуль упругости, ГПа	1002.0	740.098060	330.030446	2.436909	498.538615	741.148111	962.650230	1911.536477
Количество отвердителя, м.%	1002.0	110.479158	28.396466	17.740275	92.054117	110.162666	130.240418	198.953207
Содержание эпоксидных групп,%_2	1002.0	22.242882	2.404798	14.254985	20.563359	22.230761	23.981598	28.955094
Температура вспышки, С_2	1002.0	285.739807	41.343587	100.000000	258.469516	285.853960	313.472775	413.273418
Поверхностная плотность, г/м2	1002.0	482.649366	280.682398	0.603740	267.736782	451.944708	693.654483	1399.542362
Модуль упругости при растяжении, ГПа	1002.0	73.326808	3.118688	64.054061	71.297280	73.247594	75.365124	82.682051
Прочность при растяжении, МПа	1002.0	2467.050190	485.889244	1036.856605	2141.720311	2461.249253	2760.983489	3848.436732
Потребление смолы, г/м2	1002.0	218.290295	59.840786	33.803026	179.147494	217.277006	257.488673	414.590628
Угол нашивки, град	1002.0	44.910180	45.022382	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1002.0	6.907026	2.557644	0.037639	5.132313	6.909686	8.564373	14.440522
Плотность нашивки	1002.0	57.234866	12.330789	11.740126	49.922625	57.362576	65.094083	103.988901

При незначительном разбросе значений соотношения матрица-наполнитель достаточно большой разброс по параметрам "Модуль упругости", "Поверхностная плотность", "Прочность при растяжении".

Построим парные графики рассеяния точек командой:



```
sns.pairplot(df, kind='reg', corner=True)
```

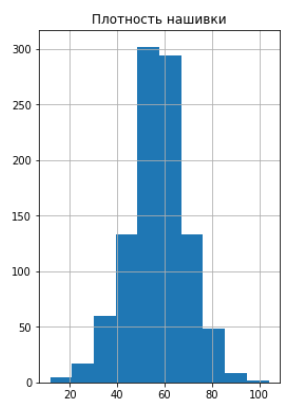
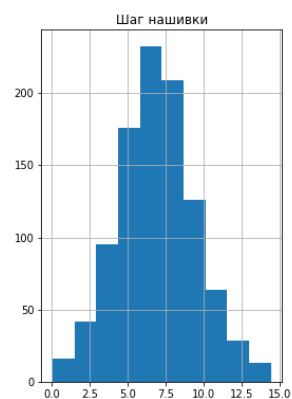
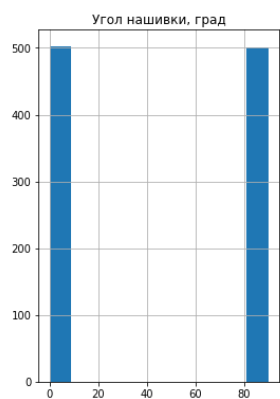
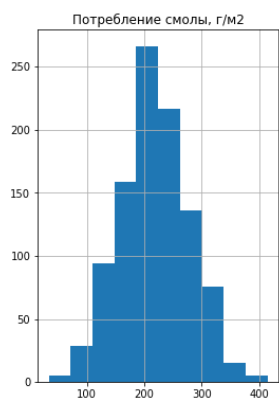
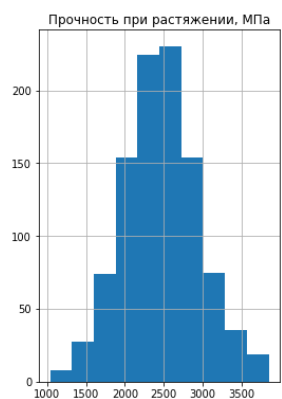
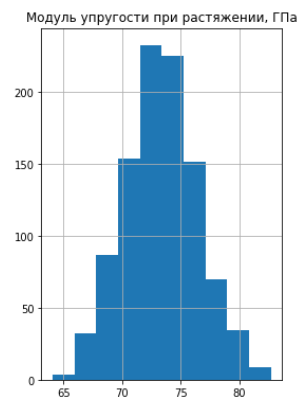
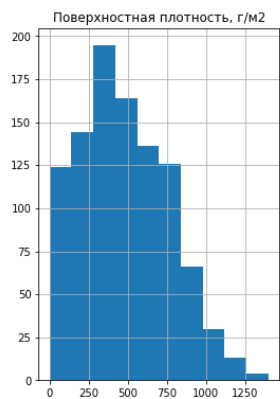
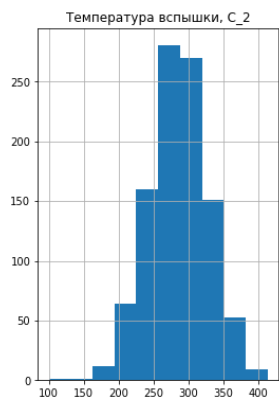
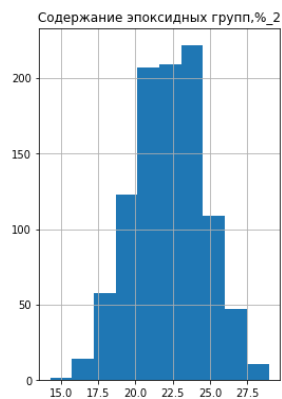
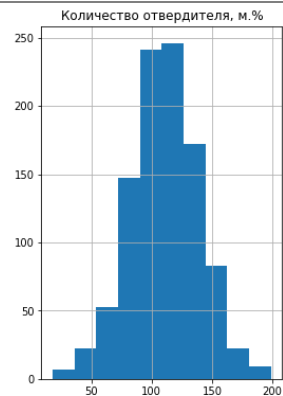
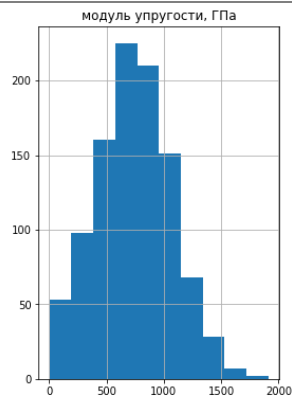
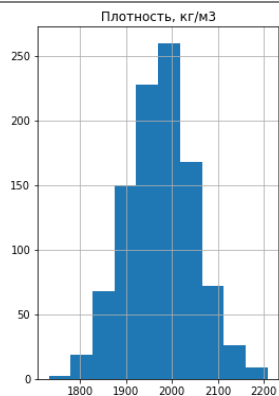
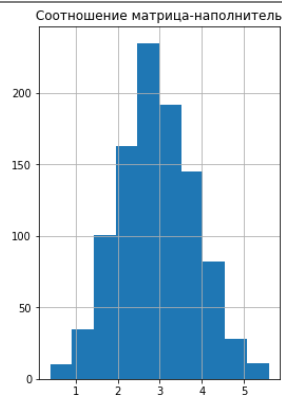


Наблюдается наличие выбросов и отсутствие корреляции.

Построим гистограммы распределения командой:

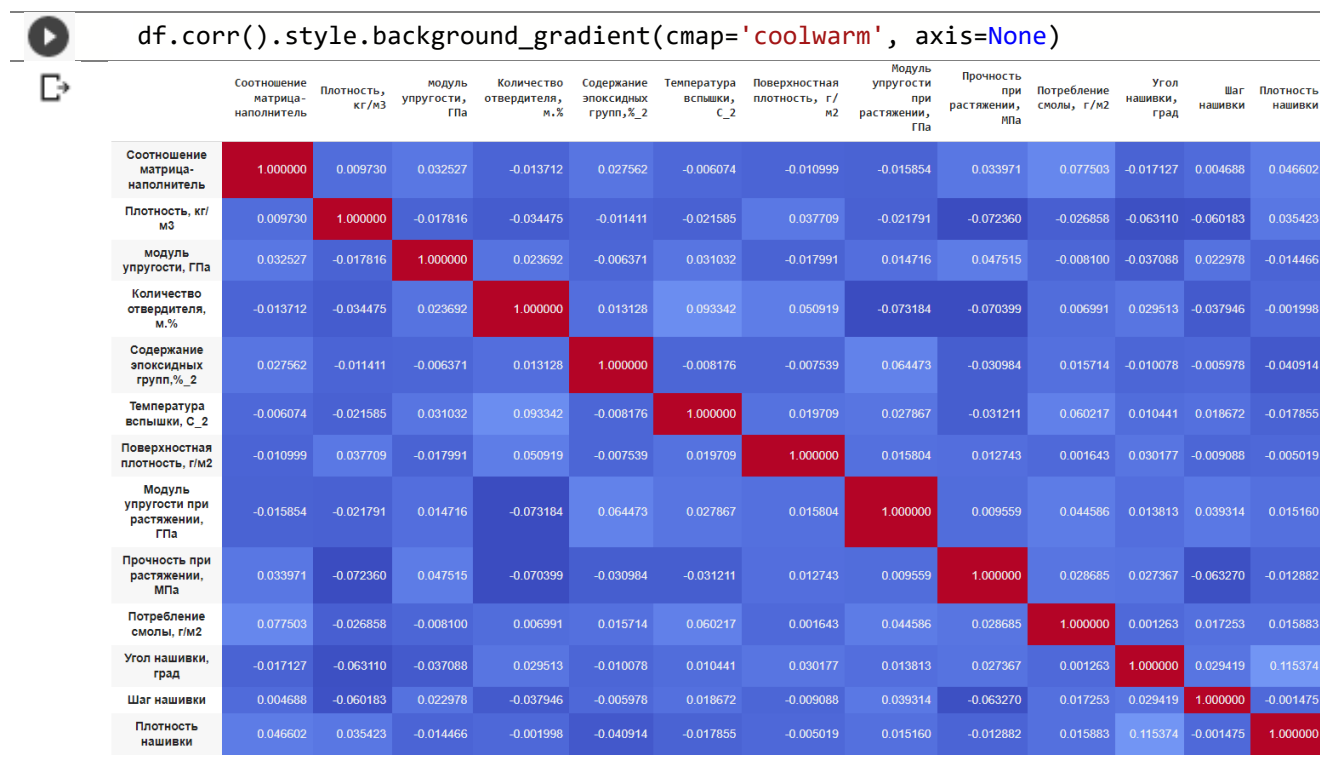


```
df.hist(bins=10, figsize=(20, 30))
```



Гистограммы имеют нормальное распределение, за исключением параметров "Поверхностная плотность" и "Угол нашивки".

Построим тепловую карту корреляции командами:



Полученная тепловая карта так же свидетельствует об отсутствии явной корреляции между признаками.

Отобразим выбросы с помощью диаграммы ящика с усами:

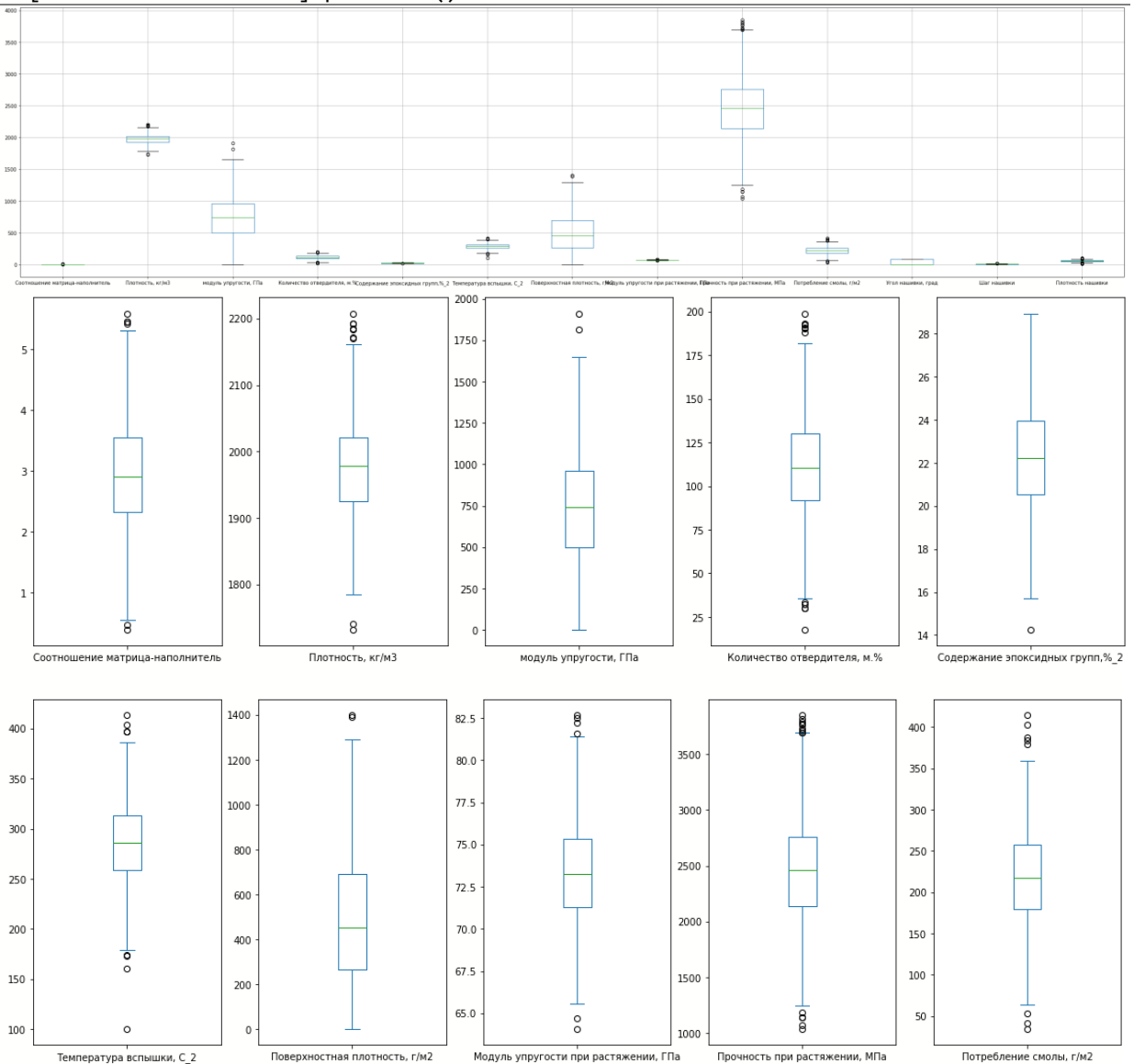
```
df.boxplot(figsize=(40, 10))

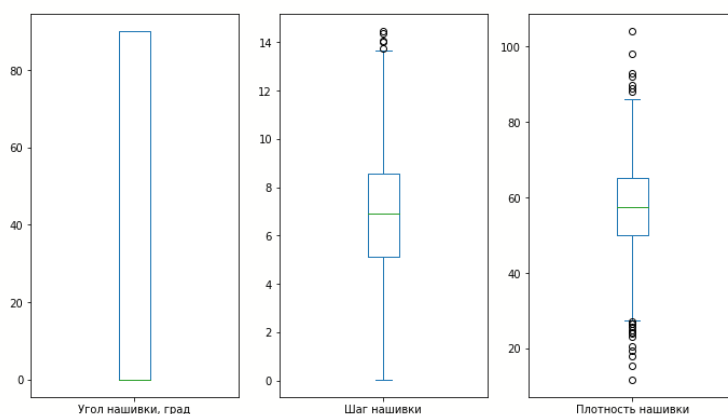
fig=plt.figure(figsize=(20, 30))
ax=fig.add_subplot(4, 5, 1)
df['Соотношение матрица-наполнитель'].plot.box()
ax=fig.add_subplot(4, 5, 2)
df['Плотность, кг/м3'].plot.box()
ax=fig.add_subplot(4, 5, 3)
df['модуль упругости, ГПа'].plot.box()
ax=fig.add_subplot(4, 5, 4)
df['Количество отвердителя, м.%'].plot.box()
ax=fig.add_subplot(4, 5, 5)
df['Содержание эпоксидных групп, %_2'].plot.box()
ax=fig.add_subplot(4, 5, 6)
df['Температура вспышки, С_2'].plot.box()
ax=fig.add_subplot(4, 5, 7)
```

```

df['Поверхностная плотность, г/м2'].plot.box()
ax=fig.add_subplot(4, 5, 8)
df['Модуль упругости при растяжении, ГПа'].plot.box()
ax=fig.add_subplot(4, 5, 9)
df['Прочность при растяжении, МПа'].plot.box()
ax=fig.add_subplot(4, 5, 10)
df['Потребление смолы, г/м2'].plot.box()
ax=fig.add_subplot(4, 5, 11)
df['Угол нашивки, град'].plot.box()
ax=fig.add_subplot(4, 5, 12)
df['Шаг нашивки'].plot.box()
ax=fig.add_subplot(4, 5, 13)
df['Плотность нашивки'].plot.box()

```





Полученный диаграммы так же свидетельствует о наличии выбросов.

## 2.2 Предобработка данных

Вся информация, поступающая в модель, должна служить целям проекта. Если данные не добавляют никакой ценности, от них следует избавиться. Можно выделить основные типы «ненужных» данных: неинформативные признаки с большим количеством одинаковых значений, дубликаты записей, нулевые значения.

Перед объединением датасетов произведем удаление «ненужных» данных командами:

```
df_bp.drop_duplicates()
df_bp.drop_duplicates(subset='Соотношение матрица-наполнитель', inplace=True)
df_bp.drop_duplicates(subset='Плотность, кг/м3', inplace=True)
df_bp.drop_duplicates(subset='модуль упругости, ГПа', inplace=True)
df_bp.drop_duplicates(subset='Количество отвердителя, м.%', inplace=True)
df_bp.drop_duplicates(subset='Содержание эпоксидных групп,%_2', inplace=True)
df_bp.drop_duplicates(subset='Температура вспышки, С_2', inplace=True)
df_bp.drop_duplicates(subset='Поверхностная плотность, г/м2', inplace=True)
df_bp.drop_duplicates(subset='Модуль упругости при растяжении, ГПа',inplace=True)
df_bp.drop_duplicates(subset='Прочность при растяжении, МПа', inplace=True)
df_bp.drop_duplicates(subset='Потребление смолы, г/м2', inplace=True)
df_bp.drop_duplicates(subset=['Соотношение матрица-наполнитель',
                              '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'], inplace=True)

print('Размер df_bp после удаления дублирующихся значений: ', df_bp.shape)

df_nup.drop_duplicates(subset=['Шаг нашивки'], inplace=True)
df_nup.drop_duplicates(subset=['Плотность нашивки'], inplace=True)
df_nup.drop_duplicates(subset=['Шаг нашивки', 'Плотность нашивки'], inplace=True)
print('Размер df_nup после удаления дублирующихся значений: ', df_nup.shape)

print()
df_bp.drop(index=df_bp[df_bp['Соотношение матрица-наполнитель'] == 0]
            [df_bp['Количество отвердителя, м.%'] == 0]
            [df_bp['Содержание эпоксидных групп,%_2'] == 0]
            [df_bp['Температура вспышки, С_2'] == 0]
            [df_bp['Поверхностная плотность, г/м2'] == 0]
            [df_bp['Модуль упругости при растяжении, ГПа'] == 0]
            [df_bp['Прочность при растяжении, МПа'] == 0]
            [df_bp['Потребление смолы, г/м2'] == 0].index, inplace=True)
print('Размер df_bp после удаления нулевых строк: ', df_bp.shape)

df_nup.drop(index=df_nup[(df_nup['Шаг нашивки'] == 0)
                        [(df_nup['Плотность нашивки'] == 0)].index, inplace=True)
print('Размер df_nup после удаления нулевых строк: ', df_nup.shape)

df_bp.reset_index(drop=True, inplace=True) #скорректируем индексы
df_nup.reset_index(drop=True, inplace=True) #скорректируем индексы
print()

```

---


 Размер df\_bp после удаления дублирующихся значений: (1002, 10)  
 Размер df\_nup после удаления дублирующихся значений: (1003, 3)

Размер df\_bp после удаления нулевых строк: (1002, 10)  
 Размер df\_nup после удаления нулевых строк: (1002, 3)

---

После удаления дублирующихся значений и нулевых значений количество записей в обоих датасетах стали одинаковыми.

Произведем их объединение с типом объединения INNER командами:


---

```

df_loaded = df_bp.join(df_nup, how='inner')
df = df_loaded.copy()
print(df.shape)

```

---


 (1002, 13)

---

Произведем удаление шумов, удалив данные, которые отклоняются от трех стандартных отклонений от среднего, следующими командами:

---

```
df_describe = df.describe()
for column in df.columns:
    mean = df_describe.loc['mean', column]
    std = df_describe.loc['std', column]
    min = mean - 3*std
    max = mean + 3*std
    df.loc[df[column] < min, column] = np.nan
    df.loc[df[column] > max, column] = np.nan

df.dropna(inplace=True)
df.shape
```

---

```
(981, 13)
```

---

Произведем удаление выбросов, удалив данные, которые превышают 1,5 межквартильного размаха командами:

---

```
df_describe = df.describe()
for column in df.columns:
    q25 = df_describe.loc['25%', column]
    q75 = df_describe.loc['75%', column]
    intr_q = q75 - q25
    max = q75 + 1.5 * intr_q
    min = q25 - 1.5 * intr_q
    df.loc[df[column] < min, column] = np.nan
    df.loc[df[column] > max, column] = np.nan

df.dropna(inplace=True)
df.shape
```

---

```
(919, 13)
```

---

Следующим этапом проведем нормализацию данных.

Нормализация - это преобразование данных к неким безразмерным единицам. Иногда — в рамках заданного диапазона, например, [0..1] или [-1..1]. Иногда — с какими-то заданным свойством, как, например, стандартным отклонением равным 1.

Ключевая цель нормализации — приведение различных данных в самых разных единицах измерения и диапазонах значений к единому виду, который позволит сравнивать их между собой или использовать для расчёта схожести объектов.



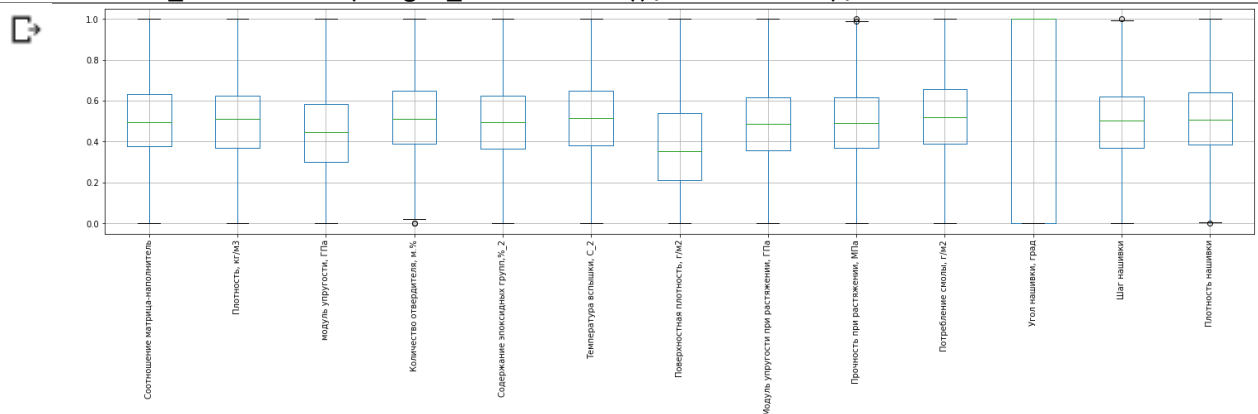
Произведем нормализацию данных командами:

```
df_norm = pd.DataFrame(data=MinMaxScaler().fit_transform(df), columns=df.keys())  
df_norm.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	919.0	0.500285	0.188013	0.0	0.375674	0.495399	0.630121	1.0
Плотность, кг/м3	919.0	0.501584	0.187496	0.0	0.369428	0.510015	0.623659	1.0
модуль упругости, ГПа	919.0	0.446858	0.199830	0.0	0.301109	0.444285	0.583184	1.0
Количество отвердителя, м.%	919.0	0.514035	0.185467	0.0	0.389190	0.512657	0.647038	1.0
Содержание эпоксидных групп, %_2	919.0	0.493728	0.182223	0.0	0.366369	0.492531	0.625220	1.0
Температура вспышки, С_2	919.0	0.514004	0.193129	0.0	0.382837	0.515364	0.646790	1.0
Поверхностная плотность, г/м2	919.0	0.374700	0.215687	0.0	0.209840	0.353522	0.538725	1.0
Модуль упругости при растяжении, ГПа	919.0	0.489157	0.191840	0.0	0.357792	0.485407	0.617966	1.0
Прочность при растяжении, МПа	919.0	0.495848	0.186729	0.0	0.367349	0.492080	0.615393	1.0
Потребление смолы, г/м2	919.0	0.521799	0.196439	0.0	0.388226	0.518199	0.654851	1.0
Угол нашивки, град	919.0	0.515778	0.500023	0.0	0.000000	1.000000	1.000000	1.0
Шаг нашивки	919.0	0.499775	0.185468	0.0	0.370471	0.501345	0.621902	1.0
Плотность нашивки	919.0	0.509439	0.194646	0.0	0.385152	0.508462	0.639992	1.0

После проведенных операций вновь построим диаграмму ящики с усами командами:

```
print("Данные после нормализации")  
ax = df_norm.boxplot(figsize=(25, 5))  
ax.set_xticklabels(ax.get_xticklabels(), rotation=90);
```



Видим, что данные нормализованы в диапазоне от 0 до 1, а также отсутствие выбросов.

## 2.3 Разработка и обучение регрессионных моделей

В ходе работы необходимо обучить несколько моделей для прогноза модуля упругости при растяжении и прочности при растяжении. При построении модели по заданию 30% данных следует оставить на тестирование модели, остальные данные выделить для обучения моделей. При построении моделей необходимо было провести поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10.

Характеристики качества пригодности моделей прогнозирования описывают, насколько достоверно, выбранная в качестве генератора прогноза, модель описывает ретроспективу исследуемого явления. Чем точнее построенная модель объясняла прошлое, тем больше вероятность того, что она будет удачно предсказывать будущее. Надежность моделей прогнозирования оценивается путем сравнения фактических и предсказанных значений. Эта разница позволяет проверить, применима ли к конкретным данным рассматриваемая модель и те предположения, на которых она основана. Основными оценочными характеристиками качества регрессионной модели являются нижеследующие показатели:

MAE - измеряет среднюю абсолютную ошибку прогнозов. Для каждой точки вычисляется разница между прогнозами и целью, а затем усредняются эти значения. Чем дальше это значение от нуля, тем хуже модель. Показатель принимает так же нулевое значение, что может интерпретироваться как обучение идеальной модели, но и как, то, что сумма положительных и отрицательных значений прогнозов в итоге дала ноль.

MSE - измеряет среднюю квадратичную ошибку прогнозов. Для каждой точки вычисляется квадратная разница между прогнозами и целью, а затем усредняются эти значения. Чем выше это значение, тем хуже модель. Он никогда не бывает отрицательным, поскольку мы возводим в квадрат отдельные ошибки прогнозирования, прежде чем их суммировать, но для идеальной модели это будет ноль.

$R^2$  - Коэффициент детерминации, или R-квадрат, является еще одним показателем, который мы можем использовать для оценки модели, и он тесно связан с MSE, но имеет преимущество в том, что не имеет значения, являются ли выходные значения очень большими или очень маленькими,  $R^2$  всегда будет между  $-\infty$  и 1. Когда  $R^2$  отрицательно, это означает, что модель хуже, чем предсказание среднего значения.

### 2.3.1 Модели для прогноза модуля упругости при растяжении

В переменную X сохраняем копию нормализованного дата фрейма, но убираем столбец «Модуль упругости при растяжении». В переменную Y добавляем непосредственно данные по столбцу «Модуль упругости при растяжении», которые будем предсказывать. Далее из sklearn применим метод train\_test\_split и разделим выборку на обучающую и тестовую. Тестовая часть составит 30%.

---

```
X = df_norm.copy()
y = pd.DataFrame(data=X.pop('Модуль упругости при растяжении, ГПа'),
                  columns=['Модуль упругости при растяжении, ГПа'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=42)

print("Size of train data set: ", X_train.shape)
print("Size of train label set: ", y_train.shape)
print("Size of test data set: ", X_test.shape)
print("Size of test label set: ", y_test.shape)
```

---

```
Size of train data set: (643, 12)
Size of train label set: (643, 1)
Size of test data set: (276, 12)
Size of test label set: (276, 1)
```

---

Для подбора гиперпараметров модели дерева решений использовался поиск по сетке с перекрестной проверкой командами:

---

```
param_grid = {
    'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
    'max_depth' : [1, 3, 13, 23, 53]
}
grid = GridSearchCV(DecisionTreeRegressor(), param_grid, cv=10).fit(X_train, y_train)
print('Best params: ', grid.best_params_)
```

---

```
print('Best score: ', grid.best_score_)
```



```
Best params: {'criterion': 'absolute_error', 'max_depth': 1}  
Best score: -0.01584959771618255
```

Для подбора гиперпараметров модели случайный лес использовался поиск по сетке с перекрестной проверкой командами:



```
param_grid = {  
    'criterion': ['squared_error', 'absolute_error'],  
    'n_estimators' : [100, 500],  
    'max_depth' : [30, 100]  
}  
grid = GridSearchCV(RandomForestRegressor(), param_grid, cv=10).fit(X_train, y_train)  
print('Best params: ', grid.best_params_)  
print('Best score: ', grid.best_score_)
```



```
Best params: {'criterion': 'absolute_error', 'max_depth': 30, 'n_estimators': 500}  
Best score: -0.04127576881378331
```

В таблице 1 приведена сводная таблица показатели прогноза моделей.

Таблица 1. Показатели прогноза моделей

	MAE	MSE	R2
Линейная регрессия	0.159633	0.038589	-0.0259701
Дерево решений	0.15958	0.0377647	-0.00405595
Случайный лес	0.162879	0.0392461	-0.0434404

Коэффициент детерминации близкий к нулю, а тем более отрицательное его значение свидетельствует о низком качестве модели и отсутствии линейных связей. MAE и MSE показывает высокие показатели, что так же подтверждает отсутствие линейных связей.

### 2.3.2 Модели для прогноза прочности при растяжении

В переменную X сохраняем копию нормализованного дата фрейма, но убираем столбец «Прочность при растяжении, МПа». В переменную Y добавляем непосредственно данные по столбцу «Прочность при растяжении, МПа», которую будем предсказывать. Далее из sklearn применим метод `train_test_split` и разделим выборку на обучающую и тестовую. Тестовая часть составит 30%.

---

```

▶ X = df_norm.copy()
  y = pd.DataFrame(data=X.pop('Прочность при растяжении, МПа'),
                    columns=['Прочность при растяжении, МПа'])
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=42)

  print("Size of train data set: ", X_train.shape)
  print("Size of train label set: ", y_train.shape)
  print("Size of test data set: ", X_test.shape)
  print("Size of test label set: ", y_test.shape)

```

---

```

↳ Size of train data set: (643, 12)
  Size of train label set: (643, 1)
  Size of test data set: (276, 12)
  Size of test label set: (276, 1)

```

---

Для подбора гиперпараметров модели дерева решений использовался поиск по сетке с перекрестной проверкой командами:

---

```

▶ param_grid = {
    'criterion': ['squared_error'],
    'max_depth' : [1, 2 ,3, 11, 13, 15, 17, 19]
}
grid = GridSearchCV(DecisionTreeRegressor(), param_grid, cv=10).fit(X_train, y_train)
print('Best params: ', grid.best_params_)
print('Best score: ', grid.best_score_)

```

---

```

↳ Best params: {'criterion': 'squared_error', 'max_depth': 1}
  Best score: -0.029239438279347895

```

---

Для подбора гиперпараметров модели случайный лес использовался поиск по сетке с перекрестной проверкой командами:

---

```

▶ param_grid = {
    'criterion': ['squared_error', 'absolute_error'],
    'n_estimators' : [100, 500],
    'max_depth' : [30, 100]
}
grid = GridSearchCV(RandomForestRegressor(), param_grid, cv=10).fit(X_train, y_train)
print('Best params: ', grid.best_params_)
print('Best score: ', grid.best_score_)

```

---

```

↳ Best params: {'criterion': 'absolute_error', 'max_depth': 100, 'n_estimators': 500}
  Best score: -0.043433428509049

```

---

В таблице 2 приведена сводная таблица показатели прогноза моделей.

Таблица 2. Показатели прогноза моделей

	MAE	MSE	R2
Линейная регрессия	0.15369	0.0385339	-0.0653631
Дерево решений	0.151809	0.0377405	-0.0434284
Случайный лес	0.151761	0.0377587	-0.0439306

Коэффициент детерминации близкий к нулю, а тем более отрицательное его значение свидетельствует о низком качестве модели и отсутствии линейных связей. MAE и MSE показывает высокие показатели, что так же подтверждает отсутствие линейных связей.

## 2.4 Разработка модели нейронной сети

В переменную X сохраняем копию дата фрейма, но убираем столбец «Соотношение матрица-наполнитель». В переменную Y добавляем непосредственно данные по столбцу «Соотношение матрица-наполнитель», которые будем предсказывать. Далее из sklearn применим метод train\_test\_split и разделим выборку на обучающую и тестовую. Тестовая часть составит 30%.

```
▶ X = df.copy()
  y = pd.DataFrame(data=X.pop('Соотношение матрица-
    наполнитель'), columns=['Соотношение матрица-наполнитель'])
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_st
    ate=42)

  print("Size of train data set: ", X_train.shape)
  print("Size of train label set: ", y_train.shape)
  print("Size of test data set: ", X_test.shape)
  print("Size of test label set: ", y_test.shape)
```

---

```
↳ Size of train data set: (643, 12)
  Size of train label set: (643, 1)
  Size of test data set: (276, 12)
  Size of test label set: (276, 1)
```

Строим класс обертку над моделью нейронной сети. Необходимо создать функцию инициализации (\_\_init\_\_), которая собирает и компилирует модель

многослойного персептрона. Модель состоит из входного нормализованного слоя и двух скрытых слоев размерностями соответствующими входной модели и активационной функцией ReLu, а также выходной слой размерностью в 1 нейрон. Для компиляции выбираем оптимизатор Adam, а в качестве функции потерь используем среднеквадратичную ошибку. С помощью метода Summary выведем информацию о полученной модели.

В классе обертке дополнительно определяем следующие методы:

- `fit(self)` - метод для обучения модели с предустановленными параметрами: количество эпох – 30; валидационная выборка - 20 % данных;
- `plot_model(self)` - метод для построения графа зависимостей слоев модели;
- `predict(self)` - метод предсказания данных на тестовой выборке;
- `fit_and_predict(self)` - метод для обучения модели и предсказания данных на тестовой выборке;
- `print_metrics(self)` – метод вывода на печать показателей прогнозирования модели;
- `plot_history(self)` – метод построения диаграммы зависимости средней квадратичной ошибки от количества эпох обучения;
- `print_plot(self, lims=[0, 1])` – метод построения графика корреляции предсказанных значений модели от истинных тестовых данных. Параметр `lims` задает диапазон графика и имеет значение по умолчанию `[0, 1]`.



#Класс обертка

```
class NNModel:
```

```
def __init__(self, loss='mean_squared_error', optimizer='adam'):
```

```
    norm = np.array(X_train)
```

```
    self.norm_layer = layers.Normalization(axis=None, input_dim=X_train.shape[1])
```

```
    self.norm_layer.adapt(norm)
```

```
    self.model = keras.Sequential()
```

```
    self.model.add(self.norm_layer)
```

```
    self.model.add(Dense(X_train.shape[1], activation='relu'))
```

```
    self.model.add(Dense(X_train.shape[1], activation='relu'))
```

```
    self.model.add(layers.Dense(1))
```

```
    self.model.compile(loss=loss, optimizer=optimizer)
```

```
    self.model.summary()
```

---

```

def fit(self):
    self.history = self.model.fit(X_train, y_train, validation_split=0.2, verbose=1
, epochs=20, validation_data=(X_test, y_test))

def plot_model(self):
    print(plot_model(self.model, to_file='model_plot.png', show_shapes=True, show_l
ayer_names=True))

def predict(self):
    self.y_pred = self.model.predict(X_test).flatten()

def fit_and_predict(self):
    self.fit()
    self.predict()

def print_metrics(self):
    self.mae = mean_absolute_error(y_test, self.y_pred)
    self.mse = mean_squared_error(y_test, self.y_pred)
    self.r2 = r2_score(y_test, self.y_pred)

    print('MAE: ', self.mae)
    print('MSE: ', self.mse)
    print('R2: ', self.r2)

def plot_history(self):
    plt.plot(self.history.history['loss'], label = 'ошибка на обучающей выборке')
    plt.plot(self.history.history['val_loss'], label = 'ошибка на тестовой выборке'
)
    plt.ylim([0, 3])
    plt.xlabel('Эпоха')
    plt.ylabel('MSE')
    plt.legend()
    plt.grid(True)

def print_plot(self, lims=[0, 1]):
    a = plt.axes(aspect='equal')
    plt.scatter(y_test, self.y_pred)
    plt.xlabel('Истинные значения')
    plt.ylabel('Предсказания')
    plt.xlim(lims)
    plt.ylim(lims)
    _ = plt.plot(lims, lims)

```

---

Строим модель нейронной сети.



```
nn_mf = NNModel()
```

---





Model: "sequential\_2"

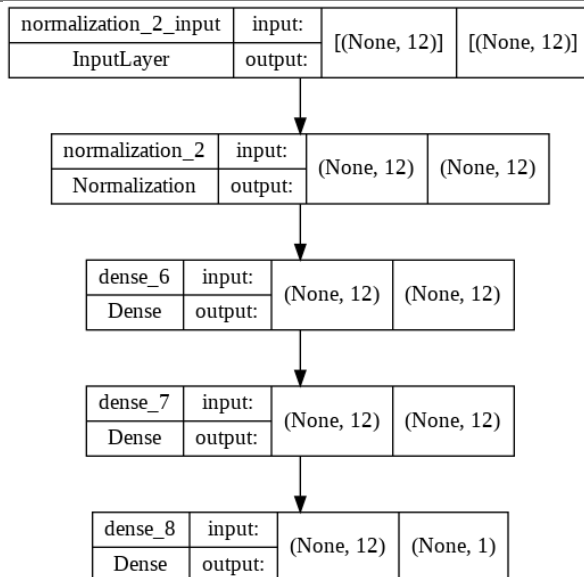
Layer (type)	Output Shape	Param #
=====		
normalization_2 (Normalizat ion)	(None, 12)	3
dense_6 (Dense)	(None, 12)	156
dense_7 (Dense)	(None, 12)	156
dense_8 (Dense)	(None, 1)	13
=====		
Total params: 328		
Trainable params: 325		
Non-trainable params: 3		

Тренировочных параметров в модели 325.

Отообразим модель в виде графа зависимостей слоев модели.



```
plot_model(nn_mf.model, to_file='model_plot.png', show_shapes=True,  
            show_layer_names=True)
```



Вызовем метод обучения модели и предсказания данных на тестовой выборке.



```
nn_mf.fit_and_predict()
```



```
Epoch 1/20  
17/17 [=====] - 1s 12ms/step - loss: 9.2618 - val_loss:  
9.5636  
Epoch 2/20
```

---

```

17/17 [=====] - 0s 3ms/step - loss: 8.4483 - val_loss: 8.8450
Epoch 3/20
17/17 [=====] - 0s 4ms/step - loss: 7.7987 - val_loss: 8.2512
Epoch 4/20
17/17 [=====] - 0s 4ms/step - loss: 7.2838 - val_loss: 7.7365
Epoch 5/20
17/17 [=====] - 0s 4ms/step - loss: 6.7663 - val_loss: 7.1766
Epoch 6/20
17/17 [=====] - 0s 3ms/step - loss: 6.2155 - val_loss: 6.5609
Epoch 7/20
17/17 [=====] - 0s 3ms/step - loss: 5.6026 - val_loss: 5.8896
Epoch 8/20
17/17 [=====] - 0s 3ms/step - loss: 4.9265 - val_loss: 5.1656
Epoch 9/20
17/17 [=====] - 0s 5ms/step - loss: 4.2235 - val_loss: 4.3893
Epoch 10/20
17/17 [=====] - 0s 3ms/step - loss: 3.4946 - val_loss: 3.6170
Epoch 11/20
17/17 [=====] - 0s 3ms/step - loss: 2.7945 - val_loss: 2.9025
Epoch 12/20
17/17 [=====] - 0s 5ms/step - loss: 2.1595 - val_loss: 2.2587
Epoch 13/20
17/17 [=====] - 0s 4ms/step - loss: 1.6295 - val_loss: 1.7653
Epoch 14/20
17/17 [=====] - 0s 4ms/step - loss: 1.2557 - val_loss: 1.4243
Epoch 15/20
17/17 [=====] - 0s 3ms/step - loss: 1.0187 - val_loss: 1.2108
Epoch 16/20
17/17 [=====] - 0s 4ms/step - loss: 0.8791 - val_loss: 1.0752
Epoch 17/20
17/17 [=====] - 0s 3ms/step - loss: 0.8050 - val_loss: 1.0053
Epoch 18/20
17/17 [=====] - 0s 4ms/step - loss: 0.7827 - val_loss: 0.9823
Epoch 19/20
17/17 [=====] - 0s 4ms/step - loss: 0.7767 - val_loss: 0.9765
Epoch 20/20
17/17 [=====] - 0s 4ms/step - loss: 0.7759 - val_loss: 0.9752

```

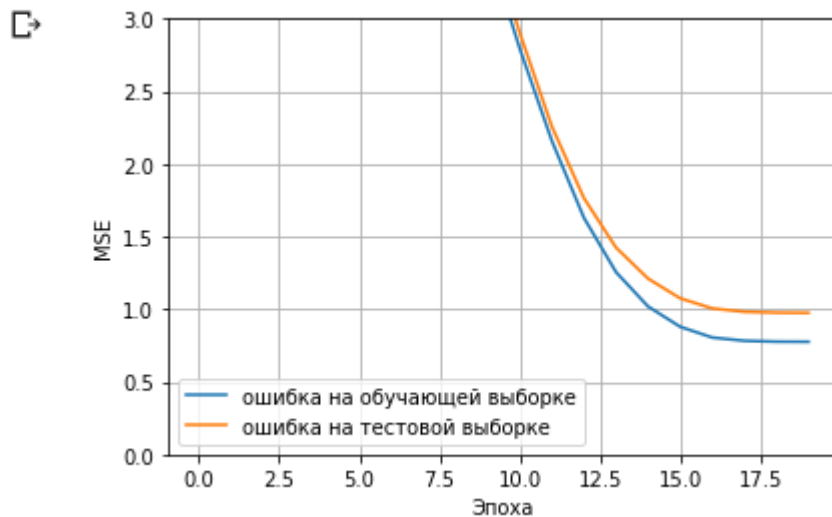
---

Вызовем метод построения диаграммы зависимости средней квадратичной ошибки от количества эпох обучения.



```
nn_mf.plot_history()
```

---

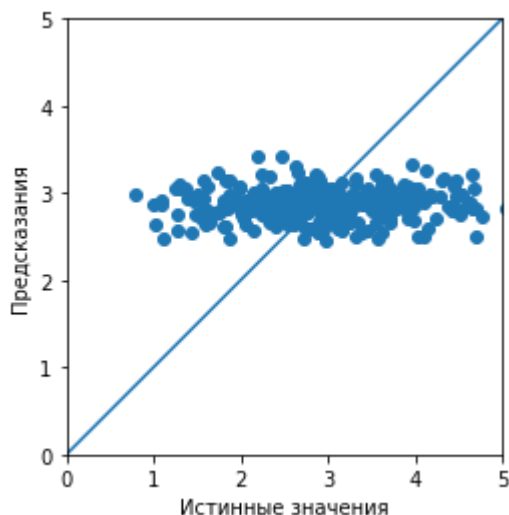


Видно, как на тренировочной части ошибка упала после пятой эпохи. В дальнейшем при необходимости количество эпох обучения можно сократить.

Вызовем метод построения графика корреляции предсказанных значений модели от истинных тестовых данных.

```
nn_mf.print_metrics()  
nn_mf.print_plot(lims=[0, 5]);
```

MAE: 0.7529630048280076  
MSE: 0.8525619071439539  
R2: -0.025004005511227057



Коэффициент детерминации близкий к нулю, а тем более отрицательное его значение свидетельствует о низком качестве модели и отсутствии линейных

связей. MAE и MSE показывает высокие показатели, что так же подтверждает отсутствие линейных связей.

## 2.4 Сериализация модели нейронной сети

Сериализуем модель нейронной сети для последующего использования при разработке приложения. Для сериализации данных используем команды библиотеки Pickle:

```
pickle.dump(nn_mf.model,  
            open('/content/drive/MyDrive/Datasets/MGTU-DS-VKR-2022/mf.pkl',  
                  'wb'))
```

Для проверки десериализуем модель командой:

```
loaded_nn_mf = pickle.load(  
    open('/content/drive/MyDrive/Datasets/MGTU-DS-VKR-2022/mf.pkl', 'rb'))
```

Выведем данные одной из записей тестовой выборки для последующей установки их в приложении для тестирования командой:


```
X_test[:1]
```

	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, C_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
80	1988.301966	177.034495	52.074364	16.977083	292.072967	793.834135	75.102747	2371.373132	172.99571	0.0	5.867705	59.620426


Выведем предсказанное значение командой:

```
loaded_nn_mf.predict(X_test[:1])  
array([[2.5918632]], dtype=float32)
```

Выведем истинное значение командой:



```
y_test[:1]
```



Соотношение матрица-наполнитель

80

3.107885

## 2.5 Разработка приложения

Разработано веб-приложение для рекомендательной системы «Соотношение матрица-наполнитель». Приложение разработано в среде разработки VSCode. Для разработки приложения был использован веб-фреймворк Flask. На рисунках 1 и 2 показан интерфейс приложения. Пользователь вводит 12 параметров в соответствующие окна ввода и при нажатии кнопки «Рекомендовать» приложение выводит ниже кнопки «Рекомендовать» строчку расчетное значение показателя «Соотношение матрица-наполнитель».

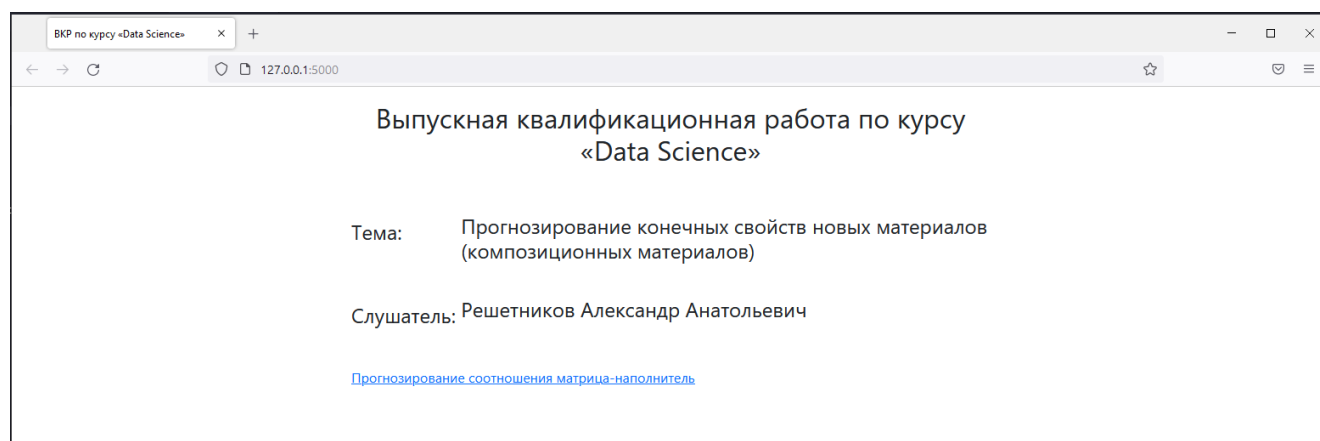


Рисунок 1. Интерфейс главного окна

[На главную](#)

## Прогнозирование соотношения матрица-наполнитель

Плотность, кг/м3	<input type="text" value="1988.301966"/>
Модуль упругости, ГПа	<input type="text" value="177.034495"/>
Количество отвердителя, м.%	<input type="text" value="52.074364"/>
Содержание эпоксидных групп,%_2	<input type="text" value="16.977083"/>
Температура вспышки, C_2	<input type="text" value="292.072967"/>
Поверхностная плотность, г/м2	<input type="text" value="793.834135"/>
Модуль упругости при растяжении, ГПа	<input type="text" value="75.102747"/>
Прочность при растяжении, МПа	<input type="text" value="2371.373132"/>
Потребление смолы, г/м2	<input type="text" value="172.99571"/>
Угол нашивки, град	<input type="text" value="0.0"/>
Шаг нашивки	<input type="text" value="5.867705"/>
Плотность нашивки	<input type="text" value="59.620426"/>
<input type="button" value="Рекомендовать"/>	
Результат:	2.5918632

Рисунок 2. Интерфейс ввода данных

## 2.6 Репозиторий и результаты

Для данного исследования был создан удаленный репозиторий на GitHub, который находится по адресу <https://github.com/realExant/mgtu-ds-vkr>. В него загружены результаты работы: исследовательский notebook, код веб-приложения, пояснительная записка и презентация.

## Заключение

В ходе выполнения данной работы были изучены теоретические основы методов машинного обучения, изучены основные библиотеки Python, как одного из основных инструментов для работы аналитика данных.

На основании практической задачи было выполнено:

- разведочный анализ данных;
- предобработка данных;
- построение регрессионных моделей и модели нейронной сети;
- визуализация модели и оценка качества прогноза;
- сериализация моделей;
- разработка и тестирование веб-приложения.

Результаты построения и обучения моделей не дали положительного результата. Возможные причины неудовлетворительной работы моделей:

- нечеткая постановка задачи, отсутствие дополнительной информации о зависимости между входными датасетами, что привело к неверному объединению датасетов;

- исследование проводилось на предварительно обработанных датасетах. Возможно, на исходных датасетах можно было бы получить более качественные регрессионные модели;

- недостаток знаний и опыта, были испробованы не все возможные методы прогноза.

На основании проведенного исследования можно сделать следующие основные выводы по теме:

- распределение данных близко к нормальному;
- отсутствие корреляции между парами признаков;
- регрессионные модели не показали высокой эффективности в прогнозировании свойств композитов, необходимы дополнительные исходные данные для улучшения моделей.

## Библиографический список

1. Андерсон, Карл. Аналитическая культура. От сбора данных до бизнес-результатов / Карл Андерсон ; пер. с англ. Юлии Константиновой ; [науч. ред. Руслан Салахияев]. — М. : Манн, Иванов и Фербер, 2017. — 336 с.
2. Библиотека Keras: – Режим доступа: <https://keras.io/> (дата обращения: 15.06.2022).
3. Библиотека Matplotlib: – Режим доступа: <https://matplotlib.org/> (дата обращения: 15.06.2022).
4. Библиотека NumPy [электронный ресурс]: – Режим доступа: <https://numpy.org/> (дата обращения: 15.06.2022).
5. Библиотека pickle [электронный ресурс]: – Режим доступа: <https://docs.python.org/3/library/pickle.html/> (дата обращения: 15.06.2022).
6. Библиотека pandas [электронный ресурс]: — Режим доступа: <https://pandas.pydata.org/> (дата обращения: 15.06.2022).
7. Библиотека seaborn: — Режим доступа: <https://seaborn.pydata.org/> (дата обращения: 15.06.2022).
8. Библиотека scikit-learn: — Режим доступа: <https://scikit-learn.org/stable/> (дата обращения: 15.06.2022).
9. Библиотека tabulate: — Режим доступа: <https://pypi.org/project/tabulate/> (дата обращения: 15.06.2022).
10. Библиотека TensorFlow: — Режим доступа: <https://www.tensorflow.org/> (дата обращения: 15.06.2022).
11. Библиотека Flask: — Режим доступа: <https://flask.palletsprojects.com/en/2.1.x/> (дата обращения: 15.06.2022).
12. Билл Любанович. Простой Python. Современный стиль программирования. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O'Reilly»).
13. Брюс П. Практическая статистика для специалистов Data Science. Пер. с англ. / П. Брюс, Э. Брюс, П. Гедек — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2021. — 352 с.: ил.



14. 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.
15. Жерон, Орельен. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем. Пер. с англ. — СПб.: ООО "'Альфа-книга': 2018. — 688 с.: ил
16. Devpractice Team. Pandas. Работа с данными. 2-е изд. - devpractice.ru. 2020. - 170 с.: ил.
17. Джулли А., Пал С. Библиотека Keras - инструмент глубокого обучения. Реализация нейронных сетей с помощью библиотек Theano и TensorFlow / пер. с англ. Слинкин А.А. - М.: ДМК Пресс, 2018. — 294 с.: ил.