**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc130832920)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 4](#_Toc130832921)

[1.1. Постановка задачи 4](#_Toc130832922)

[1.2. Описание используемых методов 5](#_Toc130832923)

[1.3. Разведочный анализ данных 12](#_Toc130832924)

[1.4. Выводы по аналитической части 15](#_Toc130832925)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 16](#_Toc130832926)

[2.1. Обоснование выбора средств разработки 16](#_Toc130832927)

[2.2. Предобработка данных 19](#_Toc130832928)

[2.3. Разработка и обучение модели нейронной сети 25](#_Toc130832929)

[2.4. Тестирование модели нейронной сети 30](#_Toc130832930)

[2.5. Демонстрация работы разработанной системы 32](#_Toc130832931)

[2.6. Выводы по практической части 35](#_Toc130832932)

[ЗАКЛЮЧЕНИЕ 36](#_Toc130832933)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 37](#_Toc130832934)

[ПРИЛОЖЕНИЕ 40](#_Toc130832935)

# **ВВЕДЕНИЕ**

Композиционные материалы – это искусственно созданные материалы, состоящие из нескольких других с четкой границей между ними. 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. На выходе необходимо спрогнозировать ряд конечных свойств получаемых композиционных материалов.

*Актуальность* данной работы состоит в создание прогнозных моделей, которые помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов возможными новыми характеристиками материалов, и цифровыми двойниками новых композитов.

*Объектом* исследования есть композиционные материалы. *Предметом* исследования есть прогнозирование свойств композиционных материалов.

*Цель* работы состоит в разработке модели для прогнозирования свойств композиционных материалов. Для достижения данной цели необходимо решить следующие *задачи*:

1. Выбрать используемые методы прогнозирования.
2. Провести разведочный анализ и предобработку данных.
3. Подобрать гиперпараметры и обучить модели прогнозирования.
4. Протестировать и выбрать лучшую модель.
5. Разработать систему прогнозирования с выбранной моделью.

ВКР состоит из 44 листов и включает 8 таблиц, 13 рисунков, 26 источников и 1 приложение.

# **1. АНАЛИТИЧЕСКАЯ ЧАСТЬ**

## **1.1. Постановка задачи**

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. Таким образом, задача состоит в прогнозировании конечных свойств получаемых композиционных материалов на основе имеющихся начальных свойств компонентов композиционных материалов.

Используемый для решения поставленной задачи датасет имеет 13 переменных и его объем равен 1023. Датасет не имеет пропусков. Методом z-оценки было выявлено 24 выброса, следовательно, объем датасета после удаления выбросов равен 999.

Задача прогнозирования разделена на 2 подзадачи. В первом случае нужно спрогнозировать значения 2 выходных переменных, следовательно, на вход будут подаваться остальные 11 переменных. Во втором случае нужно спрогнозировать значение 1 переменной, следовательно, на вход будут подаваться до 12 переменных.

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

## **1.2. Описание используемых методов**

Для решения поставленной задачи предлагается использовать следующие методы:

1. Линейная регрессия (linear regression)
2. Метод случайного леса (random forest)
3. Метод k-ближайших соседей (k-nearest neighbors algorithm, k-NN)
4. Дерево решений (decision tree)
5. Многослойный перцептрон (multilayer perceptron, MLP)

Данные методы являются наиболее популярными для задачи регрессии. Также выбранные методы поддерживают прогнозирование одновременно нескольких значений, т.е. модель может иметь больше одного выхода, что и нужно для решения поставленной задачи. Рассмотрим каждый выбранный метод более подробно.

**Линейная регрессия**

Обычная линейная регрессия выполняется методом наименьших квадратов. *Метод наименьших квадратов (МНК)* — математический метод, основанный на минимизации суммы квадратов отклонений некоторых функций от экспериментальных входных данных. Он может использоваться для «решения» переопределенных систем уравнений (когда количество уравнений превышает количество неизвестных), для поиска решения в случае обычных (не переопределенных) нелинейных систем уравнений, для аппроксимации точечных значений некоторой функции. МНК является одним из базовых методов регрессионного анализа для оценки неизвестных параметров регрессионных моделей по выборочным данным.

Цель состоит в настройке параметров функции модели для наилучшего соответствия набору данных. Простой набор данных состоит из *n* точек (пар данных) (*xi*, *yi*), *i* = 1, …, *n*, где *xi* является независимой переменной и *yi* является зависимой переменной, значение которой находится путем наблюдения. Модельная функция имеет вид *f*(*x*, *β*), где в векторе содержится *m* регулируемых параметров *β*. Цель состоит в том, чтобы найти значения параметров модели, которые «лучше всего» соответствуют данным. Соответствие модели точке данных измеряется ее остатком, определяемым как разница между наблюдаемым значением зависимой переменной и значением, предсказанным моделью (формула 1):

|  |  |
| --- | --- |
|  | (1) |

Метод наименьших квадратов находит оптимальные значения параметров путем минимизации суммы квадратов невязок, *S* (формула 2):

|  |  |
| --- | --- |
|  | (2) |

В простейшем случае *f*(*xi*, *β*) = *β*,а результатом метода наименьших квадратов является среднее арифметическое входных данных.

Примером модели в двух измерениях является прямая линия. Обозначая *y*-перехват как *β0* и наклон как *β1*, модельная функция определяется выражением *f*(*x*, *β*) =*β0* + *β1x*.

Точка данных может состоять из более чем одной независимой переменной. В самом общем случае в каждой точке данных может быть одна или несколько независимых переменных и одна или несколько зависимых переменных. [1]

**Метод случайного леса**

*Метод случайного леса* — это ансамблевый метод обучения для классификации, регрессии и других задач, который работает путем построения множества деревьев решений во время обучения. Для задач регрессии возвращается среднее значение или средний прогноз отдельных деревьев. [2,3]

Алгоритм обучения для случайных лесов применяет общую технику начальной загрузки или бэггинга к обучающимся деревьям. Сопоставляя обучающий набор *X* = *x1*, …, *xn* с ответами *Y* = *y1*, …, *yn*, повторенный (*B* раз) бэггинг выбирает случайную выборку с заменой обучающего набора и подгоняет деревья к образцам:

Для *b* = *1*, ..., *B*:

1. Выборка с заменой *n* обучающих примеров из *X*, *Y*; назовем их *Xb*, *Yb*.
2. Обучение дерева регрессии (или классификации) *fb* на *Xb*, *Yb*.

После обучения прогнозы для невидимых выборок *x'* могут быть сделаны путем усреднения прогнозов всех отдельных деревьев регрессии по *x'* (формула 3):

|  |  |
| --- | --- |
|  | (3) |

Эта процедура начальной загрузки приводит к повышению производительности модели, поскольку она уменьшает дисперсию модели без увеличения смещения. Это означает, что, хотя прогнозы одного дерева очень чувствительны к шуму в его обучающей выборке, среднее значение многих деревьев нет, если деревья не коррелированы. Простое обучение многих деревьев на одном обучающем наборе даст сильно коррелированные деревья (или даже одно и то же дерево много раз, если алгоритм обучения детерминистический); Начальная выборка — это способ упорядочить деревья, показывая им разные обучающие наборы. [4]

Хотя случайные леса часто обеспечивают более высокую точность, чем одиночное дерево решений, они жертвуют внутренней интерпретируемостью, присущей деревьям решений. Деревья решений относятся к довольно небольшому семейству моделей машинного обучения, которые легко интерпретируются наряду с линейными моделями, моделями на основе правил и моделями на основе внимания. Эта интерпретируемость является одним из наиболее желательных качеств деревьев решений. Это позволяет разработчикам подтвердить, что модель извлекла реалистичную информацию из данных, и позволяет конечным пользователям доверять и доверять решениям, принятым моделью. [5,6]

**Метод k-ближайших соседей**

*Метод k-ближайших соседей* — метрический алгоритм для автоматической классификации объектов или регрессии.

В случае использования метода для регрессии, объекту присваивается среднее значение по *k* ближайшим к нему объектам, значения которых уже известны.

Алгоритм может быть применим к выборкам с большим количеством атрибутов (многомерным). Для этого перед применением нужно определить функцию расстояния; классический вариант такой функции — евклидова метрика.[7,8]

Обучающие примеры представляют собой векторы в многомерном пространстве признаков, каждый из которых имеет метку класса. Фаза обучения алгоритма состоит только из хранения векторов признаков и меток классов обучающих выборок.

На этапе классификации *k* является определяемой пользователем константой, а непомеченный вектор (запрос или контрольная точка) классифицируется путем присвоения метки, наиболее часто встречающейся среди k обучающих выборок, ближайших к этой точке запроса.

Недостаток базовой классификации «мажоритарного голосования» возникает, когда распределение классов асимметрично. То есть примеры более частого класса имеют тенденцию доминировать в предсказании нового примера, потому что они имеют тенденцию быть общими среди *k* ближайших соседей из-за их большого количества. Одним из способов решения этой проблемы является взвешивание классификации с учетом расстояния от контрольной точки до каждого из *k* ее ближайших соседей. Класс (или значение в задачах регрессии) каждой из *k* ближайших точек умножается на вес, пропорциональный обратному расстоянию от этой точки до контрольной точки. Другой способ преодоления перекоса — абстракция в представлении данных. Например, в самоорганизующейся карте (SOM) каждый узел является представителем (центром) кластера подобных точек независимо от их плотности в исходных обучающих данных. Затем k-NN можно применить к SOM. [9]

**Дерево решений**

*Дерево решений* — это инструмент поддержки принятия решений, который использует древовидную модель решений и их возможных последствий, включая исходы случайных событий, затраты ресурсов и полезность. Это один из способов отображения алгоритма, который содержит только операторы условного управления.

Деревья решений обычно используются в исследованиях операций, особенно в анализе решений, чтобы помочь определить стратегию, которая с наибольшей вероятностью приведет к цели, но они также являются популярным инструментом в машинном обучении. [10]

Дерево решений представляет собой структуру, подобную блок-схеме, в которой каждый внутренний узел представляет собой «проверку» атрибута (например, выпадает ли при подбрасывании монеты орел или решка), каждая ветвь представляет собой результат проверки, а каждый конечный узел представляет собой метку класса (решение принимается после вычисления всех атрибутов). Пути от корня к листу представляют собой правила классификации.

Среди инструментов поддержки принятия решений деревья решений (и диаграммы влияния) имеют ряд преимуществ. Деревья решений:

* Простые для понимания и интерпретации. Люди могут понять модели деревьев решений после краткого объяснения.
* Имеют ценность даже при небольшом количестве достоверных данных. Важные идеи могут быть получены на основе описания экспертами ситуации (ее альтернатив, вероятностей и затрат) и их предпочтений в отношении результатов.
* Помогают определить худшие, лучшие и ожидаемые значения для различных сценариев.
* Используют модель белого ящика. Если данный результат предоставляется моделью.
* Может сочетаться с другими методами принятия решений.
* Может учитывать действия более чем одного лица, принимающего решения.

Недостатки деревьев решений:

* Они нестабильны, а это означает, что небольшое изменение данных может привести к большому изменению структуры оптимального дерева решений.
* Часто они относительно неточны. Многие другие предикторы работают лучше с аналогичными данными. Это можно исправить, заменив одно дерево решений случайным лесом деревьев решений, но случайный лес не так легко интерпретировать, как одно дерево решений.
* Для данных, включающих категориальные переменные с разным количеством уровней, прирост информации в деревьях решений смещается в пользу атрибутов с большим количеством уровней. [11]
* Расчеты могут стать очень сложными, особенно если многие значения являются неопределенными и/или если многие результаты связаны между собой.

**Многослойный перцептрон**

*Многослойный персептрон (MLP)* — это полносвязный класс искусственных нейронных сетей с прямой связью (ANN). [12]

MLP состоит как минимум из трех слоев узлов: входного слоя, скрытого слоя и выходного слоя. За исключением входных узлов, каждый узел представляет собой нейрон, использующий нелинейную функцию активации. MLP использует метод обучения с учителем, называемый обратным распространением для обучения. Многослойность и нелинейная активация отличают MLP от линейного персептрона. Он может различать данные, которые не являются линейно разделимыми. [13,14]

MLP состоит из трех или более слоев (входной и выходной слой с одним или несколькими скрытыми слоями) нелинейно-активирующих узлов. Поскольку MLP полностью связаны, каждый узел в одном слое соединяется с определенным весом каждого узла следующего слоя.

Обучение происходит в персептроне путем изменения веса соединения после обработки каждого фрагмента данных в зависимости от количества ошибок в выходных данных по сравнению с ожидаемым результатом.

MLP полезны в исследованиях из-за их способности решать проблемы стохастически, что часто позволяет получить приближенные решения для чрезвычайно сложных задач, таких как аппроксимация пригодности.

MLP являются универсальными аппроксиматорами функций, как показано в теореме Цибенко, поэтому их можно использовать для создания математических моделей с помощью регрессионного анализа. Поскольку классификация является частным случаем регрессии, когда переменная ответа является категориальной, MLP являются хорошими алгоритмами классификатора. [15]

MLP были популярным решением для машинного обучения в 1980-х годах, находя приложения в различных областях, таких как распознавание речи, распознавание изображений и программное обеспечение для машинного перевода, но впоследствии столкнулись с сильной конкуренцией со стороны гораздо более простых машин опорных векторов. Интерес к сетям обратного распространения вернулся благодаря успехам глубокого обучения. [16,17]

Сравним выбранные методы между собой по трем параметрам: количество данных для обучения, скорость и точность метода. Для каждого метода по каждому параметру выставлен балл от 1 (минимальный) до 5 (максимальный). Априорное сравнение методов приведено в таблице 1.

Таблица 1 – Априорное сравнение методов

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Кол-во данных для обучения** | **Скорость** | **Точность** |
| **Линейная регрессия** | **5** | 4 | 4 |
| **Метод случайного леса** | 3 | 2 | 3 |
| **Метод** **k-ближайших соседей** | 3 | **5** | 2 |
| **Дерево решений** | 3 | 3 | 1 |
| **Многослойный перцептрон** | 1 | 1 | **5** |

Из таблицы 1 видно, что меньше всего данных для обучения нужно линейной регрессии, быстрей всех выполняется метод k-ближайших соседей, а самым точным является многослойный перцептрон. [18]

## **1.3. Разведочный анализ данных**

Разведочный анализ данных будет производиться путем анализа корреляции параметров датасета, в особенности корреляция входных параметров и выходных параметров датасета.

*Корреляция* — статистическая взаимосвязь двух или более случайных величин (либо величин, которые можно с некоторой допустимой степенью точности считать таковыми), при этом изменения значений одной или нескольких из этих величин сопутствуют систематическому изменению значений другой или других величин. [19]

Математической мерой корреляции двух случайных величин служит корреляционное отношение *ƞ* либо коэффициент корреляции *R* (или *r*). В случае если изменение одной случайной величины не ведёт к закономерному изменению другой случайной величины, но приводит к изменению другой статистической характеристики данной случайной величины, то подобная связь не считается корреляционной, хотя и является статистической. [19,20]

Значительная корреляция между двумя случайными величинами всегда является свидетельством существования некоторой статистической связи в данной выборке, но эта связь не обязательно должна наблюдаться для другой выборки и иметь причинно-следственный характер. Часто заманчивая простота корреляционного исследования подталкивает исследователя делать ложные интуитивные выводы о наличии причинно-следственной связи между парами признаков, в то время как коэффициенты корреляции устанавливают лишь статистические взаимосвязи. Корреляция двух величин может свидетельствовать о существовании общей причины, хотя сами явления напрямую не взаимодействуют.

В то же время, отсутствие корреляции между двумя величинами ещё не значит, что между ними нет никакой связи. Например, зависимость может иметь сложный нелинейный характер, который корреляция не выявляет.

Некоторые виды коэффициентов корреляции могут быть положительными или отрицательными. В первом случае предполагается, что мы можем определить только наличие или отсутствие связи, а во втором — также и её направление. Если предполагается, что на значениях переменных задано отношение строгого порядка, то отрицательная корреляция — корреляция, при которой увеличение одной переменной связано с уменьшением другой. При этом коэффициент корреляции будет отрицательным. Положительная корреляция в таких условиях — это такая связь, при которой увеличение одной переменной связано с увеличением другой переменной. Возможна также ситуация отсутствия статистической взаимосвязи — например, для независимых случайных величин.

*Коэффициент корреляции Пирсона* – мера линейной корреляции между двумя наборами данных. Это отношение между ковариацией двух переменных и произведением их стандартных отклонений; таким образом, это, по сути, нормализованное измерение ковариации, так что результат всегда имеет значение от -1 до 1. Как и в случае самой ковариации, мера может отражать только линейную корреляцию переменных и игнорировать многие другие типы отношений или корреляции.

Коэффициент корреляции Пирсона рассчитывается по формуле 4:

|  |  |
| --- | --- |
|  | (4) |

где ,  – среднее значение выборок.

Коэффициент корреляции изменяется в пределах от минус единицы до плюс единицы. [20,21]

Между переменными датасета будет рассчитан коэффициент корреляции Пирсона и помещен в корреляционную матрицу.

*Матрица корреляции* — это просто таблица, в которой отображаются коэффициенты корреляции для различных переменных. Матрица отображает корреляцию между всеми возможными парами значений в таблице. Это мощный инструмент для обобщения большого набора данных, а также для выявления и визуализации закономерностей в данных.

Матрица корреляции состоит из строк и столбцов, которые показывают переменные. Каждая ячейка таблицы содержит коэффициент корреляции.

Кроме того, корреляционная матрица часто используется в сочетании с другими типами статистического анализа. Например, это может быть полезно при анализе нескольких моделей линейной регрессии. Модели содержат несколько независимых переменных. В множественной линейной регрессии матрица корреляции определяет коэффициенты корреляции между независимыми переменными в модели. [22]

## **1.4. Выводы по аналитической части**

В результате выполнения данного раздела приведено смысловое описание решаемой задачи. Дана характеристика используемого датасета. Кратко описаны используемые методы, а также представлена таблица априорного сравнения выбранных методов. Представлена формула расчета коэффициента корреляции для построения корреляционной матрицы, которая используется для разведочного анализа данных.

# **2. ПРАКТИЧЕСКАЯ ЧАСТЬ**

## **2.1. Обоснование выбора средств разработки**

Для разработки системы прогнозирования конечных свойств композиционных материалов был выбран язык программирования **Python**, поскольку он обладает следующими преимуществами:

* *Легко читать, изучать и писать*

Python — это язык программирования высокого уровня с английским синтаксисом. Это облегчает чтение и понимание кода.

Python действительно легко понять и изучить. Нужно меньше строк кода для выполнения той же задачи по сравнению с другими основными языками, такими как C/C++ и Java.

* *Повышение производительности*

Python — очень продуктивный язык. Благодаря простоте Python разработчики могут сосредоточиться на решении проблемы. Им не нужно тратить слишком много времени на понимание синтаксиса или поведения языка программирования.

* *Интерпретируемый язык*

Python является интерпретируемым языком, что означает, что Python напрямую выполняет код построчно. В случае какой-либо ошибки он останавливает дальнейшее выполнение и сообщает о возникшей ошибке.

Python показывает только одну ошибку, даже если в программе несколько ошибок. Это упрощает отладку.

* *Динамически типизированный*

Python не знает тип переменной, пока мы не запустим код. Он автоматически присваивает тип данных во время выполнения. Программисту не нужно беспокоиться об объявлении переменных и их типов данных.

* *Бесплатный и с открытым исходным кодом*

Python поставляется под одобренной OSI лицензией с открытым исходным кодом. Это делает его бесплатным для использования и распространения. Вы можете загрузить исходный код, изменить его и даже распространять свою версию Python. Это полезно для организаций, которые хотят изменить определенное поведение и использовать свою версию для разработки.

* *Поддержка обширных библиотек*

Стандартная библиотека Python огромна, вы можете найти почти все функции, необходимые для вашей задачи. Таким образом, вам не нужно зависеть от внешних библиотек.

Но даже если вы это сделаете, диспетчер пакетов Python (pip) упрощает импорт других замечательных пакетов из индекса пакетов Python (PyPi). Он состоит из более чем 200 000 пакетов.

* *Портативность*

Во многих языках, таких как C/C++, нужно изменить код, чтобы запустить программу на разных платформах. Это не то же самое с Python. Python пишут один раз и запускают его где угодно.

Однако нужно быть осторожным, чтобы не включать какие-либо системно-зависимые функции.[23]

Для чтения и работы с данными была выбрана библиотека **Pandas**, поскольку она обладает следующими преимуществами:

* *Представление данных*

Pandas обеспечивает чрезвычайно оптимизированные формы представления данных. Это помогает лучше анализировать и понимать данные. Более простое представление данных способствует лучшим результатам для проектов по науке о данных.

* *Меньше писать и больше работать*

Это одно из лучших преимуществ Pandas. То, что заняло бы несколько строк в Python без каких-либо вспомогательных библиотек, может быть просто достигнуто с помощью 1-2 строк с использованием Pandas. Таким образом, использование Pandas помогает сократить процедуру обработки данных.

* *Обширный набор функций*

Pandas предоставляет огромный набор важных команд и функций, которые используются для легкого анализа данных. Можно использовать Pandas для выполнения различных задач, таких как фильтрация данных в соответствии с определенными условиями или сегментация и разделение данных в соответствии с предпочтениями и т. д.

* *Эффективно обрабатывает большие данные*

Pandas помогают сэкономить много времени, очень быстро импортируя большие объемы данных.

* *Делает данные гибкими и настраиваемыми*

Pandas предоставляют огромный набор функций для применения к имеющимся данным, чтобы иметь возможность настраивать, редактировать и поворачивать их в соответствии с задачей. Это помогает максимально эффективно использовать данные.

* *Сделано для Python*

Программирование на Python стало одним из самых востребованных языков программирования в мире благодаря широкому набору функций и высокой производительности. Таким образом, возможность писать код Pandas на Python позволяет использовать мощь различных других функций и библиотек, которые будут использоваться с Python.[24]

Для вычислений была выбрана библиотека **NumPy**, поскольку она обладает следующими преимуществами:

* *Мощные функции*

В отличие от обычных срезов, срезы NumPy немного мощнее. Обычный метод срезов Python не может реализовать намерение пользователя как NumPy. Кроме того, NumPy может выполнять многомерные срезы, что неудобно в стандартном Python.

* *Эффективное представление данных*

Доступ к массивам NumPy и их создание намного быстрее стандартного Python, при этом они занимают меньше памяти. Уменьшенный объем памяти массива NumPy становится еще более заметным для больших наборов данных.

* *Удобный*

Библиотека NumPy удобна в использовании благодаря большому количеству методов, которые существенно упрощают работу с данными и вычислениями.[25]

В качестве среды разработки была выбрана платформа **Google Colab**, поскольку она обладает следующими преимуществами:

* *Предустановленные библиотеки*

Дистрибутив Anaconda Jupyter Notebook поставляется с несколькими предустановленными библиотеками данных, такими как Pandas, NumPy, Matplotlib. Google Colab, с другой стороны, предоставляет еще больше предустановленных библиотек машинного обучения, таких как Keras, TensorFlow и PyTorch.

* *Сохранение в облаке*

Блокноты Google Colab доступны с любого устройства с помощью простого входа в систему Google. Все блокноты Google Colab сохраняются в учетной записи Google Диска.

* *Сотрудничество*

Еще одна замечательная функция, которую предлагает Google Colab, — это функция совместной работы.

* *Бесплатное использование GPU и TPU*

Google Research позволяет использовать их выделенные графические процессоры и TPU для личных проектов машинного обучения.[26]

## **2.2. Предобработка данных**

Изначально мы имеем 2 файла с данными о свойствах материалов. Содержимое первого файла представлено на рисунке 1.

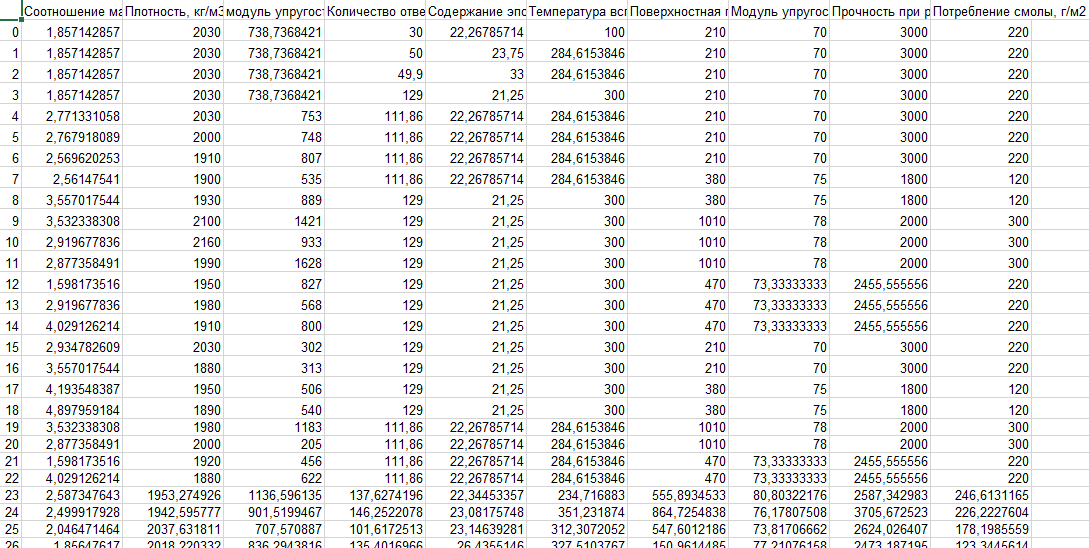


Рисунок 1 – Содержимое первого файла с данными о свойствах материалов

Содержимое второго файла с данными о свойствах материалов представлено на рисунке 2.



Рисунок 2 – Содержимое второго файла с данными о свойствах материалов

Первый столбец в файлах – это индекс материала. Соответственно, чтоб объединить данные двух файлов в один датасет, произведем INNER-объединение по индексу. В результате объединения мы получили исходный датасет, который представлен на рисунке 3.



Рисунок 3 – Исходный датасет

Из рисунка 3 видно, что исходный датасет имеет 1023 строки и 13 столбцов. Строки датасета – это материалы, а столбцы – свойства материалов.

Следующим шагом удаляем пропуски в исходном датасете. Поскольку датасет не имеет пропусков, то в результате удаления пропусков датасет не измениться и останется прежним.

Чтобы определить, есть ли выбросы в используемом датасете, построим коробчатую диаграмму (boxplot) для нашего набора данных. Диаграмма boxplot представлена на рисунке 4.

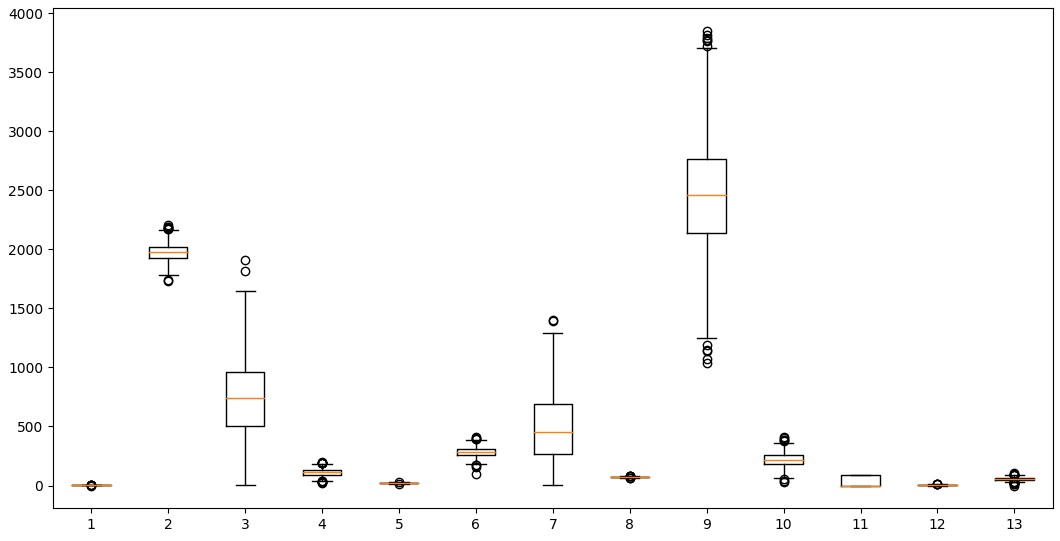


Рисунок 4 – Коробчатая диаграмма (boxplot)

Из рисунка 4 видно, что используемый набор данных действительно содержит выбросы, которые обозначены окружностями на диаграмме. Выбросы необходимо удалить, так как они могут негативно повлиять на обучение и результат прогнозирования моделей машинного обучения.

Следующим шагом удаляем выбросы. Для идентификации и удаления выбросов применим метод *Z*-значения (*Z*-оценки). *Z*-оценка показывает, сколько стандартных отклонений данного значения от среднего. Мы используем следующую формулу для расчета *Z*-значения (формула 5):

|  |  |
| --- | --- |
|  | (5) |

где:

*X* – это одно необработанное значение данных;

*μ* – среднее значение;

*σ* – стандартное отклонение.

Мы определяем наблюдение как выброс, если его *Z*-оценка меньше -3 или больше 3.

В результате удаления выбросов датасет уменьшился до 999 строк. Следовательно, 24 материала были идентифицированы как выбросы.

Далее, чтоб определить, производить нормализацию или нет, оценим признаки путем построения гистограммы распределений по столбцам и гистограммы попарного рассеивания.

Построим гистограмму распределений по столбцам, чтоб оценить диапазоны значений, в которых лежат признаки. Гистограмма распределений по столбцам представлена на рисунке 5.

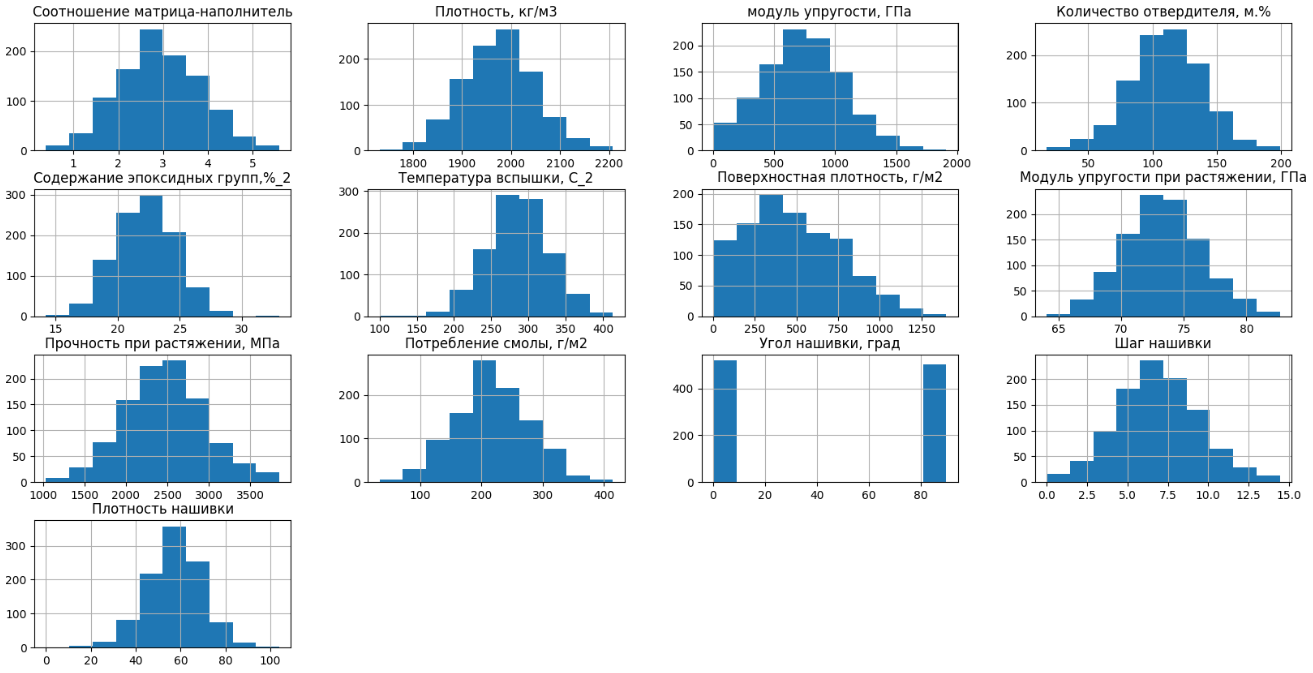


Рисунок 5 – Гистограмма распределений по столбцам

Из рисунка 5 видно, что разные признаки лежат в разных диапазонах значений. Следовательно сравнение признаков между собой будет неточным.

Гистограмма попарного рассеивания представлена на рисунке 6.

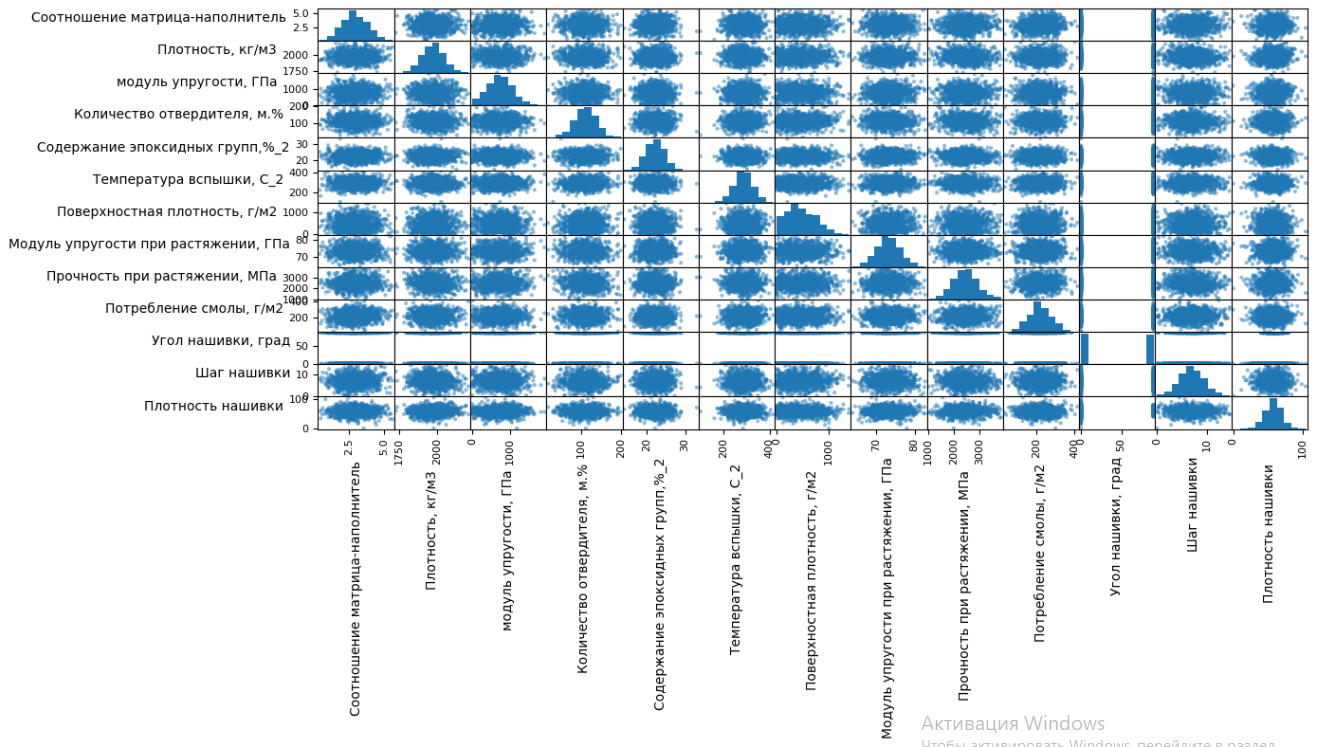


Рисунок 6 – Гистограмма попарного рассеивания

Из рисунка 6 видно, что основная масса большинства признаков лежит в центральной части, следовательно решение удалить выбросы было правильным. Также следует нормализовать данные по столбцам (признакам), чтоб их можно было использовать для обучения моделей машинного обучения.

Ненормированные значения признаков способны быстро свести на нет обучение многих моделей алгоритмов машинного обучения, поскольку веса алгоритмов станут нулевыми либо бесконечными. Чтобы избежать вышеописанных проблем, следует произвести нормализацию входных признаков. Причем нормализация целевых переменных необязательна и не будет произведена, чтоб минимизировать обработку выходных данных моделей.

Далее разделим датасет на входные (X) и выходные (y) свойства материалов. Произведем нормализацию входных свойств к значениям от 0 до 1 отдельно для каждого свойства (столбца). Для проверки корректности результата нормализации снова построим гистограмму распределений. Гистограмма распределений по столбам после нормализации представлена на рисунке 7.

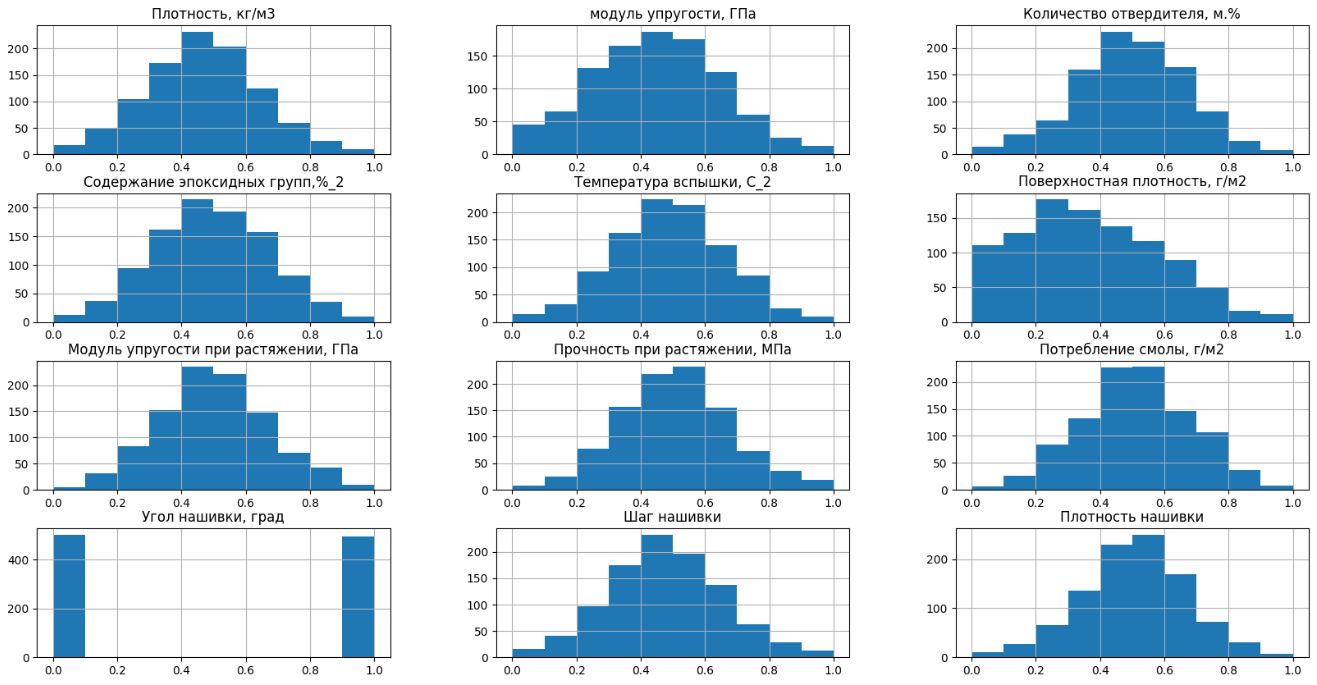


Рисунок 7 – Гистограмма распределений по столбам после нормализации

Из рисунка 7 видно, что теперь все входные признаки лежат в диапазоне от 0 до 1, следовательно нормализация произведена корректно.

Последним шагом разделим датасеты входных (X) и выходных (y) свойств на обучающую и тестовую выборки в соотношении 80% к 20% соответственно.

В результате мы получили входные и выходные обучающие и тестовые датасеты, которые готовы для применения к методам машинного обучения.

## **2.3. Разработка и обучение модели нейронной сети**

Исследуем связь между свойствами материалов путем оценки значений коэффициентов корреляции. Для этого построим матрицу корреляций для всех свойств материалов, которая представлена на рисунке 8.

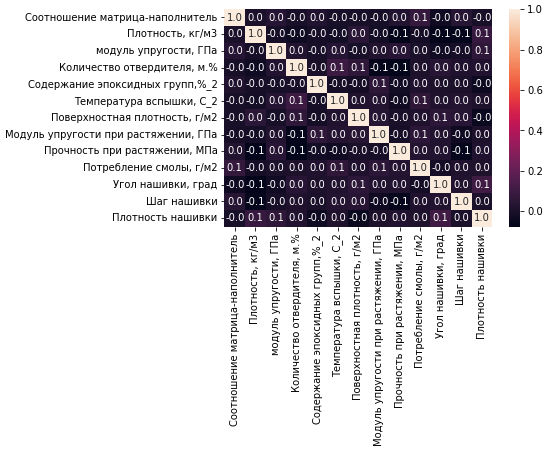


Рисунок 8 – Матрица корреляций для свойств материалов

Из рисунка 4 видно, что коэффициенты корреляции для всех свойств материалов близки к нулю, следовательно корреляционная связь между свойствами слабая либо отсутствует.

Учитывая слабую корреляцию между признаками, можно предположить, что точность прогнозирования свойств композиционных материалов будет низкой. Также во входных выборках будут учитываться все свойства материалов.

Далее при помощи GridSearchCV произведем подбор гиперпараметров для тех используемых методов, где это возможно. Например, метод линейной регрессии (LinearRegression) не требует подбора гиперпараметров, в отличии от остальных методов.

Для метода случайного леса (RandomForestRegressor) определен следующий перечень возможных значений параметров:

parameters = {

'n\_estimators': [10, 100, 1000],

'max\_features': ['sqrt', None],

'max\_depth': [10, 50, 100, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'bootstrap': [True, False],

}

В результате подбора наилучшие результаты метод случайного леса показывает при следующих гиперпараметрах:

{'bootstrap': True,

'max\_depth': 10,

'max\_features': 'sqrt',

'min\_samples\_leaf': 4,

'min\_samples\_split': 10,

'n\_estimators': 1000}

Для метода k-ближайших соседей (KNeighborsRegressor) определен следующий перечень возможных значений параметров:

parameters = {

'n\_neighbors': [2, 5, 10, 20, 50, 100],

'weights': ['uniform', 'distance'],

}

В результате подбора наилучшие результаты метод k-ближайших соседей показывает при следующих гиперпараметрах:

{'n\_neighbors': 100,

'weights': 'uniform'}

Для метода дерево решений (DecisionTreeRegressor) определен следующий перечень возможных значений параметров:

parameters = {

'criterion': ['squared\_error', 'friedman\_mse',

'absolute\_error', 'poisson'],

'max\_features': ['sqrt', None],

'max\_depth': [10, 50, 100, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

}

В результате подбора наилучшие результаты метод дерево решений показывает при следующих гиперпараметрах:

{'criterion': 'squared\_error',

'max\_depth': 10,

'max\_features': 'sqrt',

'min\_samples\_leaf': 1,

'min\_samples\_split': 10}

Для метода многослойный перцептрон (MLPRegressor) определен следующий перечень возможных значений параметров:

parameters = {

'hidden\_layer\_sizes': [(100,), (500,), (1000,)],

'activation': ['logistic', 'tanh', 'relu'],

'batch\_size': [50, 100, 200],

'learning\_rate\_init': [0.01, 0.001, 0.0001],

}

В результате подбора наилучшие результаты метод многослойный перцептрон (нейронная сеть) решений показывает при следующих гиперпараметрах:

{'activation': 'tanh',

'batch\_size': 100,

'hidden\_layer\_sizes': (1000,),

'learning\_rate\_init': 0.001}

Теперь, когда подобраны гиперпараметры, можно переходить к обучению нейронной сети и других используемых методов машинного обучения.

Время обучения для каждого метода приведено в таблице 2.

Таблица 2 – Время обучения для каждого используемого метода

|  |  |
| --- | --- |
| **Метод** | **Время обучения (с)** |
| LinearRegression | 0.008816003799438477 |
| RandomForestRegressor | 2.6903700828552246 |
| KNeighborsRegressor | 0.00585627555847168 |
| DecisionTreeRegressor | 0.007779598236083984 |
| MLPRegressor | 54.39029860496521 |

Из таблицы 2 видно, что дольше всех обучалась нейронная сеть, время обучения которой более 54 секунд. Метод случайного леса обучался чуть меньше 3 секунд. Остальные методы обучались менее 1 секунды.

Схема архитектуры нейронной сети представлена на рисунке 9.

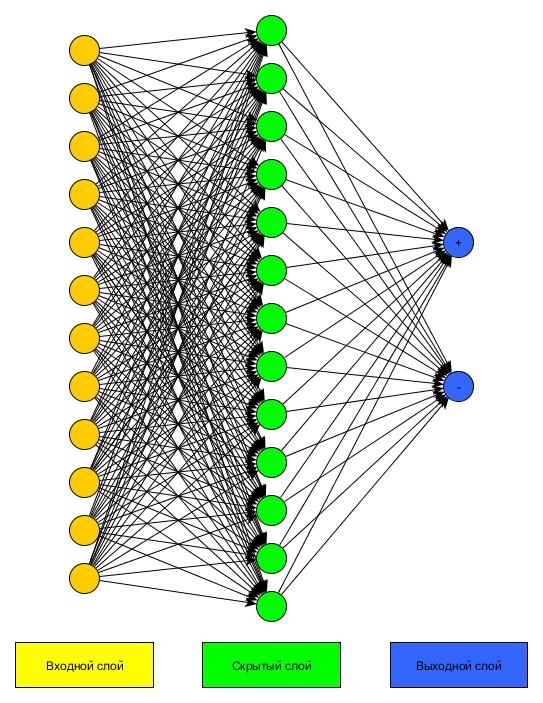


Рисунок 9 – Схема архитектуры нейронной сети

Разработанная нейронная сеть имеет архитектуру, как на рисунке 9, только количество нейронов в скрытом слое равно 1000. Также для первой задачи имеем 11 нейронов во входном слое и 2 нейрона в выходном слое. Для второй задачи имеем 12 нейронов во входном слое и 1 нейрон в выходном слое. Количество нейронов в скрытом слое для обеих задач равно 1000.

## **2.4. Тестирование модели нейронной сети**

Произведем тестирование модели нейронной сети и других используемых методов на обучающей и тестовой выборках. Для каждого метода рассчитаем коэффициент детерминации (score) и среднюю абсолютную ошибку (error) как для тренировочной, так и для тестовой выборки. Значения коэффициента детерминации лежит в промежутке от минус бесконечности до 1 (лучшее значение). Значения средней абсолютной ошибки лежит в промежутке от 0 (лучшее значение) до плюс бесконечности.

Результаты тестирования для метода линейной регрессии представлены в таблице 3.

Таблица 3 – Результаты тестирования для метода линейной регрессии

|  |  |  |
| --- | --- | --- |
| LinearRegression | **score** | **error** |
| **train** | 0.022175596881333415 | 193.1287684333332 |
| **test** | -0.01478235660886884 | 184.47907753749132 |

Результаты тестирования для метода случайного леса представлены в таблице 4.

Таблица 4 – Результаты тестирования для метода случайного леса

|  |  |  |
| --- | --- | --- |
| RandomForestRegressor | **score** | **error** |
| **train** | 0.2994177964241022 | 143.5637189954992 |
| **test** | -0.007779379837577205 | 185.3386270624726 |

Результаты тестирования для метода k-ближайших соседей представлены в таблице 5.

Таблица 5 – Результаты тестирования для метода k-ближайших соседей

|  |  |  |
| --- | --- | --- |
| KNeighborsRegressor | **score** | **error** |
| **train** | 0.015341029177597831 | 193.8081335114808 |
| **test** | -0.010984286735654991 | 183.49840109603653 |

Результаты тестирования для метода дерева решений представлены в таблице 6.

Таблица 6 – Результаты тестирования для метода дерева решений

|  |  |  |
| --- | --- | --- |
| DecisionTreeRegressor | **score** | **error** |
| **train** | 0.24984672744065822 | 148.65199996812856 |
| **test** | -0.35155303738192045 | 212.70959580575686 |

Результаты тестирования для метода многослойного перцептрона (нейронной сети) представлены в таблице 7.

Таблица 7 – Результаты тестирования для метода многослойного перцептрона (нейронной сети)

|  |  |  |
| --- | --- | --- |
| MLPRegressor | **score** | **error** |
| **train** | -0.0012320202977075256 | 194.67380245356216 |
| **test** | -0.006368987106971313 | 182.13201423674877 |

Из таблиц 3-7 видно, что коэффициент детерминации для всех методов на всех выборках близок к нулю, а средняя абсолютная ошибка больше 100. Полученные результаты тестирования свидетельствуют о низкой точности моделей, что объясняется отсутствием существенной корреляционной связи между значениями свойств материалов в датасете.

Поскольку нам важно оценить, насколько точно каждый метод прогнозирует свойства композиционных материалов исходя из имеющихся значений свойств материалов, которые не входят в обучающую выборку, то будем проводить оценку только по результатам, полученным на тестовом датасете. Сравним значения коэффициента детерминации и средней абсолютной ошибки на тестовой выборке для определения лучшего метода, и, соответственно, модели.

Результаты тестирования на тестовой выборке для всех методов собраны в таблице 8.

Таблица 8 – Результаты тестирования на тестовой выборке для всех методов

|  |  |  |
| --- | --- | --- |
| **Метод** | **score** | **error** |
| LinearRegression | -0.01478235660886884 | 184.47907753749132 |
| RandomForestRegressor | -0.007779379837577205 | 185.3386270624726 |
| KNeighborsRegressor | -0.010984286735654991 | 183.49840109603653 |
| DecisionTreeRegressor | -0.35155303738192045 | 212.70959580575686 |
| MLPRegressor | **-0.006368987106971313** | **182.13201423674877** |

Из таблицы 8 видно, что наилучшие значения коэффициента детерминации и средней абсолютной ошибки на тестовой выборке показывает метод многослойного перцептрона (нейронная сеть). Следовательно, именно нейронная сеть (многослойный перцептрон) будет использоваться для разработки системы прогнозирования конечных свойств новых композиционных материалов.

## **2.5. Демонстрация работы разработанной системы**

Разработанная система прогнозирования конечных свойств композиционных материалов представлена в виде блокнота Google Colab, и может быть открыта и запущена из любого современного браузера.

Сперва нужно загрузить в сессионное хранилище Google Colab файлы с сохраненными мин-макс значениями входных данных и обученными моделями нейронной сети. Результат загрузки необходимых файлов в сессионное хранилище Google Colab представлен на рисунке 10.

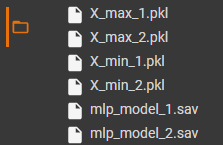


Рисунок 10 – Результат загрузки необходимых файлов в Google Colab

После загрузки необходимых файлов, нужно запустить на выполнение последний блок кода в разделе «Система прогнозирования конечных свойств композиционных материалов».

Далее нужно подготовить excel-файл с входными данными. В файле должно быть минимум 11 столбцов для первой задачи и минимум 12 столбцов для второй задачи. Пример excel-файла входных данных на рисунке 11.

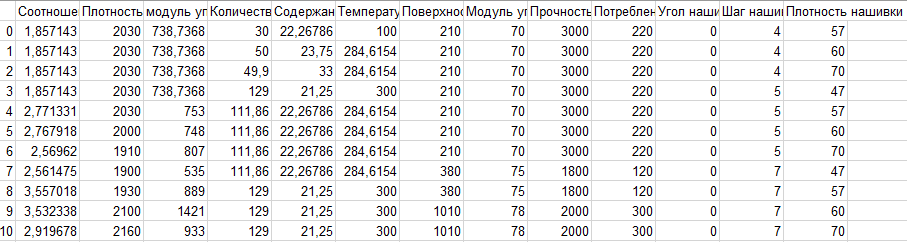


Рисунок 11 – Пример excel-файла входных данных

Разработанная система сама запросит файл входных данных, который нужно выбрать, нажатием на соответствующую кнопку. Также система представит краткое описание двух задач, и попросит выбрать задачу, введя номер задачи. По умолчанию выбирается первая задача. После выбора задачи, система прогнозирует значения свойств новых композиционных материалов и выводит их пользователю в консоль. Взаимодействие системы с пользователем представлено на рисунке 12.

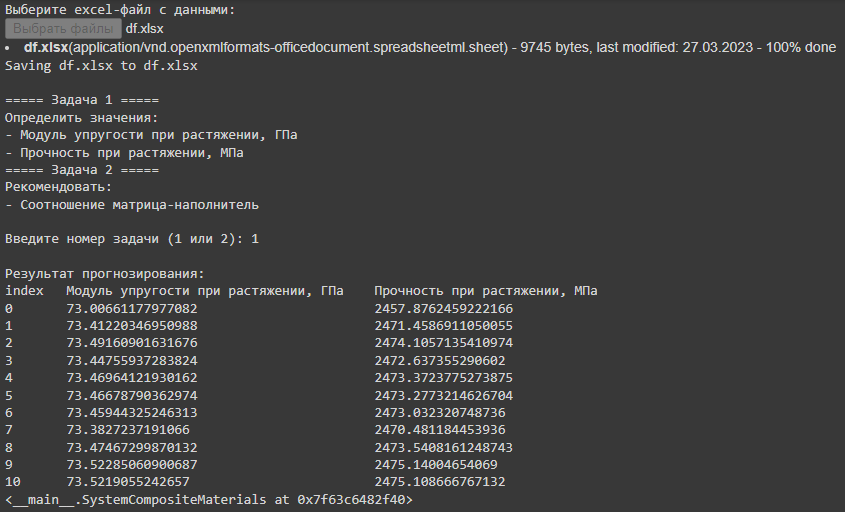


Рисунок 12 – Взаимодействие системы с пользователем

Результат прогнозирования свойств новых композиционных материалов также сохраняется в excel-файл (рисунок 13).



Рисунок 13 – Результат прогнозирования свойств в excel-файле

Полный код чтения и предобработки данных, подбора гиперпараметров для каждого метода, обучения моделей, тестирования моделей и разработанной системы прогнозирования свойств новых композиционных материалов приведен в Приложении.

## **2.6. Выводы по практической части**

В данном разделе обоснован выбор средств разработки, описаны методы предобработки данных, приведена подробная информация о процессе разработки и обучения моделей прогнозирования, представлен результат тестирования и определен лучший метод – нейронная сеть (многослойный перцептрон). В конце раздела имеется инструкция по использованию разработанной системы, а также продемонстрирована ее работа.

# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения ВКР, представлено смысловое описание решаемой задачи, а также описан используемый датасет, что позволило глубже погрузиться в тему. Описаны 5 выбранных методов, с помощью которых предлагается решать задачу прогнозирования свойств материалов. Разведочный анализ данных показал отсутствие существенной корреляционной связи между признаками, из чего можно предположить, что точность прогнозирования будет не высокой. Приведенное обоснование выбора средств разработки, подтверждает корректность выбора данных средств. Предобработка данных позволила увеличить точность результатов прогнозирования, и была описана в работе. Также в процессе выполнения был осуществлен подбор гиперпараметров и обучения моделей, что позволило эффективно использовать модели. Тестирование моделей показало, что наилучшим методом для данной задачи и данных является многослойный перцептрон (нейронная сеть). Выбранный метод и его обученная модель была встроена в систему прогнозирования свойств материалов. Инструкция и демонстрация работы разработанной системы также приведена в ВКР.

В результате выполнения ВКР, была разработана система прогнозирования свойств новых композиционных материалов. Поставленная во введении цель достигнута, все задачи решены.

# **БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

1. Dekking F. M. et al. A Modern Introduction to Probability and Statistics: Understanding why and how. – London : Springer, 2005. – Т. 488.
2. Ho T. K. Random decision forests //Proceedings of 3rd international conference on document analysis and recognition. – IEEE, 1995. – Т. 1. – С. 278-282.
3. Piñeres-Melo M. A. et al. SSwWS: structural model of information architecture //Advances in Swarm Intelligence: 10th International Conference, ICSI 2019, Chiang Mai, Thailand, July 26–30, 2019, Proceedings, Part II 10. – Springer International Publishing, 2019. – С. 400-410.
4. James G. et al. An introduction to statistical learning. – New York : springer, 2013. – Т. 112. – С. 18.
5. Hastie T. et al. The elements of statistical learning: data mining, inference, and prediction. – New York : springer, 2009. – Т. 2. – С. 1-758.
6. Madeh Piryonesi S., El-Diraby T. E. Using machine learning to examine impact of type of performance indicator on flexible pavement deterioration modeling //Journal of Infrastructure Systems. – 2021. – Т. 27. – №. 2. – С. 04021005.
7. Piryonesi S. M., El-Diraby T. E. Role of data analytics in infrastructure asset management: Overcoming data size and quality problems //Journal of Transportation Engineering, Part B: Pavements. – 2020. – Т. 146. – №. 2. – С. 04020022.
8. Hastie T. et al. The elements of statistical learning: data mining, inference, and prediction. – New York : springer, 2009. – Т. 2. – С. 1-758.
9. Burgot G., Auffret F., Burgot J. L. Determination of acetaminophen by thermometric titrimetry //Analytica chimica acta. – 1997. – Т. 343. – №. 1-2. – С. 125-128.
10. Von Winterfeldt D., Edwards W. Decision analysis and behavioral research. – 1993.
11. Deng H., Runger G., Tuv E. Bias of importance measures for multi-valued attributes and solutions //Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part II 21. – Springer Berlin Heidelberg, 2011. – С. 293-300.
12. Hastie T. et al. The elements of statistical learning: data mining, inference, and prediction. – New York : springer, 2009. – Т. 2. – С. 1-758.
13. Rosenblatt F. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. – Cornell Aeronautical Lab Inc Buffalo NY, 1961.
14. Rumelhart D. E., Hinton G. E., Williams R. J. Learning internal representations by error propagation. – California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
15. Cybenko G. Approximation by superpositions of a sigmoidal function //Mathematics of control, signals and systems. – 1989. – Т. 2. – №. 4. – С. 303-314.
16. Wasserman P. D., Schwartz T. Neural networks. II. What are they and why is everybody so interested in them now? //IEEE expert. – 1988. – Т. 3. – №. 1. – С. 10-15.
17. Collobert R., Bengio S. Links between perceptrons, MLPs and SVMs //Proceedings of the twenty-first international conference on Machine learning. – 2004. – С. 23.
18. Comparative Study on Classic Machine learning Algorithms [Электронный ресурс]. – Режим доступа: <https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222>
19. Шмойлова Р. А. Общая теория статистики //М.: Финансы и статистика. – 2002.
20. Елисеева И. и др. Общая теория статистики. – 2008.
21. Гмурман В. Е., Гмурман В. В., Колосова Т. В. Теория вероятностей и математическая статистика. – Общество с ограниченной ответственностью Издательство ЮРАЙТ, 2011. – С. 479-479.
22. Correlation Matrix [Электронный ресурс]. – Режим доступа: <https://corporatefinanceinstitute.com/resources/excel/correlation-matrix/>
23. Python Advantages and Disadvantages – Step in the right direction [Электронный ресурс]. – Режим доступа: https://techvidvan.com/tutorials/python-advantages-and-disadvantages/
24. 6 Essential Advantages of Pandas Library – Why Python Pandas are Popular? [Электронный ресурс]. – Режим доступа: https://data-flair.training/blogs/advantages-of-python-pandas/
25. [NumPy vs Python] What are Advantages of NumPy Arrays over Regular Python Lists? [Электронный ресурс]. – Режим доступа: https://blog.finxter.com/what-are-advantages-of-numpy-over-regular-python-lists/
26. 4 Reasons Why You Should Use Google Colab for Your Next Project [Электронный ресурс]. – Режим доступа: https://towardsdatascience.com/4-reasons-why-you-should-use-google-colab-for-your-next-project-b0c4aaad39ed

# **ПРИЛОЖЕНИЕ**

import time

import pickle

import numpy as np

import pandas as pd

import scipy.stats as stats

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import GridSearchCV

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.neural\_network import MLPRegressor

X\_bp = pd.read\_excel('X\_bp.xlsx', index\_col=0, header=0)

X\_nup = pd.read\_excel('X\_nup.xlsx', index\_col=0, header=0)

df = X\_bp.join(X\_nup, how='inner')

df.dropna(inplace=True) # Удаление пропусков

# Boxplot

fig = plt.figure(figsize=(10, 5))

ax = fig.add\_axes([0, 0, 1, 1])

bp = ax.boxplot(df)

plt.show()

# Гистограммы распределения по столбцам

df.hist(figsize=(20, 10))

# Гистограммы попарного рассеивания

axes = pd.plotting.scatter\_matrix(df, figsize=(14, 8))

for ax in axes.flatten():

ax.xaxis.label.set\_rotation(90)

ax.yaxis.label.set\_rotation(0)

ax.yaxis.label.set\_ha('right')

plt.tight\_layout()

plt.gcf().subplots\_adjust(wspace=0, hspace=0)

plt.show()

# Удаление выбросов (метод Z-оценки)

z = np.abs(stats.zscore(df))

df = df[(z<3).all(axis=1)]

sns.heatmap(df.corr(), annot=True, fmt='.1f')

X = df.drop(['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа'], axis=1)

y = df[['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа']]

# X = df.drop(['Соотношение матрица-наполнитель'], axis=1)

# y = df[['Соотношение матрица-наполнитель']]

# сохранение мин-макс значений входных данных

# X.min().to\_pickle('X\_min\_1.pkl')

# X.max().to\_pickle('X\_max\_1.pkl')

# X.min().to\_pickle('X\_min\_2.pkl')

# X.max().to\_pickle('X\_max\_2.pkl')

X = (X - X.min()) / (X.max() - X.min()) # нормализация

X.hist(figsize=(20, 10))

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

parameters = {

'n\_estimators': [10, 100, 1000],

'max\_features': ['sqrt', None],

'max\_depth': [10, 50, 100, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

'bootstrap': [True, False],

}

clf = GridSearchCV(RandomForestRegressor(random\_state=42), parameters).fit(X\_train, y\_train)

print(clf.best\_params\_)

parameters = {

'n\_neighbors': [2, 5, 10, 20, 50, 100],

'weights': ['uniform', 'distance'],

}

clf = GridSearchCV(KNeighborsRegressor(), parameters).fit(X\_train, y\_train)

print(clf.best\_params\_)

parameters = {

'criterion': ['squared\_error', 'friedman\_mse', 'absolute\_error', 'poisson'],

'max\_features': ['sqrt', None],

'max\_depth': [10, 50, 100, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4],

}

clf = GridSearchCV(DecisionTreeRegressor(random\_state=42), parameters).fit(X\_train, y\_train)

print(clf.best\_params\_)

parameters = {

'hidden\_layer\_sizes': [(100,), (500,), (1000,)],

'activation': ['logistic', 'tanh', 'relu'],

'batch\_size': [50, 100, 200],

'learning\_rate\_init': [0.01, 0.001, 0.0001],

}

clf = GridSearchCV(MLPRegressor(max\_iter=1000, random\_state=42), parameters).fit(X\_train, y\_train)

print(clf.best\_params\_)

regs = {

'LinearRegression': LinearRegression(),

'RandomForestRegressor': RandomForestRegressor(bootstrap=True, max\_depth=10, max\_features='sqrt', min\_samples\_leaf=4, min\_samples\_split=10, n\_estimators=1000, random\_state=42),

'KNeighborsRegressor': KNeighborsRegressor(n\_neighbors=100, weights='uniform'),

'DecisionTreeRegressor': DecisionTreeRegressor(criterion='squared\_error', max\_depth=10, max\_features='sqrt', min\_samples\_leaf=1, min\_samples\_split=10, random\_state=42),

'MLPRegressor': MLPRegressor(max\_iter=1000, activation='tanh', batch\_size=100, hidden\_layer\_sizes=(1000,), learning\_rate\_init=0.001, random\_state=42),

}

for k, v in regs.items():

start\_time = time.time()

regs[k].fit(X\_train, y\_train)

end\_time = time.time()

print(f'{k}:\t{end\_time - start\_time}')

def getAbsoluteError(X, y, reg):

pred = reg.predict(X)

return np.mean(np.absolute(np.array(y) - np.array(pred)))

for k, v in regs.items():

train\_score = regs[k].score(X\_train, y\_train)

test\_score = regs[k].score(X\_test, y\_test)

train\_error = getAbsoluteError(X\_train, y\_train, v)

test\_error = getAbsoluteError(X\_test, y\_test, v)

print(f'===== {k} =====\ntrain\_score:\t{train\_score}\ntest\_score:\t{test\_score}\ntrain\_error:\t{train\_error}\ntest\_error:\t{test\_error}\n')

# сохранение обученных моделей на диск

# pickle.dump(regs['MLPRegressor'], open('mlp\_model\_1.sav', 'wb'))

# pickle.dump(regs['MLPRegressor'], open('mlp\_model\_2.sav', 'wb'))

import pickle

import numpy as np

import pandas as pd

from sklearn.neural\_network import MLPRegressor

from google.colab import files

class SystemCompositeMaterials():

def \_\_init\_\_(self):

print('Выберите excel-файл с данными:')

self.df = pd.read\_excel(list(files.upload().keys())[0], index\_col=0, header=0)

self.task\_num = 1

self.X = self.df.drop(['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа'], axis=1)

self.reg = pickle.load(open('mlp\_model\_1.sav', 'rb'))

self.input\_task\_num()

def input\_task\_num(self):

print(f'\n===== Задача 1 =====\nОпределить значения:\n- Модуль упругости при растяжении, ГПа\n- Прочность при растяжении, МПа\n===== Задача 2 =====\nРекомендовать:\n- Соотношение матрица-наполнитель\n')

try:

if int(input('Введите номер задачи (1 или 2): ')) == 2:

self.task\_num = 2

self.X = self.df.drop(['Соотношение матрица-наполнитель'], axis=1)

self.reg = pickle.load(open('mlp\_model\_2.sav', 'rb'))

except:

pass

self.normalize()

def normalize(self):

X\_min = pd.read\_pickle(f'X\_min\_{self.task\_num}.pkl')

X\_max = pd.read\_pickle(f'X\_max\_{self.task\_num}.pkl')

self.X = (self.X - X\_min) / (X\_max - X\_min)

self.predict()

def predict(self):

predicts = self.reg.predict(self.X)

print('\nРезультат прогнозирования:')

if self.task\_num == 1:

pd.DataFrame(predicts, columns=['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа']).to\_excel(f'predicts\_for\_task\_{self.task\_num}.xlsx', )

print('index', 'Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа', sep='\t')

for i in range(predicts.shape[0]):

print(f'{i}\t{predicts[i][0]}\t\t\t{predicts[i][1]}')

else:

pd.DataFrame(predicts, columns=['Соотношение матрица-наполнитель']).to\_excel(f'predicts\_for\_task\_{self.task\_num}.xlsx')

print('index', 'Соотношение матрица-наполнитель', sep='\t')

for i in range(predicts.shape[0]):

print(i, predicts[i], sep='\t')

SystemCompositeMaterials()