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 Матиенко Евгений Александрович

Москва, 2022

1. Введение 3
2. Аналитическая часть 5
2.1. Постановка задачи 5
2.2. Описание используемых методов 6
2.2.1. Линейная регрессия. 6
2.2.2. Метод k-ближайших соседей 8
2.2.3. Случайный лес.. 10
2.2.4 Нейронная сеть, персептрон 12
2.3. Разведочный анализ данных 16
3. Практическая часть 21
3.1. Предобработка данных 21
3.2. Разработка и обучение моделей 23
3.3. Модель нейронной сети для параметра матрица-наполнитель 28
3.4. Проверка работоспособности модели 31
3.5. Создание удаленного репозитория и загрузка результатов работы на него 32
Заключение 33
Библиографический список 34

1. Введение

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

Проведены анализ и исследования предоставленных данных с использованием методов, изученных на курсе «Data Science», а также выполнена попытка построения различных моделей предсказания свойств композиционного материала по заданным параметрам.

Композит - это часто используемое сокращение от фразы "композитный материал". Подхода заключается в том, что можно взять достоинства одного материала, объединить с достоинствами другого материала, и изготовить изделие, наделенное преимуществами и одного, и другого материалов сразу. Остается только лишь понять, как объединить разнородные материалы в единое целое, чтобы они не разваливались на ходу. Ведь часто требуется соединить принципиально разные материалы, которые по определению не могут работать совместно и не совместимы на химическом уровне.

Классический композитный материал изготавливается из совершенно разных компонентов и объединить можно материалы с диаметрально противоположными свойствами. Например, если мы имеем хрупкий материал, то объединив его с материалом упругим, мы можем получить сразу несуществующий ранее образец с феноменальными свойствами. Упругий материал можно вплавить в виде сетки в толщу основного материала и тем самым армировать эту конструкцию. В итоге мы сохраним высокую твердость хрупкого материала и получим рост пластичности композита.

Объединять можно самые разные группы материалов. Органические и неорганические, металлы и неметаллы, полимеры и мономеры, полимеры и воздух, и все возможные и невозможные сочетания, которые сложно даже

себе представить. Этих материалов не обязательно должно быть два. В общей конструкции можно объединить сразу множество материалов, если получится сделать это физически.

Традиционно композиты имеют такие структурные элементы, как матрица и набивка. В разных источниках их называют по-разному, но смысл от этого не меняется.

Матрицей принято называть материал, внутри которого располагается набивка. В случае железобетона матрицей является бетон. Характерной особенностью является то, что дополняющие материал или материалы как будто плавают в материале матрицы.

Набивка или наполнитель - это те компоненты, которые призваны усовершенствовать свойства. В случае железобетона - это арматура из металла, пронизывающая весь материал насквозь и придающая большую пластичность. Не обязательно, что это один вид материала и не обязательно, что используется он в единственном числе. Наряду с волокнами, в материал могут быть добавлены некоторые дисперсные частицы.

2. Аналитическая часть.

2.1 Постановка задачи.

В данной работе исследуется композит с матрицей из базальтопластика и нашивками из углепластика. Нами был получен датасет, содержащий данные о свойствах матрицы и наполнителя, производственных параметрах и свойствах готового композита. В ходе выполнения работы требуется разработать модели, прогнозирующие значения некоторых свойств в зависимости от остальных. Так же требуется разработать приложение, делающее удобным использование данных моделей специалистом предметной области.

Датасет состоит из двух файлов: X_bp (составляющая из базальтопластика) и X_nup (составляющая из углепластика).

Файл X_bp содержит:

- признаков: 10 и индекс;
- строк: 1023.

Файл X_nup содержит:

- признаков: 3 и индекс;
- строк: 1040.

Известно, что файлы требуют объединения с типом INNER по индексу. После объединения часть строк из файла X_nup была отброшена. И дальнейшие исследования проводим с объединенным датасетом, содержащим 13 признаков и 1023 строк или объектов.

На выходе необходимо спрогнозировать такие конечные свойства получаемых композиционных материалов как модуль упругости при растяжении и прочность при растяжении. Кроме того, необходимо написать нейронную сеть, которая будет рекомендовать соотношение «матрица-наполнитель».

Для решения задачи прогнозирования значений выходных переменных требуется провести предобработку данных, разведочный анализ, построить и сравнить модели прогноза.

2.2 Описание используемых методов

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

В настоящее время разработано много методов регрессионного анализа. Например, простая и множественная линейная регрессия. Эти модели являются параметрическими в том смысле, что функция регрессии определяется конечным числом неизвестных параметров, которые оцениваются на основе данных.

2.2.1 Линейная регрессия

Линейная регрессия (Linear regression) — это математическая модель, которая описывает связь нескольких переменных. Модели линейной регрессии представляют собой статистическую процедуру, помогающую прогнозировать будущее. Она применяется в научных сферах и в бизнесе, а в последние десятилетия используется в машинном обучении.

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

Возьмем небольшой набор данных. Предположим, что это группа коттеджей, расположенных в одном районе. На оси X обозначена их площадь, а на оси Y — рыночная стоимость. Чтобы увидеть, как стоимость дома зависит от его площади, построим регрессию.

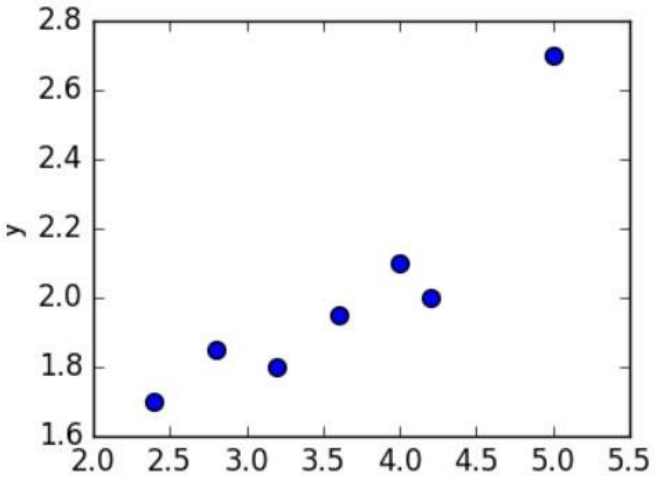

Рисунок 1 – Набор данных для построения регрессии

Это будет простая линейная регрессия с одной переменной. Изменится площадь дома — изменится и стоимость. Для вычисления используем стандартное уравнение регрессии: f (x) = b + m·x, где m — это наклон линии, а b — ее сдвиг по оси Y. То есть изменение коэффициентов m и b будет влиять на расположение прямой:

- если изменить m — прямая наклонится сильнее влево или вправо;
- если изменить b — прямая сместится вверх или вниз по оси Y.

Провести прямую линию через все точки на графике не получится, если они расположены в хаотичном порядке. Поэтому с помощью линейной регрессии определяется оптимальный вариант расположения этой прямой. Некоторые точки все равно останутся на расстоянии, но оно должно быть минимальным. Расчет этого минимального расстояния от прямой до каждой точки называется функцией потерь.

С помощью функции потерь вычисляется минимальное расстояние между объектами и прямой.

Для оценки точности регрессии используют разные метрики, например MSE (от англ. mean squared error — средняя квадратическая ошибка). Чем ниже MSE, тем лучше модель.

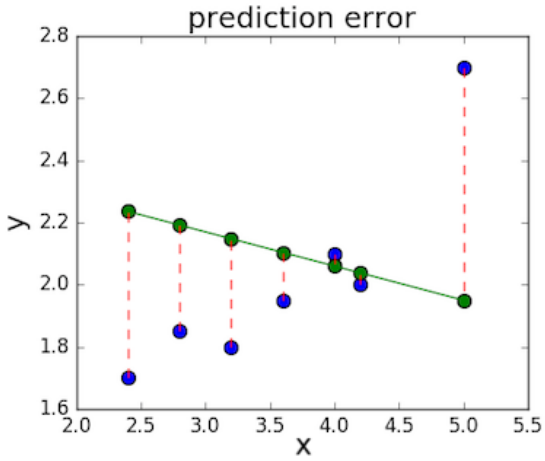

Рисунок 2 – Функция потерь

В первом случае MSE будет равна 0,17, во втором — 0,08, а в третьем — 0,02. Получается, что третья прямая лучше всего показывает зависимость цены дома от его площади.

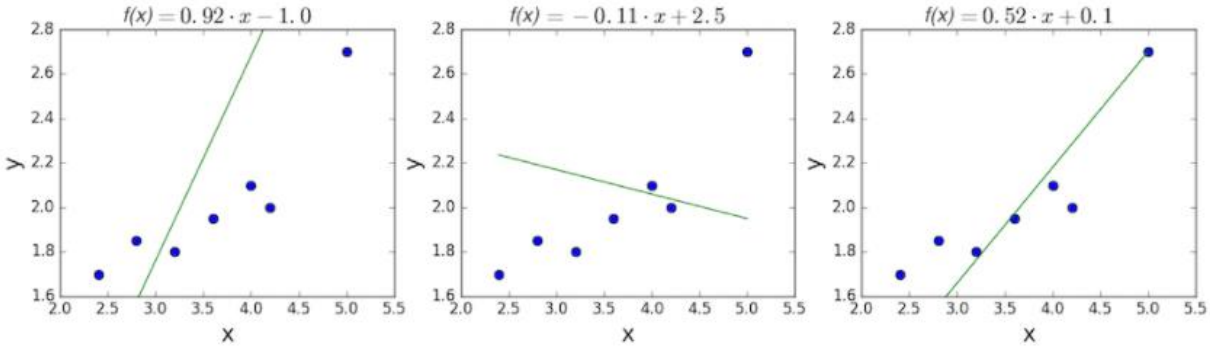

Рисунок 3 – Примеры построения линейной регресии

2.2.2 Метод k-ближайших соседей

Метод K-ближайшего соседа (англ.: k-nearest neighbors method, k-NN) – один из методов решения задачи классификации.

Предполагается, что уже имеется какое-то количество объектов с точной классификацией (т.е. для каждого них точно известно, какому классу он принадлежит). Нужно выработать правило, позволяющее отнести новый объект к одному из возможных классов (т.е. сами классы известны заранее).

В основе k-NN лежит следующее правило: объект считается принадлежащим тому классу, к которому относится большинство его ближайших соседей. Под «соседями» здесь понимаются объекты, близкие к исследуемому в том или ином смысле.

Заметим, что здесь необходимо уметь определять, насколько объекты близки друг к другу, т.е. уметь измерять «расстояние» между объектами. Это не обязательно евклидово расстояние. Это может быть мера близости объектов, например,

по цвету, форме, вкусу, запаху, интересам (если речь идёт о формировании групп людей), особенностям поведения и т.д. Следовательно, для применения метода kNN в пространстве признаков объектов должна быть введена некоторая метрика (т.е. функция расстояния).

Предполагается, что объекты с близкими значениями одних признаков будут близки и по другим признакам (т.е. относиться к одному и тому же классу).

Предположим теперь, что нам предложен новый продукт, и мы должны определить, к какому классу он относится.

Согласно методу k-NN мы отнесём его к тому классу, к которому принадлежит большинство из k его ближайших соседей. Расстояние между объектами будем понимать в смысле Евклидовой нормы, т.е. расстояние между объектами с координатами $(x1,\ y1)$ и $(x2,\ y2)$ равно $\sqrt{((\mathrm{x1}-\mathrm{y1})^2+(\mathrm{x2}-\mathrm{y2})^2)}$.

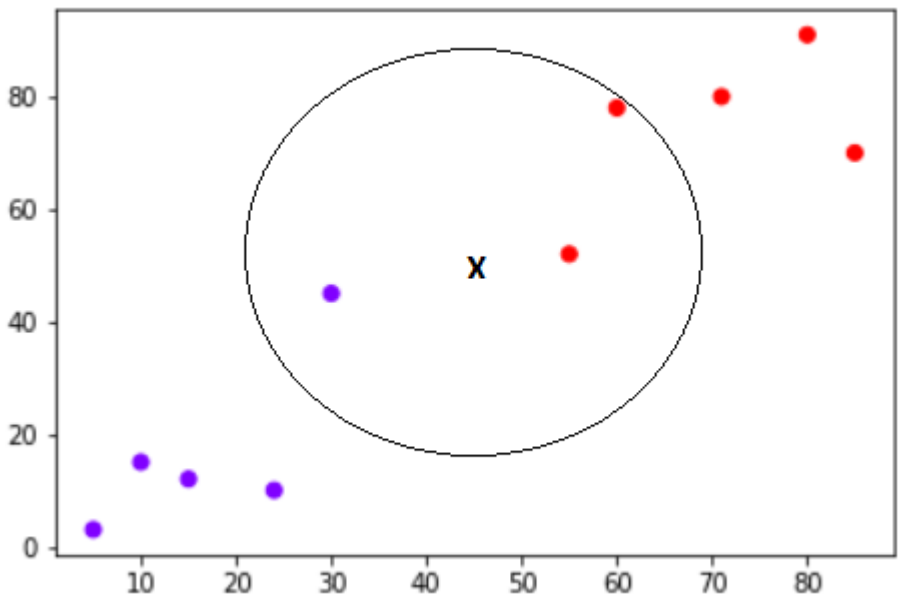

Рисунок 4 – Классификация точки X в «Красный » или в «Синий» класс

2.2.3 Случайный лес

Случайный лес — один из самых универсальных алгоритмов машинного обучения, придуманные ещё в прошлом веке. Универсальность заключается, во-первых, в том, что есть случайные леса для решения задач классификации, регрессии, кластеризации, поиска аномалий, селекции признаков и т.д.

RF (random forest) — это множество решающих деревьев. В задаче регрессии их ответы усредняются, в задаче классификации принимается решение голосованием по большинству. Все деревья строятся независимо по следующей схеме:

- выбирается подвыборка обучающей выборки размера samplesize (м.б. с возвращением) – по ней строится дерево (для каждого дерева — своя подвыборка);

- для построения каждого расщепления в дереве просматриваем max_features случайных признаков (для каждого нового расщепления — свои случайные признаки).

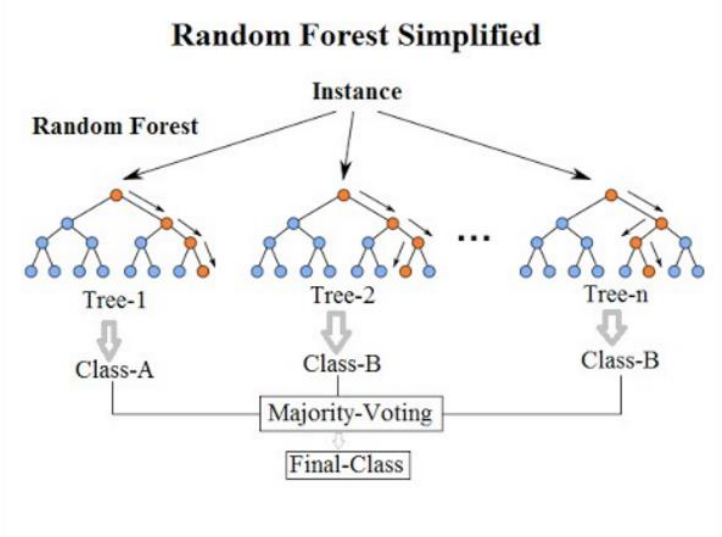

Рисунок 5 – Пример построения случайного леса

Выбираем наилучшие признак и расщепление по нему (по заранее заданному критерию). Дерево строится, как правило, до исчерпания выборки (пока в листьях не останутся представители только одного класса), но в современных реализациях есть параметры, которые ограничивают высоту

дерева, число объектов в листьях и число объектов в подвыборке, при котором проводится расщепление.

Понятно, что такая схема построения соответствует главному принципу ансамблирования (построению алгоритма машинного обучения на базе нескольких, в данном случае решающих деревьев): базовые алгоритмы должны быть хорошими и разнообразными (поэтому каждое дерево строится на своей обучающей выборке и при выборе расщеплений есть элемент случайности).

Достоинства алгоритма

- Высокая скорость обучения

- Неитеративное обучение — алгоритм завершается за фиксированное число операций

- Масштабируемость (способность обрабатывать большие объемы данных)

- Высокое качество получаемых моделей (сравнимое с нейронными сетями и ансамблями нейронных сетей)

- Отсутствие чувствительности к выбросам в данных из-за случайного сэмплирования

- Малое количество настраиваемых параметров

- Отсутствие чувствительности к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков, благодаря выбору случайных подпространств

- Не требует тщательной настройки параметров, хорошо работает «из коробки». С помощью «тюнинга» параметров можно достичь прироста от 0.5 до 3% точности в зависимости от задачи и данных

- Хорошо работает с пропущенными данными — сохраняет хорошую точность, даже если большая часть данных пропущена

- Внутренняя оценка способности модели к обобщению

- Возможность работы с сырыми данными, без препроцессинга

Недостатки алгоритма

- Построенная модель занимает большое количество памяти. Если мы строим комитет из K деревьев на основе обучающего множества размером N, то требования к памяти составят O(K·N). Например, для K=100 и N=1000 модель займет порядка 1 Мб памяти. Но объем оперативной памяти у современных ПК достаточно велик, так что это не самый серьезный недостаток.

- Обученная модель работает несколько медленнее других алгоритмов (если в модель входит 100 деревьев, мы должны пройтись по всем, чтобы получить результат). Но на современных быстрых машинах и это не столь заметно.

- Алгоритм работает хуже многих линейных методов, когда в выборке очень много разреженных признаков (тексты, Bag of words) или когда классифицируемые объекты заведомо могут быть разделены линейно.

- Алгоритм склонен к переобучению, особенно на зашумленных задачах.

- Для данных, включающих категориальные переменные с различным количеством уровней, случайные леса предвзяты в пользу признаков с большим количеством уровней. Дерево будет сильнее подстраиваться именно под такие признаки, поскольку на них можно получить более высокое значение оптимизируемого функционала (типа прироста информации).

- Как и деревья решений, алгоритм абсолютно не способен к экстраполяции (но это можно считать и плюсом, так как не будет экстремальных значений в случае попадания выброса).

2.2.4 Нейронные сети, персептрон.

Искусственная нейронная сеть (ИНС) (англ. Artificial neural network (ANN)) — упрощенная модель биологической нейронной сети, представляющая собой совокупность искусственных нейронов, взаимодействующих между собой.

Основные принципы работы нейронных сетей были описаны еще в 1943 году Уорреном Мак-Каллоком и Уолтером Питтсом. В 1957 году нейрофизиолог Фрэнк Розенблатт разработал первую нейронную сеть, а в 2010 году большие объемы данных для обучения открыли возможность использовать нейронные сети для машинного обучения.

На данный момент нейронные сети используются в многочисленных областях машинного обучения и решают проблемы различной сложности.

Хорошим примером биологической нейронной сети является человеческий мозг. Наш мозг — сложнейшая биологическая нейронная сеть, которая принимает информацию от органов чувств и каким-то образом ее обрабатывает (узнавание лиц, возникновение ощущений и т.д.). Мозг же, в свою очередь, состоит из нейронов, взаимодействующих между собой.

Для построения искусственной нейронной сети будем использовать ту же структуру. Как и биологическая нейронная сеть, искусственная состоит из нейронов, взаимодействующих между собой, однако представляет собой упрощенную модель. Так, например, искусственный нейрон, из которых состоит ИНС, имеет намного более простую структуру: у него есть несколько входов, на которых он принимает различные сигналы, преобразует их и передает другим нейронам. Другими словами, искусственный нейрон — это такая функция Rn→R, которая преобразует несколько входных параметров в один выходной.

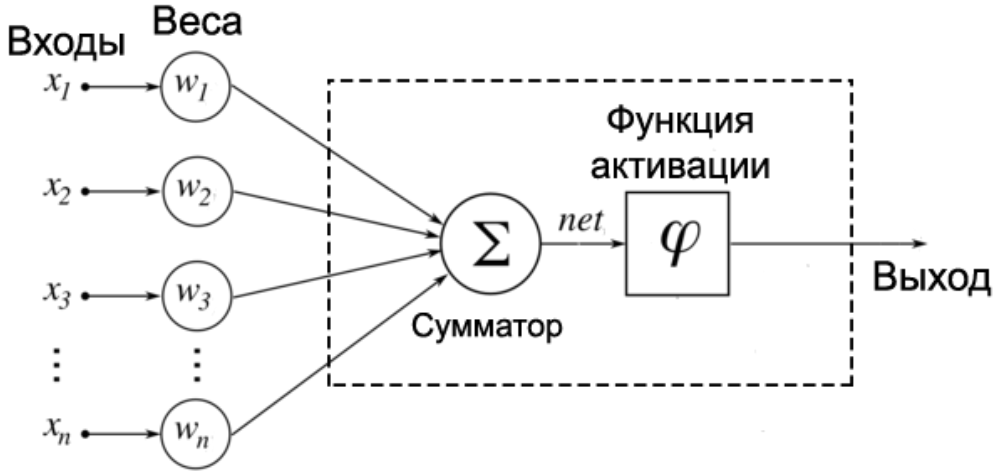

Рисунок 6 – Схема искусственного нейрона.

Просто так передавать взвешенную сумму net на выход достаточно бессмысленно — нейрон должен ее как-то обработать и сформировать адекватный выходной сигнал. Для этих целей используют функцию активации, которая преобразует взвешенную сумму в какое-то число, которое и будет являться выходом нейрона. Функция активации обозначается ϕ(net). Таким образом, выходов искусственного нейрона является ϕ(net).

Для разных типов нейронов используют самые разные функции активации, но одними из самых популярных являются:

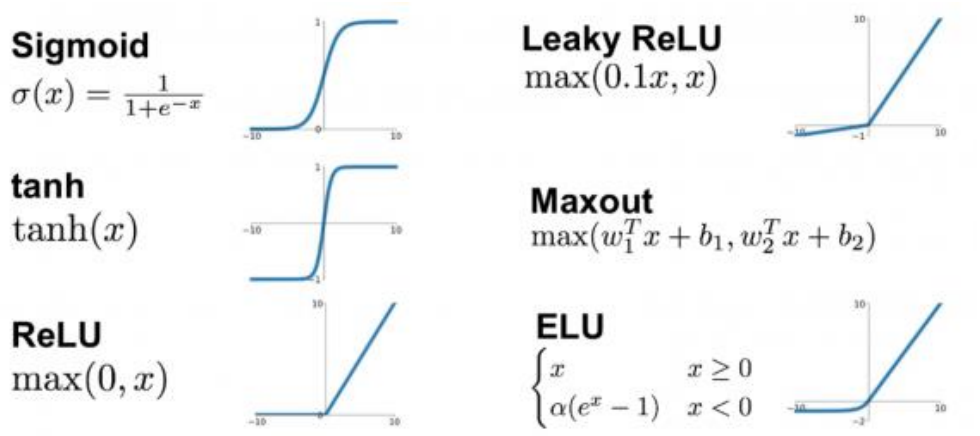

Рисунок 7 Виды функций активации

Различают следующие виды нейронных сетей

Однослойная нейронная сеть (англ. Single-layer neural network) — сеть, в которой сигналы от входного слоя сразу подаются на выходной слой, который и преобразует сигнал и сразу же выдает ответ.

Многослойная нейронная сеть (англ. Multilayer neural network) — нейронная сеть, состоящая из входного, выходного и расположенного(ых) между ними одного (нескольких) скрытых слоев нейронов.

Помимо входного и выходного слоев эти нейронные сети содержат промежуточные, скрытые слои. Такие сети обладают гораздо большими возможностями, чем однослойные нейронные сети, однако методы обучения нейронов скрытого слоя были разработаны относительно недавно.

Сети прямого распространения (англ. Feedforward neural network) (feedforward сети) — искусственные нейронные сети, в которых сигнал

распространяется строго от входного слоя к выходному. В обратном направлении сигнал не распространяется.

Сети с обратными связями (англ. Recurrent neural network) — искусственные нейронные сети, в которых выход нейрона может вновь подаваться на его вход. В более общем случае это означает возможность распространения сигнала от выходов к входам.

В сетях прямого распространения выход сети определяется входным сигналом и весовыми коэффициентами при искусственных нейронах. В сетях с обратными связями выходы нейронов могут возвращаться на входы. Это означает, что выход какого-нибудь нейрона определяется не только его весами и входным сигналом, но еще и предыдущими выходами (так как они снова вернулись на входы).

Обучение нейронной сети — поиск такого набора весовых коэффициентов, при котором входной сигнал после прохода по сети преобразуется в нужный нам выходной.

Это определение «обучения нейронной сети» соответствует и биологическим нейросетям. Наш мозг состоит из огромного количества связанных друг с другом нейросетей, каждая из которых в отдельности состоит из нейронов одного типа (с одинаковой функцией активации). Наш мозг обучается благодаря изменению синапсов — элементов, которые усиливают или ослабляют входной сигнал.

Если обучать сеть, используя только один входной сигнал, то сеть просто «запомнит правильный ответ», а как только мы подадим немного измененный сигнал, вместо правильного ответа получим бессмыслицу. Мы ждем от сети способности обобщать какие-то признаки и решать задачу на различных входных данных. Именно с этой целью и создаются обучающие выборки.

2.3 Разведочный анализ данных

Разведочный анализ данных – предварительное исследование Датасета с целью определения его основных характеристик, взаимосвязей между признаками, а также сужения набора методов, используемых для создания Модели Машинного обучения.

Описание признаков объединенного датасета приведено в таблице 1. Все признаки имеют тип float64, то есть вещественный. Пропусков в данных нет. Все признаки, кроме «Угол нашивки», являются непрерывными, количественными. «Угол нашивки» принимает только два значения и будет рассматриваться как категориальный признак.

Таблица 1 — Описание признаков датасета

Название	Файл	Тип данных	Непустых значений	Уникальных значений
Соотношение матрица-наполнител	X_bp	float64	1023	1014
Плотность, кг/м3	X_bp	float64	1023	1013
модуль упругости, ГПа	X_bp	float64	1023	1020
Количество отвердителя, м.%	X_bp	float64	1023	1005
Содержание эпоксидных групп,%_2	X_bp	float64	1023	1004
Температура вспышки, С_2	X_bp	float64	1023	1003
Поверхностная плотность, г/м2	X_bp	float64	1023	1004
Модуль упругости при растяжении, ГПа	X_bp	float64	1023	1004
Прочность при растяжении, МПа	X_bp	float64	1023	1004
Потребление смолы, г/м2	X_bp	float64	1023	1003
Угол нашивки, град	X_nup	float64	1023	2
Шаг нашивки	X_nup	float64	1023	989
Плотность нашивки	X_nup	float64	1023	988

Для первоначального анализа данных в настоящей работе использованы описательная статистика, а также методы графической

визуализации данных. Раздел описательной статистики включает в себя проверку на нормальность распределения и определение прочих статистических метрик, необходимых для того, чтобы лучше «понять» конкретный датасет и удостовериться в применимости к нему тех или иных моделей и методов обучения.

В рассматриваемом случае получены общие статистические сведения о датафрейме, включая среднее значение, медианное значение, стандартное отклонение, минимальный элемент, максимальный элемент и др.

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Рисунок 8 – Описательная статистика датафрейма

При разведочном анализе данных установлено, что распределение значений переменных близко к нормальному.

Гистограмма представляет собой диаграмму, которая используется в статистике для графического представления распределения вероятностей значений случайной величины. По горизонтальной оси гистограммы откладывается диапазон наблюдаемых значений величины, разбитый на определенное число (обычно 10 - 15) интервалов, а по вертикальной — вероятность или частота ее попадания в каждый интервал.

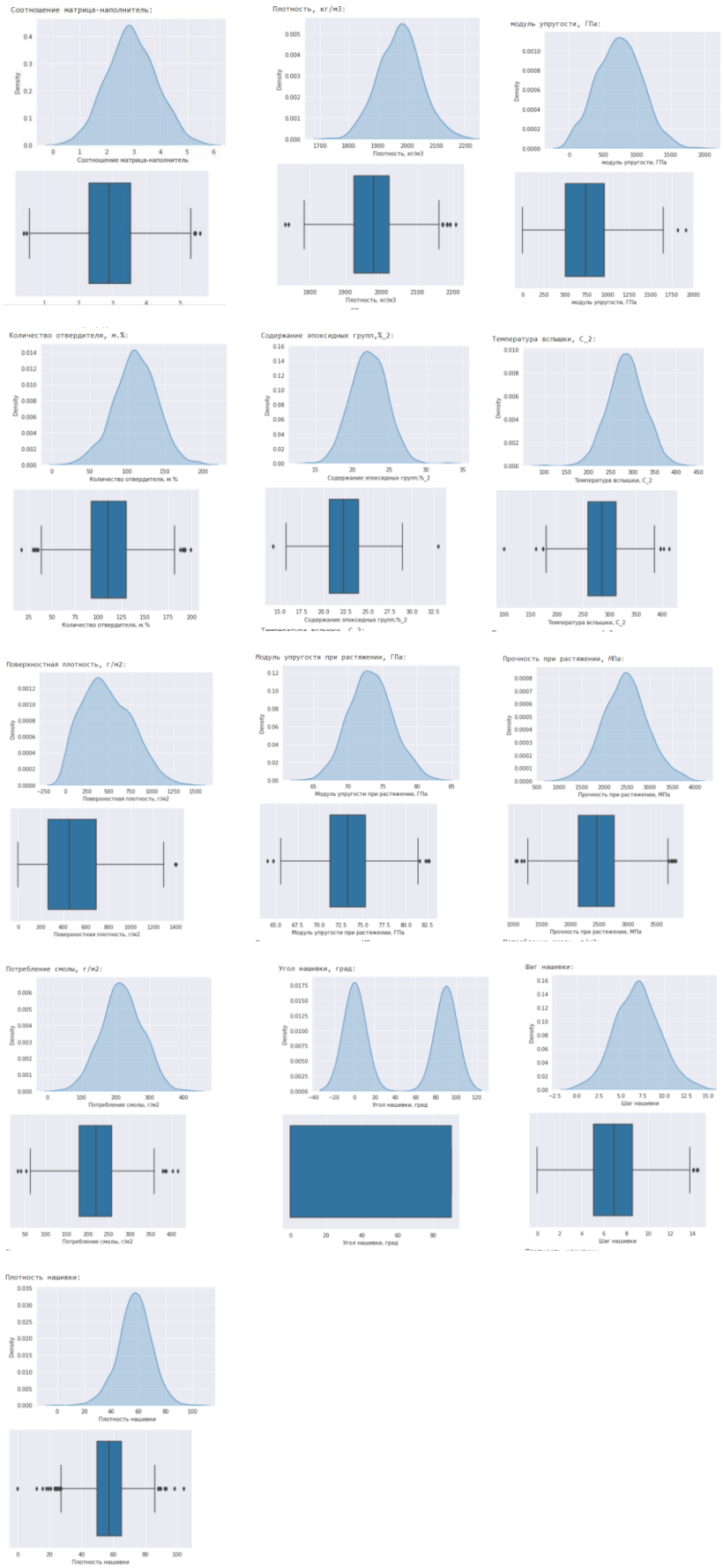

Рисунок 9 - Гистограммы распределения переменных и диаграммы «ящик с усами»

Гистограммы распределения переменных и диаграммы «ящик с усами» приведены на рисунке 9. По ниму видно, что все признаки, кроме «Угол нашивки», имеют нормальное распределение и принимают неотрицательные значения. «Угол нашивки» принимает значения: 0, 90.

Из рисунков выше усматривается, что отдельные выбросы присутствуют, но не выглядят значительными. Для обнаружения корреляции между данными построены попарные графики рассеяния точек, представляющие наблюдаемые признаки в пространстве двух измерений.

Построенные графики рассеяния представлены на рисунке 10.

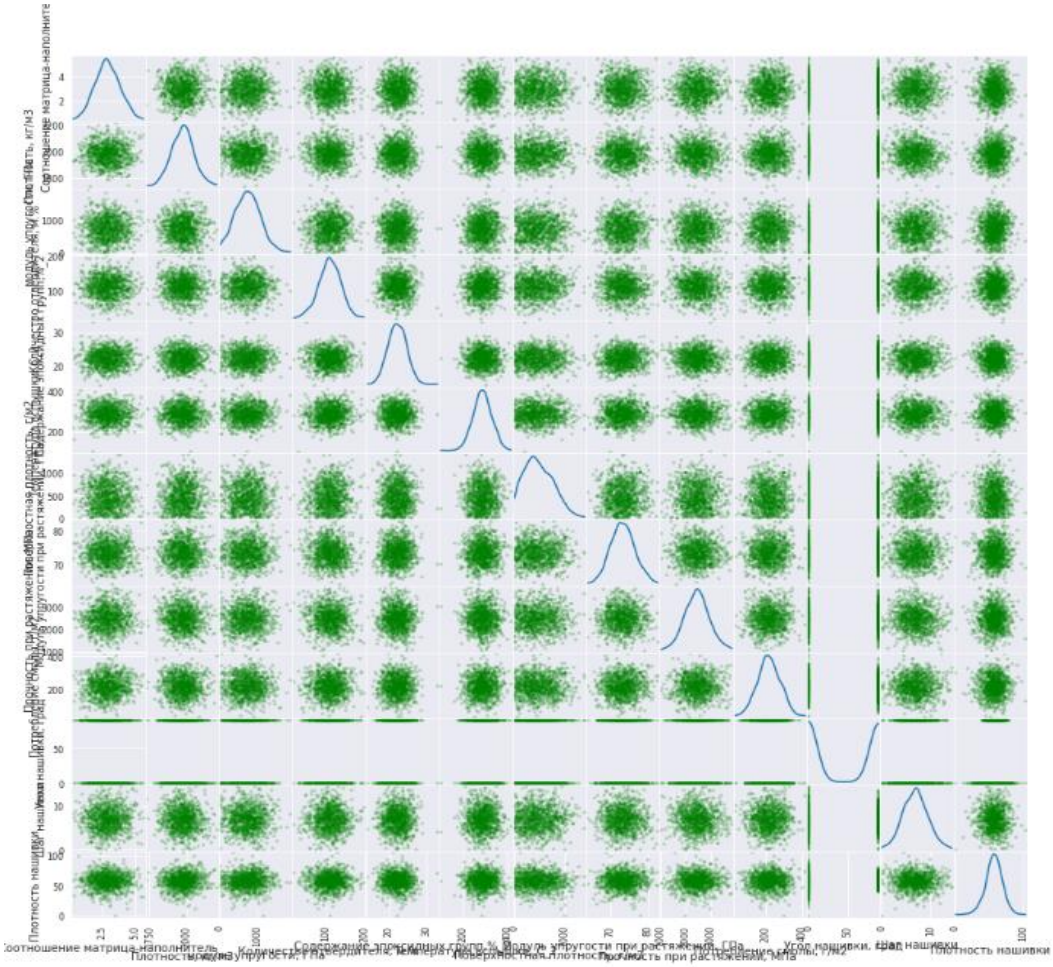

Рисунок 10 - Попарные графики рассеяния точек.

Как усматривается из графиков на рисунке 10, влияние одних переменных на другие не проявлено, подтверждается вывод о наличии некоторых выбросов в данных (точек, удаленных от общего кластера точек).

Для дополнительной проверки предположения об отсутствии корреляции построена тепловая карта корреляции, представленная на рисунке 11, содержащая значения от -1 до +1.

Рисунок 11 – Тепловая карта корреляции.

Все значения корреляции близки к 0, что свидетельствует о том, что наблюдаемые признаки, в целом, не зависят друг от друга.

Как следует из проведенного разведочного анализа данных, явные зависимости между показателями не определены, что может свидетельствовать как об ошибках в данных, так и о необходимости серьёзного изучения предметной области исследования, понимания самих данных.

3 Практическая часть

3.1 Предобработка данных

Выполним проверку полученного датасета на пропуски

```
[19] df.isnull().sum()
```

```
Соотношение матрица-наполнитель         0
Плотность, кг/м3                        0
модуль упругости, ГПа                   0
Количество отвердителя, м.%             0
Содержание эпоксидных групп,%_2         0
Температура вспышки, С_2                0
Поверхностная плотность, г/м2           0
Модуль упругости при растяжении, ГПа    0
Прочность при растяжении, МПа           0
Потребление смолы, г/м2                 0
Угол нашивки, град                      0
Шаг нашивки                             0
Плотность нашивки                       0
dtype: int64
```

Пропуски отсутствуют

Рисунок 12 - Пропуски отсутствуют

После проведения разведочного анализа предоставленных данных было установлено наличие в них некоторых выбросов.

Проверку датасета на наличие выбросов проведем 2-мя методами: методом 3 σ и методом межкартильных расстояний.

```
count_3s = 0
count_iq = 0
for column in df:
    d = df.loc[:, [column]]
    # методом 3-х сигм
    zscore = (df[column] - df[column].mean()) / df[column].std()
    d['3s'] = zscore.abs() > 3
    count_3s += d['3s'].sum()
    # методом межквартильных расстояний
    q1 = np.quantile(df[column], 0.25)
    q3 = np.quantile(df[column], 0.75)
    iqr = q3 - q1
    lower = q1 - 1.5 * iqr
    upper = q3 + 1.5 * iqr
    d['iq'] = (df[column] <= lower) | (df[column] >= upper)
    count_iq += d['iq'].sum()
    # визуализация выбросов
    print('{}: 3s={} iq={}'.format(column, d['3s'].sum(), d['iq'].sum()))
    fig, axes = plt.subplots(1, 2, figsize=(12, 2.5))
    sns.histplot(data=d, x=column, hue='3s', multiple='stack', legend=False, ax=axes[0])
    sns.boxplot(data=d, x=column, color='tab:gray', ax=axes[1])
    sns.stripplot(data=d[d['iq']==False], x=column, ax=axes[1])
    sns.stripplot(data=d[d['iq']==True], x=column, color='tab:orange', ax=axes[1])
    plt.show()
```

Рисунок 13 – Определение выбросов

После определения выбросов разными методами получим следующие диаграммы на рисунке 14.

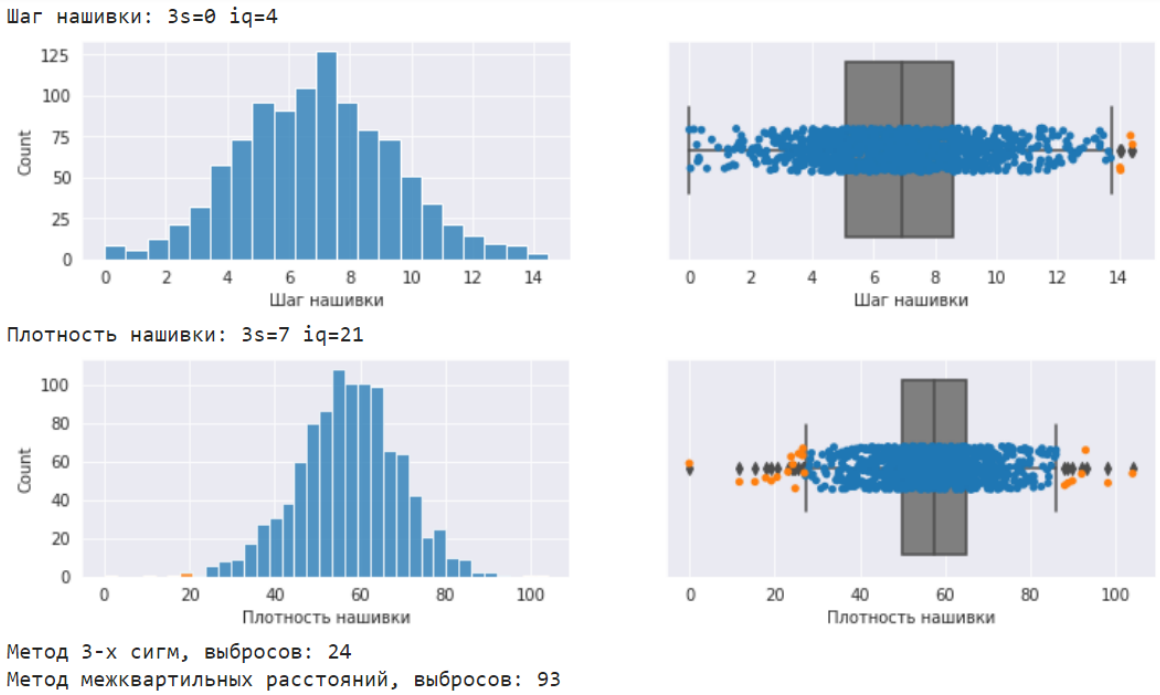

Рисунок 14 – Определение выбросов на примере переменных шаг нашивки и плотность нашивки..

Применив эти методы на нашем датасете было найдено:

- методом 3-х сигм — 24 выброса;

- методом межквартильных расстояний — 93 выброса.

Поскольку известно, что датасет очищен от явного шума, следует применить метод 3-х сигм как более деликатный, чтобы не потерять значимые данные. Значения, определенные как выбросы, удаляем. После этого осталось в датасете осталось 1000 строк и 13 признаков-переменных.

В задании целевыми переменными указаны:

- модуль упругости при растяжении, Гпа;

- прочность при растяжении, МПа;

- соотношение матрица-наполнитель.

Выполним нормализацию данных

```
norm_df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1000.0	0.489568	0.174687	0.0	0.370964	0.484284	0.608289	1.0
Плотность, кг/м3	1000.0	0.467648	0.178696	0.0	0.340831	0.472347	0.579727	1.0
модуль упругости, ГПа	1000.0	0.447024	0.198876	0.0	0.302576	0.448525	0.582408	1.0
Количество отвердителя, м.%	1000.0	0.496427	0.171089	0.0	0.384097	0.495388	0.613258	1.0
Содержание эпоксидных групп,%_2	1000.0	0.493216	0.179818	0.0	0.368597	0.492154	0.624396	1.0
Температура вспышки, С_2	1000.0	0.488654	0.174792	0.0	0.371985	0.488205	0.606271	1.0
Поверхностная плотность, г/м2	1000.0	0.371301	0.215155	0.0	0.206374	0.348844	0.535295	1.0
Модуль упругости при растяжении, ГПа	1000.0	0.497322	0.167158	0.0	0.386234	0.492609	0.605138	1.0
Прочность при растяжении, МПа	1000.0	0.507902	0.172506	0.0	0.390414	0.504890	0.612932	1.0
Потребление смолы, г/м2	1000.0	0.512370	0.170432	0.0	0.401220	0.513653	0.625772	1.0
Угол нашивки, град	1000.0	0.496000	0.500234	0.0	0.000000	0.000000	1.000000	1.0
Шаг нашивки	1000.0	0.477193	0.177586	0.0	0.351886	0.477999	0.593714	1.0
Плотность нашивки	1000.0	0.507027	0.163634	0.0	0.405037	0.509730	0.612766	1.0

Рисунок 15 – Нормализация данных.

3.2 Разработка и обучение моделей

В соответствии с постановкой задачи, при построении моделей 30 процентов данных оставлено на тестирование модели, на остальных происходит обучение.

Для прогноза модуля упругости при растяжении и прочности при растяжении использованы модели:

- линейной регрессии;
- k-ближайших соседей;
- случайного леса.

Результаты применения модели линейной регрессии для прогноза модуля упругости при растяжении и для прогноза прочности при растяжении представлены на рисунках 16 и 17 соответственно.

Заметно, что при прогнозе обоих показателей качество модели низкое.

```
[ ]  print('Среднеквадратичная ошибка - ', test_set_rmse)
     print('Коэфициент детерминации - ', test_set_r2)
     print('Точность - ', accur, '%')
```

```
Среднеквадратичная ошибка -  0.1566792084714014
Коэфициент детерминации -  -0.01619201139595594
Точность -  67.46961522978057 %
```

```
[ ]  test_predictions = lin_reg_mod.predict(Xtest).flatten()
     chart_predict_true(Xtest, Ytest, test_predictions)
```

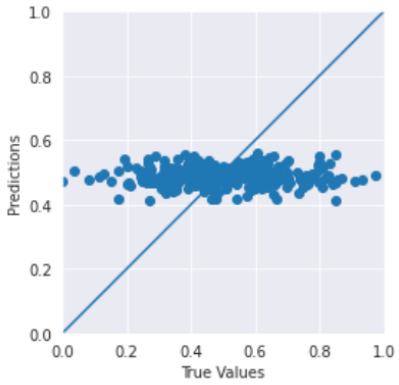

Рисунок 16 - Модель линейной регрессии для прогноза модуля упругости.

```
[ ]  print('Среднеквадратичная ошибка - ', test_set_rmse)
     print('Коэфициент детерминации - ', test_set_r2)
     print('Точность - ', accur, '%')
```

```
Среднеквадратичная ошибка -  0.16763751800446028
Коэфициент детерминации -  0.014738274682774333
Точность -  64.76910365700874 %
```

```
[ ]  test_predictions = lin_reg_mod.predict(Xtest).flatten()
     chart_predict_true(Xtest, Ytest, test_predictions)
```

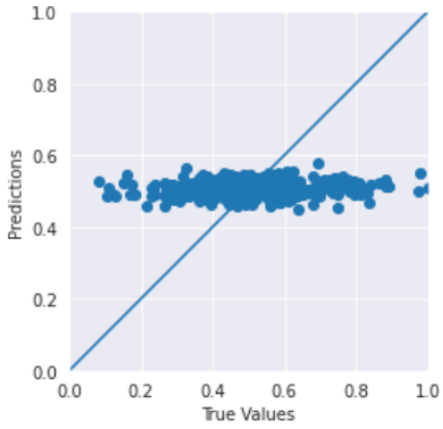

Рисунок 17 Модель линейной регрессии для прогноза модуля прочности.

Результаты применения модели k-ближайших соседей для прогноза модуля упругости при растяжении и для прогноза прочности при растяжении представлены на рисунках 18 и 19 соответственно

```
Среднеквадратичная ошибка -  0.16904296799993562
Коэфициент детерминации -  -0.18289789062088624
Точность -  65.48569179785088 %
```

```
[ ]  test_predictions = knn.predict(Xtest).flatten()
     chart_predict_true(Xtest, Ytest, test_predictions)
```

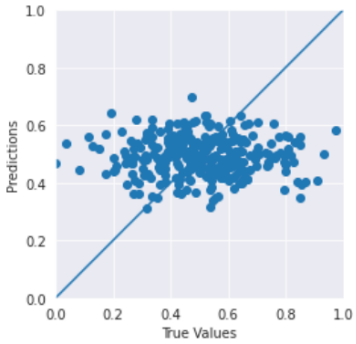

Рисунок 18 - Модель k-ближайших соседей для прогноза модуля упругости.

```
Среднеквадратичная ошибка -  0.17740829736278182
Коэфициент детерминации -  -0.10346108902697027
Точность -  63.33463979327355 %
```

```
[ ]  test_predictions = knn.predict(Xtest).flatten()
     chart_predict_true(Xtest, Ytest, test_predictions)
```

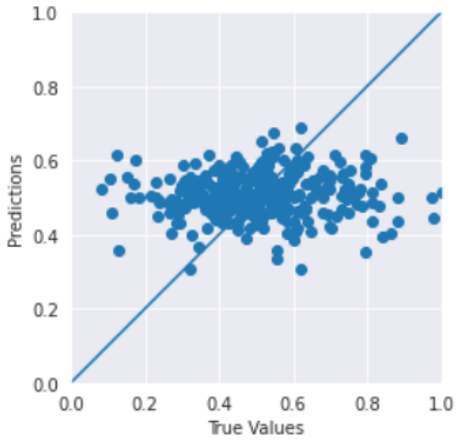

Рисунок 19 - Модель k-ближайших соседей для прогноза модуля прочности.

Результаты применения модели случайного леса для прогноза модуля упругости при растяжении и для прогноза прочности при растяжении представлены на рисунках 20 и 21 соответственно.

```
Среднеквадратичная ошибка -  0.15893730809213732
Коэфициент детерминации -  -0.04569431030090776
Точность -  67.57028239115407 %
```

```
[ ]  test_predictions = model.predict(Xtest).flatten()
     chart_predict_true(Xtest, Ytest, test_predictions)
```

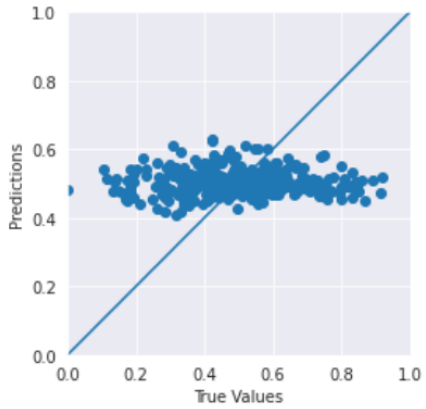

Рисунок 20 - Модель случайного леса для прогноза модуля упругости.

```
Среднеквадратичная ошибка -  0.17262402116215833
Коэфициент детерминации -  -0.04474818574594308
Точность -  63.38700908002451 %
```

```
test_predictions = model.predict(Xtest).flatten()
chart_predict_true(Xtest, Ytest, test_predictions)
```

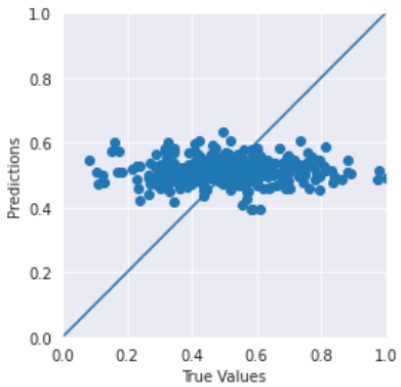

Рисунок 21 - Модель случайного леса для прогноза модуля прочности.

Поиск лучших гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10 для моделей случайного леса приведён на рисунках 22 и 23.

```
n_estimators = [3,5,9,11]
max_depth = [1,2,3,4,5]
min_samples_split = [10,20,30]
min_samples_leaf = [2,4,6]
bootstrap = [True]
param_grid = {'n_estimators': n_estimators,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}
gs = GridSearchCV(model, param_grid, cv = 10, verbose = 1, n_jobs=1)
gs.fit(Xtrn, Ytrn)
gs.best_params_
```

```
Fitting 10 folds for each of 180 candidates, totalling 1800 fits
{'bootstrap': True,
 'max_depth': 1,
 'min_samples_leaf': 2,
 'min_samples_split': 10,
 'n_estimators': 5}
```

Рисунок 22 – Гиперпараметры модели случайного леса для прогноза модуля упругости.

```
n_estimators = [3,5,9,11]
max_depth = [1,2,3,4,5]
min_samples_split = [10,20,30]
min_samples_leaf = [2,4,6]
bootstrap = [True]
param_grid = {'n_estimators': n_estimators,
              'max_depth': max_depth,
              'min_samples_split': min_samples_split,
              'min_samples_leaf': min_samples_leaf,
              'bootstrap': bootstrap}
gs = GridSearchCV(model, param_grid, cv = 10, verbose = 1, n_jobs=1)
gs.fit(Xtrn, Ytrn)
gs.best_params_
```

```
Fitting 10 folds for each of 180 candidates, totalling 1800 fits
{'bootstrap': True,
 'max_depth': 2,
 'min_samples_leaf': 2,
 'min_samples_split': 10,
 'n_estimators': 5}
```

Рисунок 23 – Гиперпараметры модели случайного леса для прогноза модуля прочности.

Результаты применения модели линейной регрессии, модели k-ближайших соседей и модели случайного леса для прогноза модуля

упругости при растяжении и для прогноза прочности при растяжении не достаточны и не решают поставленных задач.

3.3 Модель нейронной сети для параметра матрица-наполнитель.

Построение модели нейронной сети выполним в библиотеке keras. Программный код, параметры и архитектура нейронной сети предоставлены на рисунках 24. 25 и 26.

```
[ ]  print(tf.__version__)

     2.8.2
```

```
[ ]  model = tf.keras.models.Sequential([
       tf.keras.layers.Dense(24, activation='relu',
                             input_shape=(12,)),
       tf.keras.layers.Dense(12, activation='sigmoid'),
       tf.keras.layers.Dense(6, activation='sigmoid'),
       tf.keras.layers.Dense(1)
     ])
```

```
[ ]  model.compile(optimizer=tf.keras.optimizers.Adam(0.01),
                   loss='mse',
                   metrics=['mae'])
```

Рисунок 24 – Конструктор нейронной сети.

```
[ ]  model.summary()

     Model: "sequential_4"
     _________________________________________________________________
      Layer (type)                Output Shape              Param #
     =================================================================
      dense_16 (Dense)            (None, 24)                312

      dense_17 (Dense)            (None, 12)                300

      dense_18 (Dense)            (None, 6)                 78

      dense_19 (Dense)            (None, 1)                 7

     =================================================================
     Total params: 697
     Trainable params: 697
     Non-trainable params: 0
     _________________________________________________________________
```

Рисунок 25 – Параметры нейронной сети.

```
[ ]  # Архитектура нейросети визуально
     keras.utils.plot_model(model, show_shapes=True, show_layer_names=True, show_layer_activations=True)
```

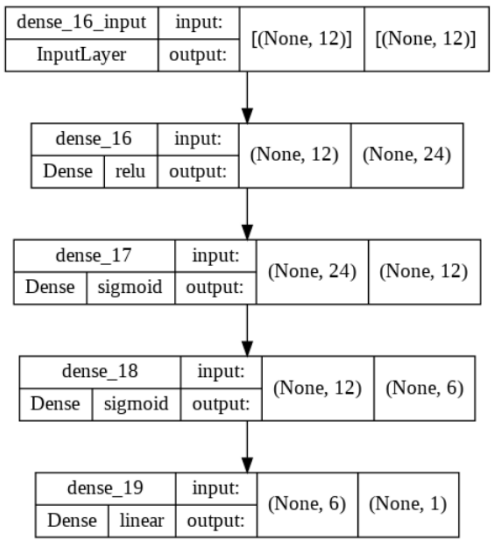

Рисунок 26 – Архитектура нейронной сети.

Проведем обучение нейронной сети на тестовой выборке.

```
[ ]  # Обучение нейросети
     history = model.fit(
         Xn_train.values,
         Yn_train.values,
         epochs=100,
         verbose=1,
         validation_split = 0.4)
```

Рисунок 27 – Обучение нейронной сети

Проведем проверку работы нейронной сети на тестовой выборке

```
[ ]  Yn_pred = model.predict(Xn_test)
```

```
[ ]  plt.figure(figsize=(15,5))
     plt.title(str('Соотношение матрица-наполнитель'))
     plt.plot(history.history['loss'], label='loss')
     plt.plot(history.history['val_loss'], label='val_loss')
     #plt.ylim([0,0.8])
     plt.xlabel('Epoch')
     plt.ylabel('Error')
     plt.legend()
     plt.grid(True)
```

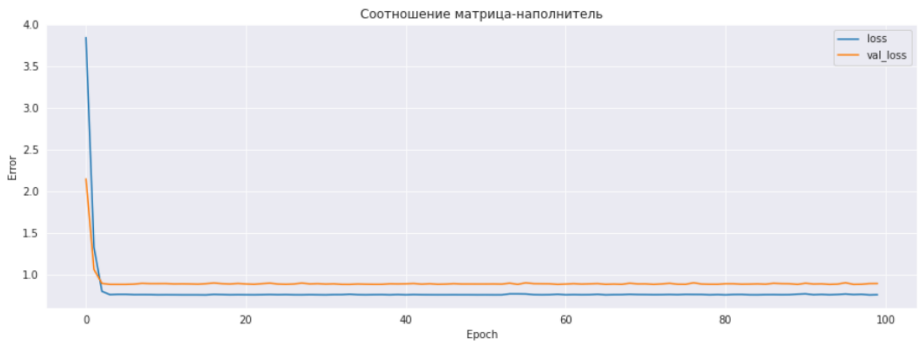

Рисунок 28 – Характеристика изменения среднеквадратической ошибки при обучении модели.

Проведем построение модели предсказания параметра Соотношения матрица-наполнитель по данным тестовой выборки.

```
[ ] print('Проверка построенной модели')
    plt.figure(figsize=(15,5))
    plt.scatter(range(0, len(Yn_test)), Yn_test, label = 'data', s =20)
    plt.plot(range(0, len(Yn_test)), Yn_pred, label = 'predict', color = 'green')
    plt.title(str('Соотношение матрица-наполнитель'))
    plt.legend()
    plt.show()
```

```
Проверка построенной модели
```

Рисунок 29 – Проверка нейронной сети

```
plt.figure(figsize=(15,5))
sns.kdeplot(data=Yn_pred, label = 'Yn_pred', color='b')
sns.kdeplot(data=Yn_test, label = 'Yn_test', color='green')
plt.xlabel("Значение")
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:316: UserWarning: Dataset has 0 variance; skipping densi
  warnings.warn(msg, UserWarning)
```

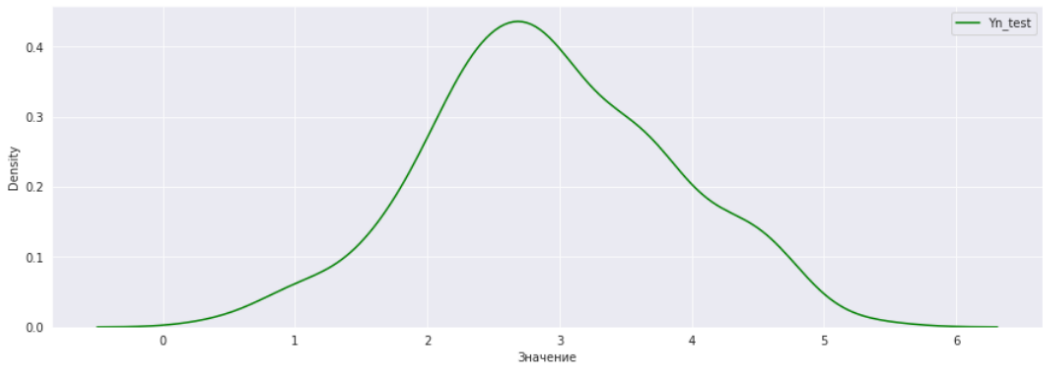

Рисунок 30 – Значение параметра Соотношения матрица-наполнитель от частоты появления параметра.

Суть прогнозирования заключается в симуляции представительного элемента объема композита, на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

Оценим эффективность построенной модели

```
[ ]  # Оценим эффективность модели
     test_set_rmse = (np.sqrt(mean_squared_error(Yn_test, Yn_pred)))
     test_set_r2 = r2_score(Yn_test, Yn_pred)
```

```
[ ]  print('Среднеквадратичная ошибка - ', test_set_rmse)
     print('Коэфициент детерминации - ', test_set_r2)
     print('Точность - ', accur, '%')
```

```
Среднеквадратичная ошибка -  0.9243354275191141
Коэфициент детерминации -  -0.0013813076090694043
Точность -  63.38700908002451 %
```

Рисунок 31 – эффективность модели.

Таким образом, построенная модель неэффективна, и не решает поставленную задачу. Для создания эффективной модели нейронной сети необходимо вернуться к этапу подготовки данных и разделить датасет на два кластера провести обучение нейронной сети по каждому кластеру.

3.4 Проверка работоспособности модели

ПРОВЕРКА РАБОТОСПОСОБНОСТИ МОДЕЛИ

```
[ ]  #Ввод параметров и получение рекомендуемого значения

     print("Введите параметры для прогноза соотношения 'Матрица-наполнител
     print("Плотность, кг/м3")
     f1 = np.float64(input())
     print("Модуль упругости, ГПа")
     f2 = np.float64(input())
     print("Количество отвердителя, м.%")
     f3 = np.float64(input())
     print("Содержание эпоксидных групп,%_2")
     f4 = np.float64(input())
     print("Температура вспышки, С_2")
     f5 = np.float64(input())
     print("Поверхностная плотность, г/м2")
     f6 = np.float64(input())
     print("Модуль упругости при растяжении, ГПа")
     f7 = np.float64(input())
     print("Прочность при растяжении, МПа")
     f8 = np.float64(input())
     print("Потребление смолы, г/м2")
     f9 = np.float64(input())
     print("Угол нашивки, град")
     f10 = np.float64(input())
     print("Шаг нашивки")
     f11 = np.float64(input())
     print("Плотность нашивки")
     f12 = np.float64(input())

     args = np.array([[f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12]])
```

Рисунок 32 – Программа для ввода параметров конечным пользователем.

Результат работы программы цифровой симуляции параметра матрицы-наполнитель предоставлен на рисунке 33.

```
Введите параметры для прогноза соотношения 'Матрица-наполнитель'
Плотность, кг/м3
2000
Модуль упругости, ГПа
800
Количество отвердителя, м.%
50
Содержание эпоксидных групп,%_2
30
Температура вспышки, С_2
280
Поверхностная плотность, г/м2
210
Модуль упругости при растяжении, ГПа
73
Прочность при растяжении, МПа
2000
Потребление смолы, г/м2
200
Угол нашивки, град
90
Шаг нашивки
10
Плотность нашивки
65
```

```
rekom = model.predict(args)
print("Рекомендуемое соотношение 'Матрица-наполнитель' =", rekom)
```

```
Рекомендуемое соотношение 'Матрица-наполнитель' = [[2.8843265]]
```

Рисунок 33 – Пример ввода параметров композита конечным пользователем и получение результата.

В запросе уже стоят данные, на которых можно проверить модель, либо данные можно заменить собственными с соблюдением требований диапазонов. Если какие-то значения диапазонов выходят за рамки допустимых пользователь получает соответствующую ошибку. Если все в норме – программа распаковывает необходимые предобработчики и модели и запускает прогноз.

На выходе пользователь видит значения соотношения матрица/наполнитель в зависимости от его запроса.

3.5 Создание удаленного репозитория и загрузка результатов работы на него.

Создан репозиторий в GitHub, где размещен код исследования, оформлен файл README.

Созданный репозиторий: https://github.com/Evgeniy1V/Frankenshtein

Заключение

В выпускной квалификационной работе была осуществлена постановка задачи прогнозирования ряда конечных свойств получаемых композиционных материалов, было проведено теоретико-методологическое обоснование используемых регрессионных методов (случайного леса, линейной регрессии, k-ближайших соседей), был подготовлен набор данных для анализа (датасет).

При проведении анализа установлено, что распределение значений переменных близко к нормальному, очевидные зависимости между переменными отсутствуют, корреляционная зависимость между переменными также отсутствует.

Прогноз прочности при растяжении и прогноз модуля упругости при растяжении были разработаны по трем моделям, давшим близкие результаты. Подбором гиперпараметров определены наилучшие модели. Результаты построения и обучения моделей не дали положительного результата, но позволили приобрести опыт по выбору модели для решения задач регрессии, опыт по настройке таких моделей, опыт по оценке качества моделей и расчета различных метрик, характеризующих качество построенной модели,

Для рекомендации соотношения матрица-наполнитель разработана модель на основе искусственной нейронной трехслойной сети.

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

Результаты работы размещены в удаленном репозитории GitHub.

Библиографический список

1. Д.А. Иванов А.И., Ситников, С.Д Шляпин – Композиционные материалы: учебное пособие для вузов, 2019. 13 с.

2. А.А. Миронов. Машинное обучение часть I ст.9 – Режим доступа: http://is.ifmo.ru/verification/machine-learning-mironov.pdf.

3 Язык программирования Python- Режим доступа: https://www.python.org/.

4. Библиотека Pandas- Режим доступа: https://pandas.pydata.org/.

5. Библиотека Matplotlib- Режим доступа: https://matplotlib.org/.

6. Библиотека Seaborn- Режим доступа: https://seaborn.pydata.org/.

7. Библиотека Sklearn- Режим доступа: https://scikit-learn.org/stable/.

8. Среда разработки Jupyter Notebook- Режим доступа: https://jupyter.org/.

9. Библиотека Tensorflow: Режим доступа: https://www.tensorflow.org/.

10. Грас, Джоэл. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил

11. Бизли Д. Python. Подробный справочник: учебное пособие. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с., ил.

12. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.

13. Траск Эндрю. Грокаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.: ил.

14. Документация по библиотеке keras: – Режим доступа: https://keras.io/api/.