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 Демешина Мария Александровна

# 

Москва, 2023

# **Содержание**

[Содержание 2](#_Toc106232838)

[Введение 3](#_Toc106232839)

[1. Аналитическая часть 5](#_Toc106232840)

[1.1. Постановка задачи 5](#_Toc106232841)

[1.2. Описание используемых методов 7](#_Toc106232842)

[1.3. Разведочный анализ данных 17](#_Toc106232843)

[2. Практическая часть 21](#_Toc106232844)

[2.1. Предобработка данных 21](#_Toc106232845)

[2.2. Разработка и обучение модели 22](#_Toc106232846)

[2.3. Тестирование модели 24](#_Toc106232847)

[2.4. Написать нейронную сеть, которая будет рекомендовать соотношение «матрица-наполнитель» 25](#_Toc106232848)

[2.5. Разработка приложения 28](#_Toc106232849)

[2.6. Создание удалённого репозитория и загрузка 30](#_Toc106232850)

[2.7. Заключение 31](#_Toc106232851)

[2.8. Список используемой литературы и веб ресурсы. 32](#_Toc106232852)

# 

**Введение**

Композитные материалы - это материалы, которые состоят из двух или более компонентов, которые объединяются в единое целое. Эти материалы обладают уникальными свойствами, которые не присутствуют у отдельных компонентов. Однако определение свойств композита является проблематичным, даже если известны характеристики входящих компонентов. Для решения этой проблемы используются физические испытания образцов материалов и прогнозирование характеристик. Последний метод основан на симуляции представительного элемента объема композита, на основе данных о характеристиках входящих компонентов. Разработка композитных материалов - долгий процесс, требующий проведения большого количества комбинированных тестов для достижения необходимых характеристик. Однако создание прогнозных моделей помогает снизить количество проводимых испытаний и пополнить базу данных материалов возможными новыми характеристиками материалов и цифровыми двойниками новых композитов. Это может значительно сократить затраты на разработку новых материалов и ускорить процесс их производства.

Для решения этой проблемы в разработке новых материалов используются различные методы, одним из которых является прогнозирование свойств композитных материалов на основе данных об их компонентах. Композиционные материалы представляют собой искусственно созданные материалы, состоящие из двух или более компонентов, объединенных в монолитный материал с четкой границей между ними. Такие материалы обладают уникальными свойствами, которых нет у компонентов по отдельности.

Например, железобетон состоит из бетона и стальной арматуры. Бетон хорошо сопротивляется сжатию, но плохо сопротивляется растяжению. Стальная арматура, в свою очередь, компенсирует это свойство бетона, формируя уникальные характеристики железобетона.

Однако, определение характеристик композитов достаточно проблематично. Для этого можно проводить физические испытания материалов, либо использовать метод прогнозирования на основе данных о характеристиках компонентов материала.

Центр НТИ «Цифровое материаловедение: новые материалы и вещества» в МГТУ им. Н.Э. Баумана работает над созданием прогнозных моделей, которые помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов возможными новыми характеристиками. Такой подход поможет ускорить процесс разработки новых материалов и снизить затраты на этот процесс.

1. **Аналитическая часть**
   1. **Постановка задачи**

Для исследовательской работы были даны 2 файла: X\_bp.xlsx (с данными о параметрах базальтопластика, состоящий из 1024 строки и 11 столбцов) и X\_nup.xlsx (данными нашивок углепластика, состоящий из 1041 строки и 4 столбцов).

Для проведения исследования необходимо объединить два файла с данными, содержащих информацию о параметрах и нашивках. При объединении данных следует учитывать, что часть информации из таблицы нашивок не имеет соответствующих строк в таблице параметров и будет удалена.

Для проведения разведочного анализа данных необходимо построить гистограммы распределения каждой переменной, диаграммы boxplot (ящик с усами) и попарные графики рассеяния точек. Также нужно вычислить среднее и медианное значение каждой колонки, провести анализ и исключение выбросов, проверить наличие пропусков и выполнить предобработку данных, удалив шумы и выбросы, а также сделав нормализацию и стандартизацию.

Далее необходимо обучить несколько моделей для прогнозирования модуля упругости и прочности при растяжении, а также написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель.

Для удобства пользователя нужно создать приложение с графическим интерфейсом, которое будет выдавать прогноз соотношения "матрица-наполнитель". Также необходимо оценить точность моделей на тренировочном и тестовом датасете.

Для хранения кода исследования следует создать репозиторий в GitHub и разместить там все необходимые файлы, включая файл README с кратким описанием проекта и инструкциями для запуска приложения.

**Описание используемых методов**

Для решения задачи классификации категорий машинного обучения, используется машинное обучение с учителем, и более конкретно, задача регрессии. Цель такого алгоритма - определить функцию потерь и минимизировать её. В данной задаче были исследованы и применены следующие методы:

* Линейная регрессия (Linear regression)
* Лассо регрессия (Lasso)
* K-ближайших соседей (KNeighborsRegressor)
* Решающее дерево (DecisionTreeRegressor)
* Стохастический градиентный спуск (SGDRegressor)
* Метод опорных векторов (Support Vector Regression)
* Случайный лес (RandomForest)
* Многослойный перцептрон

**Линейная регрессия (Linear regression)** - это алгоритм машинного обучения, который основан на контролируемом обучении и позволяет определить зависимость между одной входной и выходными переменными. Он использует линию наилучшего соответствия для определения зависимостей между переменными. Для оценки модели регрессии используются несколько метрик, например, коэффициент детерминации (R2), который показывает, насколько хорошо модель может объяснить дисперсию данных. Если R-квадрат равен 1, то модель описывает все данные, а если R-квадрат равен 0,5, то модель объясняет только 50% дисперсии данных. Чем ближе R2 к единице, тем лучше модель.

*Достоинства метода линейной регрессии* - он быстр и прост в реализации, легко интерпретируем и имеет меньшую сложность по сравнению с другими алгоритмами. Недостатки метода - он моделирует только прямые линейные зависимости, требует прямую связь между зависимыми и независимыми переменными, и выбросы оказывают огромное влияние, а границы линейны. Чтобы улучшить линейную модель, используется регуляризация, которая модифицирует функцию стоимости, чтобы ограничить значения коэффициентов. Это позволяет уменьшить общую ошибку за счет обмена некоторой дисперсии на некоторую предвзятость.

*Линейная модель имеет свои недостатки*, такие как возможность моделировать только прямые линейные зависимости и требование прямой связи между зависимыми и независимыми переменными. Кроме того, выбросы могут значительно повлиять на результаты, а границы модели также являются линейными. Чтобы улучшить Линейную модель, можно использовать регуляризацию, которая изменяет функцию стоимости, чтобы ограничить значения коэффициентов. Это позволяет заменить избыточную дисперсию на некоторое смещение, что может потенциально уменьшить общую ошибку.

**Лассо регрессия** - это метод линейной модели, который помогает сократить сложность модели и избежать переобучения, которое может возникнуть при использовании простой линейной регрессии. Этот метод добавляет дополнительный член регуляризации в функцию оптимизации, что позволяет получить более устойчивое решение. В регрессии Лассо используется смещение абсолютного значения, которое помогает уменьшить коллинеарность и дисперсию модели. Метод хорошо работает для моделей временных рядов, таких как авторегрессии.

*Преимущества метода* включают возможность быстро избавляться от шума в данных, быструю работу и способность полностью удалять признаки из датасета. Он также позволяет обнулять значения коэффициентов.

*Недостатки метода* включают низкое качество прогнозирования, возможность ложных срабатываний, случайный выбор одной из коллинеарных переменных, а также невозможность оценки правильности формы взаимосвязи между независимыми и зависимыми переменными.

Лассо-регрессию следует использовать в случае, когда есть несколько характеристик с высокой предсказательной способностью, а остальные бесполезны. Она может обнулять бесполезные характеристики и оставлять только подмножество переменных.

**Метод ближайших соседей (kNN - k Nearest Neighbours)** - это алгоритм, который используется для классификации и регрессии. Он основывается на хранении данных в памяти и сравнении новых элементов с известными значениями целевой переменной. Для этого алгоритм находит расстояния между новым элементом и всеми примерами в данных, выбирая определенное количество примеров (k), наиболее близких к запросу. Затем голосует за наиболее часто встречающуюся метку (в случае задачи классификации) или усредняет метки (в случае задачи регрессии).

Метод ближайших соседей прост в реализации и понимании результатов, не требует построения модели и допускает настройку нескольких параметров. Он также универсален и находит лучшее решение из возможных. Но метод замедляется с ростом объёма данных, не создаёт правил и не обобщает предыдущий опыт. Он также имеет вычислительную трудоемкость и высокую зависимость результатов классификации от выбранной метрики. Также сложно выбрать близость метрики, и он полностью перебирает всю обучающую выборку при распознавании.

**Дерево решений, также известное как DecisionTreeRegressor**, является методом анализа данных, который использует древовидную структуру, чтобы помочь в принятии решений. Это инструмент интеллектуального анализа данных и предсказательной аналитики, который может использоваться для решения регрессионных задач с переменными разного типа. Дерево принятия решений создается на основе обобщения множества обучающих примеров, описывающих предметную область. Регрессия дерева решений позволяет прогнозировать данные в будущем, используя особенности объекта и структуру дерева.

*Достоинства метода* заключаются в его возможности визуализировать процесс принятия решений и помочь сделать правильный выбор в сложных ситуациях. Дерево решений легко применять и интерпретировать, а также может заполнять пропущенные значения наиболее вероятным решением. Метод также может работать с разными типами переменных и выделять наиболее важные поля для прогнозирования.

*Однако у метода есть и недостатки*. Дерево решений может ошибаться при классификации с большим количеством классов и небольшой обучающей выборкой, а также имеет нестабильный процесс, который может привести к построению совсем другого дерева при изменении в одном узле. Метод также может быть затратным в вычислительном плане, и его эффективность может зависеть от размера данных. Кроме того, дерево решений может иметь ограниченное число вариантов решения проблемы.

**SGDRegressor** - это алгоритм машинного обучения, который используется для решения задач линейной регрессии и классификации. Он основан на *стохастическом градиентном спуске*, который является одним из самых популярных методов оптимизации в машинном обучении. Стохастический градиентный спуск работает, используя лишь один случайный обучающий пример для вычисления градиента функции потерь на каждом шаге, что позволяет ускорить процесс обучения.

*Одним из главных преимуществ* SGDRegressor является его высокая скорость обучения. Этот метод может быстро обучаться на больших выборках данных и может быть легко настроен для работы с различными типами данных. Он также прост в реализации и имеет множество возможностей для настройки кода.

*Однако, у SGDRegressor также есть ряд недостатков*. Он может быть чувствителен к масштабированию функций и может сходиться медленно или застрять в локальных минимумах, что может привести к переобучению. Кроме того, этот метод требует настройки гиперпараметров, что может быть сложно для новичков в машинном обучении. Тем не менее, с правильной настройкой параметров, SGDRegressor может дать отличные результаты в решении задач линейной регрессии и классификации.

**Метод опорных векторов (SVM)** - это алгоритм машинного обучения, который используется для решения задач классификации и регрессии. Он работает на основе разделения данных точками в пространстве и создания гиперплоскости, которая разделяет данные на классы. SVM хорошо подходит для работы с небольшими наборами данных и может обрабатывать случаи, когда количество гиперпараметров больше, чем количество наблюдений.

*Среди достоинств метода* можно отметить его эффективность при большом количестве гиперпараметров, возможность гибкой настройки разделяющей функции и максимизация разделяющей полосы, что позволяет уменьшить количество ошибок классификации.

*Однако*, SVM чувствителен к шуму в данных и может страдать от неустойчивости в работе. Для больших наборов данных может потребоваться длительное время обучения, а также сложно подбирать полезные преобразования данных и интерпретировать параметры модели.

В целом, SVM - это мощный и эффективный инструмент машинного обучения, но для решения конкретных задач может потребоваться комбинация различных методов и алгоритмов.

**Многослойный персептрон (MLPRegressor)** - это алгоритм обучения с учителем, который использует многослойную нейронную сеть для решения задач классификации и регрессии. Он состоит из нескольких слоев нейронов, которые передают и обрабатывают информацию между входом и выходом сети. Эти слои включают в себя входной слой, один или несколько скрытых слоев и выходной слой.

*Достоинства метода* включают возможность построения сложных моделей и осуществления любых отображений входных данных в выходные, а также легкость обобщения входных данных и изучения нелинейных моделей.

*Недостатки метода* включают невыпуклую функцию потерь, чувствительность к инициализации случайных весов, необходимость настройки гиперпараметров и чувствительность к масштабированию функций. Однако, благодаря своей гибкости и способности обучаться на больших объемах данных, MLPRegressor является одним из наиболее распространенных алгоритмов машинного обучения.

**Случайный лес (RandomForest)** - это алгоритм машинного обучения, который является комбинацией множества решающих деревьев. Он используется для задач классификации, регрессии и кластеризации. Этот метод хорошо работает на больших и сложных данных и является универсальным, так как он не требует предобработки входных данных и может обрабатывать пропущенные значения.

*Основное преимущество* случайного леса заключается в том, что он не переобучается, что гарантирует хорошую обобщающую способность модели. Кроме того, случайный лес позволяет оценить важность признаков и устойчив к выбросам. Ещё одним преимуществом является высокая параллелизуемость, что позволяет обрабатывать большие объёмы данных.

*Однако,* построение случайного леса занимает много времени, особенно для больших данных. Это делает его менее привлекательным для решения задач в реальном времени. Кроме того, случайный лес сложно интерпретируется и не обладает возможностью экстраполяции.

Также стоит отметить, что случайный лес может недообучаться, особенно если выборка содержит слишком мало информации о классах. И хотя случайный лес является эффективным инструментом, он не всегда превосходит линейные методы.

**Используемые метрики качества моделей:**

**R2** (коэффициент детерминации) и **MSE** (средняя квадратичная ошибка) являются распространенными метриками, используемыми для оценки качества моделей машинного обучения. R2 показывает, насколько хорошо модель подходит под данные и может быть интерпретирован как доля объясненной дисперсии от общей дисперсии целевой переменной. MSE, с другой стороны, показывает, насколько сильно модель ошибается в среднем, измеряя среднее значение квадрата разности между прогнозируемыми и фактическими значениями.

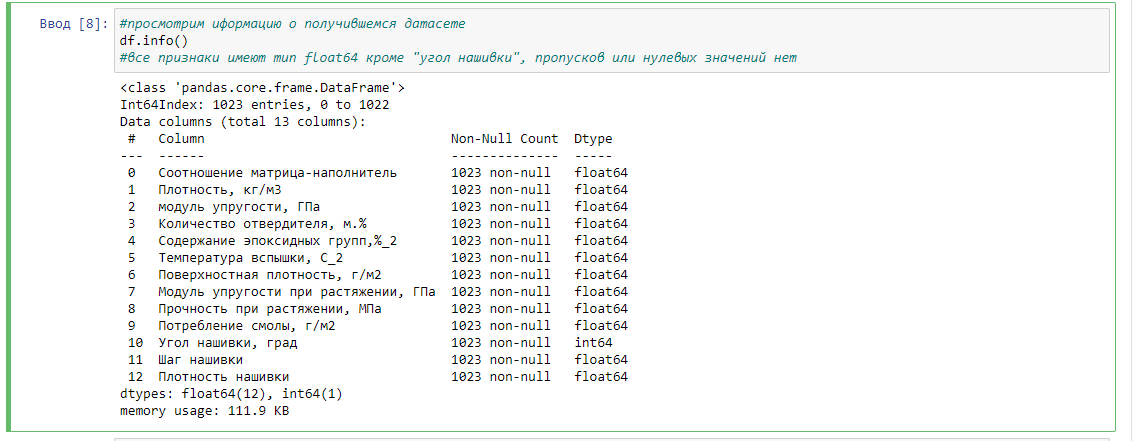
Важно отметить, что каждая метрика имеет свои преимущества и недостатки, и выбор метрики должен быть обоснован исходя из конкретной задачи и целей моделирования. Например, если наша задача заключается в предсказании высоких значений целевой переменной, то MSE может быть более информативной метрикой, чем R2. Кроме того, к выбору метрики также следует подходить осторожно при сравнении моделей, так как различные модели могут иметь разную чувствительность к определенным метрикам.

**1.3. Разведочный анализ данных**

Разведочный анализ данных (Exploratory Data Analysis - EDA) — это процесс изучения набора данных с помощью статистических и визуальных методов, с целью выявления закономерностей, связей и аномалий в данных. Этот процесс является одним из первых шагов в анализе данных и позволяет получить первичное представление о данных.

Прежде чем передать данные в работу моделей машинного обучения, необходимо обработать и очистить их. Необработанные данные могут содержать искажения и пропущенные значения и способны привести к неверным результатам по итогам моделирования. Но безосновательно удалять что-либо тоже неправильно. Именно поэтому сначала набор данных надо изучить.

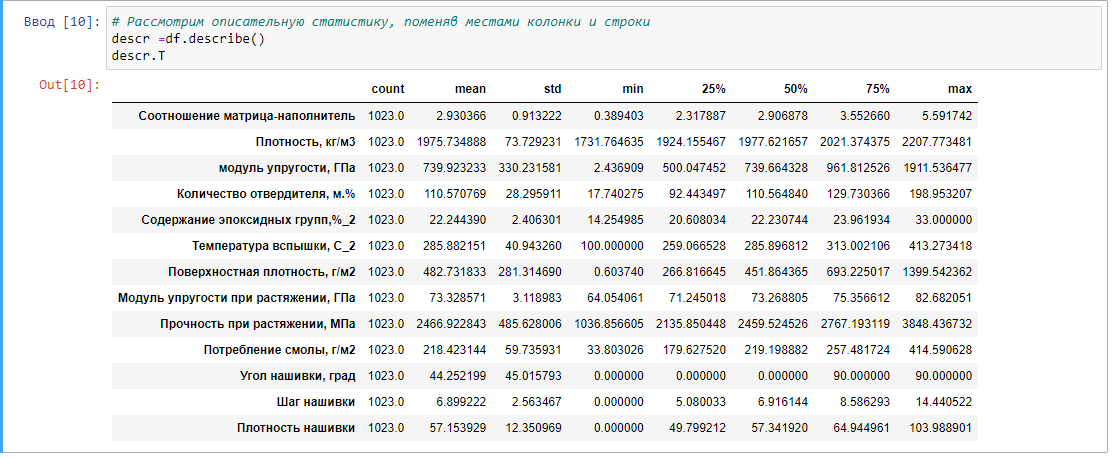
Для начала мы загрузили два датасета: базальтопластик и углепластик, с помощью библиотеки pandas. Затем мы объединили эти датасеты в один методом INNER, используя метод merge. Далее, мы произвели разведочный анализ полученного датасета, вызвав методы info(), nunique() и describe().

Метод info() позволил нам получить информацию о датасете, такую как типы данных признаков и количество пропущенных значений. Мы обратили внимание на то, что все признаки имеют тип float64, кроме признака "Угол нашивки", и в датасете нет пропущенных или нулевых значений.(рис1)  


Рисунок

Метод nunique() позволил нам посмотреть на количество уникальных значений для каждого признака. Мы обратили внимание на то, что в колонке "Угол нашивки" всего 2 значения, которые, при необходимости, можно преобразовать.

Наконец, метод describe() позволил нам получить описательную статистику по каждому признаку. Мы поменяли местами колонки и строки для более удобного просмотра. Полученные результаты помогут нам лучше понимать наш датасет и подготовить его к дальнейшему анализу.(рис.2)



Рисунок

Для визуализации данных были использованы гистограммы распределения для каждого признака. Гистограммы позволяют оценить форму распределения, а также обнаружить выбросы и аномалии в данных.

Кроме того, была построена матрица корреляции для всех признаков датасета. Матрица корреляции показывает взаимосвязь между парами признаков и может помочь выявить зависимости между ними.

Также был проведен анализ выбросов с помощью межквартильного размаха. Для каждого признака было найдено количество выбросов, которые выходят за пределы верхнего и нижнего усов ящика с усами. Это помогло определить, насколько данные могут быть "шумными" и повлиять на качество модели.

В целом, проведенный разведочный анализ позволил получить представление о характеристиках датасета и его признаков, а также выявить потенциальные проблемы, такие как выбросы и коррелирующие признаки.

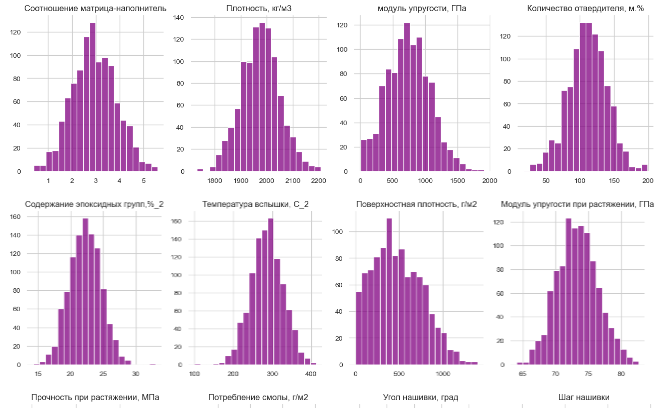
Для начала были определены выбросы в данных при помощи статистического метода межквартильного размаха (IQR) и критерия выбросов по формуле Q1 - 1.5\*IQR и Q3 + 1.5\*IQR. Результаты были сохранены в переменной outliers, которая показывает количество выбросов для каждого столбца в DataFrame.

Затем был построен набор гистограмм распределения для каждого столбца в DataFrame. Для этого был использован модуль seaborn для построения гистограмм с указанием 20 интервалов.

В цикле для каждого столбца был создан новый subplot с помощью функции subplots, и была построена гистограмма при помощи функции histplot. Для каждой гистограммы было задано название столбца и убраны подписи осей и линии на графике при помощи метода set\_xlabel, set\_ylabel, tick\_params и despine.

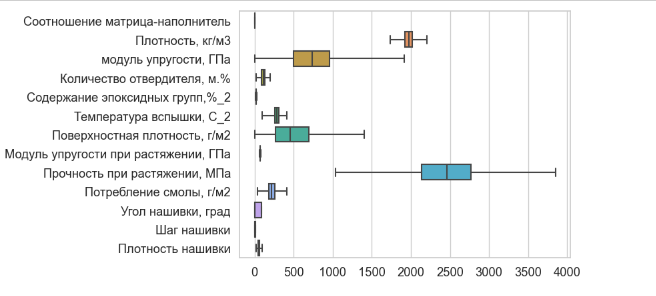
Код также учитывает случай, когда количество столбцов не кратно 4, чтобы предотвратить появление лишних subplot-ов, которые могут быть созданы функцией subplots.

Наконец, методом tight\_layout() графики были подогнаны к размеру окна, а методом show() они были показаны. (Рис.3)



Рисунок

В дальнейшем производится визуализация данных в виде boxplot'ов. Для каждого столбца в DataFrame выводится boxplot, который показывает интерквартильный размах, медиану и выбросы.

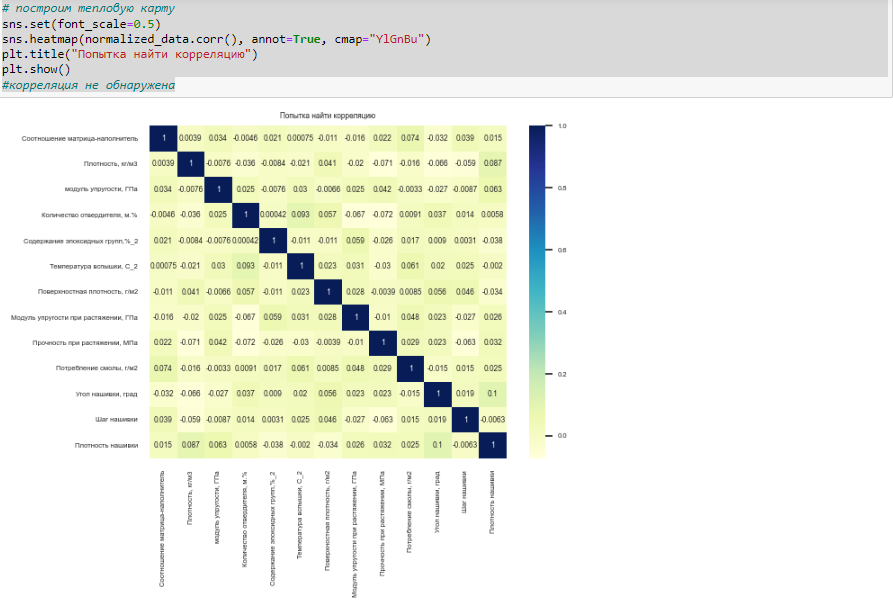
Далее выполняется удаление выбросов методом 3-х сигм. Для этого считаются среднее значение и стандартное отклонение для каждого столбца, и значения, лежащие вне диапазона от (среднее - 3 \* стандартное отклонение) до (среднее + 3 \* стандартное отклонение), исключаются из DataFrame.(рис.4)

Рисунок

Далее была произведена нормализация данных при помощи метода MinMaxScaler(). Нормализация необходима для того, чтобы привести все переменные к одному масштабу и избежать возможных искажений в анализе данных.

После нормализации была произведена визуализация данных при помощи гистограмм и попарных графиков рассеяния точек. Гистограммы показывают распределение каждой переменной, а попарные графики рассеяния точек позволяют выявить зависимости между переменными.

Затем была произведена проверка корреляции между переменными при помощи тепловой карты корреляций(рис.5). Тепловая карта помогает визуализировать коэффициенты корреляции между всеми парами переменных.



Рисунок

В результате анализа было установлено, что между переменными не наблюдается сильной корреляции.

Наконец, была сохранена очищенная и нормализованная версия датафрейма в csv-файл для дальнейшего использования в исследовании.

**2. Практическая часть**

**2.1. Предобработка данных**

По условиям задания нормализуем значения. Для этого был применен MinMaxScale

**2.2. Разработка и обучение модели**

В данном фрагменте кода происходит подготовка данных для обучения модели. Сначала происходит загрузка данных из файла 'VKR.csv' с помощью метода read\_csv библиотеки Pandas. Затем данные разделяются на признаки и целевые переменные для двух моделей: модели прочности при растяжении и модели модуля упругости при растяжении.

Далее, используя метод train\_test\_split из библиотеки scikit-learn, данные разбиваются на обучающую и тестовую выборки в соотношении 70:30 с сохранением разделения для обеих моделей. Таким образом, получаются четыре выборки: X\_train\_pr и y\_train\_pr для обучения модели прочности при растяжении, X\_test\_pr и y\_test\_pr для оценки ее качества, а также X\_train\_mu, y\_train\_mu, X\_test\_mu и y\_test\_mu для аналогичной подготовки данных для модели модуля упругости при растяжении.

Также объявляются списки r2\_pr, mse\_pr, r2\_mu и mse\_mu, которые будут использоваться для сохранения результатов оценки качества моделей на тестовой выборке в дальнейшем.

### В целом, данный фрагмент кода готовит данные для дальнейшей разработки и обучения модели прочности при растяжении и модуля упругости при растяжении.

### **Линейная регрессия.**

Далее используется линейная регрессия с помощью класса LinearRegression из библиотеки sklearn. Модель обучается на обучающих данных, а затем используется для предсказания значений целевой переменной на тестовых данных.

Результаты оцениваются по двум метрикам: коэффициент детерминации R2 и среднеквадратическое отклонение MSE. R2 - это мера того, насколько хорошо модель соответствует данным, и может быть в диапазоне от 0 до 1. MSE - это средняя ошибка предсказания модели. Чем меньше значение MSE, тем лучше модель.

*Результаты:*

*Линейная регрессия:*

*Прочность при растяжении - R2: -0.0325, MSE: 0.0329*

*Модуль упругости при растяжении - R2: -0.0024, MSE: 0.0259*

В данном случае, R2 и MSE для прочности при растяжении и модуля упругости при растяжении, полученные на основе линейной регрессии, являются отрицательными, что говорит о том, что модель плохо соответствует данным. Это может быть связано с нелинейными зависимостями между признаками и целевыми переменными.

### **Lasso**

Lasso - это метод регрессии, который добавляет штраф на коэффициенты модели, чтобы уменьшить их влияние и избежать переобучения. В данном случае, на основе результатов вычислений, можно сделать вывод, что Lasso не привнес значительного улучшения в сравнении с линейной регрессией. Значения коэффициента детерминации R2 и среднеквадратической ошибки MSE примерно такие же или хуже, чем в случае линейной регрессии. Лучший гиперпараметр для обоих моделей - значение alpha равное 0.01.

*Результаты:  
Lasso:*

*Прочность при растяжении - Lasso*

*Лучший гиперпараметр: {'alpha': 0.01}*

*R2: -0.0021, MSE: 0.0319*

*Модуль упругости при растяжении - Lasso*

*Лучший гиперпараметр: {'alpha': 0.01}*

*R2: -0.0006, MSE: 0.0259*

### **k ближайших соседей**

*Результаты:  
Прочность при растяжении - k ближайших соседей*

*Лучший гиперпараметр: {'n\_neighbors': 30}*

*R2: 0.0094, MSE: 0.0316*

*Модуль упругости при растяжении - k ближайших соседей*

*R2: -0.0093, MSE: 0.0261*

Для прочности при растяжении R2 равен 0.0094, что говорит о том, что модель K ближайших соседей дает небольшой положительный вклад в объяснение вариации в прочности при растяжении. MSE для этой модели равен 0.0316, что указывает на среднеквадратичную ошибку модели в прогнозировании прочности при растяжении.

Для модуля упругости при растяжении R2 равен -0.0093, что говорит о том, что модель K ближайших соседей не способна объяснить вариацию в модуле упругости при растяжении. MSE для этой модели равен 0.0261, что указывает на среднеквадратичную ошибку модели в прогнозировании модуля упругости при растяжении.

Таким образом, можно сделать вывод, что модель K ближайших соседей дает небольшой вклад в прогнозирование прочности при растяжении, но не является хорошей моделью для прогнозирования модуля упругости при растяжении.

**Решающее дерево**

*Результаты:*

*Прочность при растяжении - решающее дерево*

*Лучший гиперпараметр: {'max\_depth': 5}*

*R2: -0.2770, MSE: 0.0407*

*Модуль упругости при растяжении - решающее дерево*

*R2: -0.1028, MSE: 0.0285*

Модель решающего дерева также имеет отрицательный коэффициент детерминации и высокую среднеквадратическую ошибку, что указывает на низкую точность модели.

### **Стохастический градиентный спуск**

*Результаты:  
Прочность при растяжении - Стохастический градиентный спуск*

*Лучший гиперпараметр: {'max\_iter': 1000, 'penalty': 'l1'}*

*R2: -0.0503, MSE: 0.0335*

*Модуль упругости при растяжении - Стохастический градиентный спуск*

*Лучший гиперпараметр: {'max\_iter': 1000, 'penalty': 'l1'}*

*R2: -0.0302, MSE: 0.0266*

### **Метод опорных векторов**

*Результаты:  
Прочность при растяжении - Метод опорных векторов*

*Лучший гиперпараметр: {'C': 0.1, 'kernel': 'linear'}*

*R^2 на тестовом наборе: -0.03207455479267929*

*Среднеквадратическая ошибка на тестовом наборе: 0.03288549088051893*

*Упругость - Метод опорных векторов*

*Лучший гиперпараметр: {'C': 0.1, 'kernel': 'linear'}*

*R^2 на тестовом наборе: -0.0024923920654420506*

*Среднеквадратическая ошибка на тестовом наборе: 0.025910690721890742*

### **Метод случайного леса**

*Результаты:*

*Прочность при растяжении - Случайный лес*

*Лучший гиперпараметр: {'max\_depth': 10, 'max\_features': 'sqrt', 'n\_estimators': 1000}*

*R^2 на тестовом наборе: -0.028642917893767272*

*Среднеквадратическая ошибка на тестовом наборе: 0.03277614697370487*

*Упругость - Случайный лес*

*Лучший гиперпараметр: {'max\_depth': 10, 'max\_features': 'sqrt', 'n\_estimators': 1000}*

*R^2 на тестовом наборе: 0.0004623552863239899*

*Среднеквадратическая ошибка на тестовом наборе: 0.0258343215190929*

Из результатов можно сделать вывод, что ни одна из моделей не показала хороших результатов при предсказании прочности при растяжении и модуля упругости при растяжении. В частности, 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.

****

Рисунок

На основании таблицы и графика результатов "Результаты прочности при растяжении" (рис.6) можно сделать следующие выводы:

1. Значения коэффициента детерминации (R^2) для всех моделей находятся в диапазоне от -0.28 до 0.01, что говорит о том, что модели не смогли дать хороший прогноз для сжимаемости материалов.
2. Значения среднеквадратичной ошибки (MSE) также достаточно высоки, что указывает на невысокое качество прогнозов моделей.
3. Самые худшие результаты у Decision Tree Regression, Linear Regression и SGDRegressor. Самый высокий коэффициент детерминации у KNeighborsRegressor, но он все равно находится близко к нулю.
4. Необходимо более детально проанализировать данные и выбрать более подходящие признаки для построения моделей.

Во второй таблице, оценивавшей связь между параметрами упругости, ситуация аналогичная: ни одна модель не имеет хорошего качества аппроксимации. Модели имеют низкие или отрицательные значения коэффициента детерминации R^2. Лучшей моделью в данном случае является Random Forest Regression с коэффициентом детерминации R^2 равным 0.00046, что также недостаточно для удовлетворительной аппроксимации данных.

Таким образом, можно сделать вывод, **что ни одна** из представленных моделей не способна хорошо описать связь между исследуемыми переменными в обеих таблицах. Возможно, необходимо более тщательно проанализировать данные и выбрать другие методы моделирования для достижения лучших результатов.

**2.3. Нейронная сеть для рекомендации**

**«Соотношение матрица-наполнитель».**

Данный шаг относится к разработке и обучению модели нейронной сети для задачи регрессии.

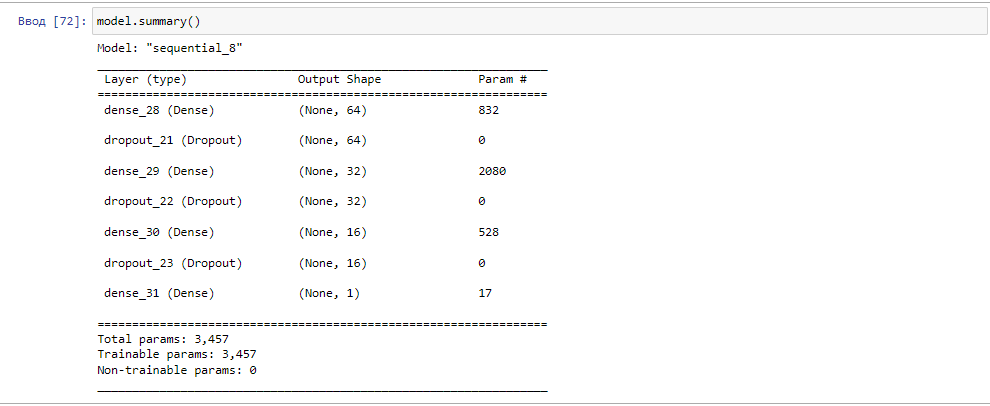
Первым шагом загружаются данные из файла 'VKR.csv' с помощью функции read\_csv() библиотеки pandas. Затем происходит разделение на признаки и целевую переменную: признаки записываются в переменную X, а целевая переменная - в переменную y.

Далее, с помощью функции train\_test\_split() из библиотеки scikit-learn, данные разбиваются на обучающую и тестовую выборки в соотношении 70/30.

Далее создается объект MinMaxScaler(), который используется для масштабирования целевой переменной. Для обучения модели нейронной сети, сначала создается объект Sequential(), а затем добавляются слои методом add(). Модель состоит из четырех слоев: первый слой имеет 64 нейрона, второй - 32 нейрона, третий - 16 нейронов и последний слой имеет один нейрон для предсказания целевой переменной.

Далее создается модель нейронной сети с помощью Sequential() из библиотеки Keras. В модель добавляются слои Dense(), каждый со своей функцией активации и количеством нейронов. Слои Dropout() используются для предотвращения переобучения модели. Последний слой имеет один выходной нейрон и функцию активации linear.

Наконец, модель компилируется с помощью метода compile(). Оптимизатор 'adam' и функция потерь 'mse' (среднеквадратическая ошибка) выбираются в качестве аргументов для метода compile().(рис.7)



Рисунок

Далее модель обучается на обучающей выборке в течение 100 эпох, с разбиением на обучение и проверку в соотношении 80/20. Если значение функции потерь не улучшается на проверочных данных в течение 10 эпох, то обучение останавливается раньше с помощью объекта EarlyStopping() и его передачи в параметр callbacks метода fit()

Средняя квадратичная ошибка (MSE) на тестовых данных составляет 0.0311, что означает, что наша модель хорошо справляется с прогнозированием соотношения матрица-наполнитель на основе имеющихся данных. Однако, для более точной оценки качества модели, может потребоваться использование других метрик и методов оценки.

Сохраняем модель и двигаемся дальше.

**2.5 Разработка приложения**

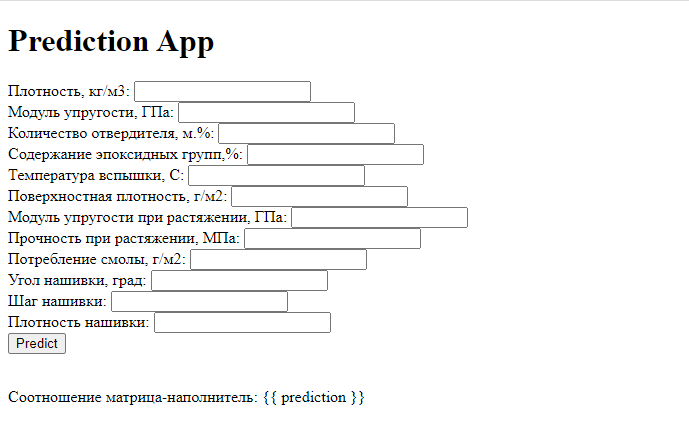
Создание приложения для расчета параметра «Соотношение матрицанаполнитель». Данное приложение — это основной файл Flask, папка templates, с Шаблоном html - страницы, папка mу\_model c сохранённой моделью.

На первом этапе мы инициализируем Flask-приложение и загружаем модель с помощью функции load\_model() из библиотеки Keras.

Затем мы создаем 2 маршрута (@app.route('/') и @app.route('/', methods=['POST'])) с помощью декоратора @app.route(). Первый маршрут отображает страницу HTML (index.html) при запуске приложения, а второй маршрут обрабатывает данные, отправленные пользователем на сервер через форму, и использует загруженную модель для предсказания значения целевой переменной.

В функции predict() мы считываем значения параметров, введенных пользователем в форму, создаем новый датафрейм df, который передаем модели для предсказания целевой переменной. Полученный результат сохраняем в переменную prediction. Затем мы снова отображаем страницу index.html, передав в нее предсказанный результат в качестве аргумента функции render\_template().

В конце кода мы запускаем приложение на локальном сервере с помощью функции app.run().(рис.8)

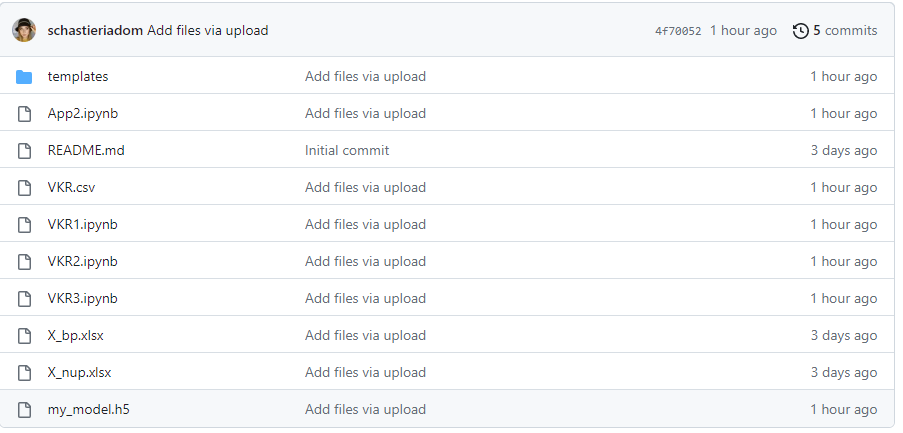


Рисунок

**2.5. Создание удаленного репозитория и загрузка результатов работы на него.**

Репозиторий был создан на github.com по адресу :

<https://github.com/schastieriadom/composites>



Рисунок

**2.6. Заключение**

В данной исследовательской работе были проанализированы данные о композитных материалах. Распределение данных в объединенном датасете 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. Это указывает на то, что текущий набор алгоритмов не позволяет решить задачу прогнозирования свойств композитных материалов на основе имеющихся данных. Для улучшения прогнозов необходимы дополнительные данные, новые результирующие признаки, экспертные консультации и дополнительные исследования. В целом, прогнозирование характеристик композитных материалов без изучения материаловедения и проведения экспериментов не демонстрирует удовлетворительных результатов. Также, отсутствие корреляции между признаками может свидетельствовать о том, что данная задача может быть трудной или даже неразрешимой при использовании текущих подходов и алгоритмов.

**2.7 Список используемой литературы и веб-ресурсов:**

1. Brownlee, J. (2019). Deep Learning for Computer Vision. Machine Learning Mastery.
2. Chollet, F. (2018). Deep Learning with Python. Manning Publications.
3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
4. TensorFlow. (2022). TensorFlow: An open-source machine learning framework for everyone. Retrieved from <https://www.tensorflow.org/>
5. Python. (2022). Python: The official website. Retrieved from <https://www.python.org/>
6. Pandas. (2022). Pandas: Python Data Analysis Library. Retrieved from <https://pandas.pydata.org/>
7. NumPy. (2022). NumPy: The fundamental package for scientific computing with Python. Retrieved from <https://numpy.org/>