МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский государственный технический университет имени Н.Э.

Баумана

(национальный исследовательский университет)»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

по курсу

«Data Science»

Слушатель Кочетова Гульсина Рафиковна

Москва, 2022

**Введение**

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

У такого подхода есть и недостаток: даже если мы знаем характеристики исходных компонентов, определить характеристики композита, состоящего из этих компонентов, достаточно проблематично. Для решения этой проблемы есть два пути: физические испытания образцов материалов, или прогнозирование характеристик. Суть прогнозирования заключается в симуляции представительного элемента объема композита, на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

На входе имеются данные о начальных свойствах компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.). На выходе необходимо спрогнозировать ряд конечных свойств получаемых композиционных материалов. Кейс основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

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

**1. Аналитическая часть**

**1.1. Постановка задачи**

Требования к пластичности материала матрицы, его совместимо­сти с армирующим волоконном в какой-то степени обусловливают его выбор. Однако обычно основным определяющим фактором служит об­ласть температур, в которой предполагается эксплуатировать изделие из КМ. Если температура эксплуатации 100-200°С, то обычно для КМ выбирают полимерную матрицу. Из них изготавливают лодки, корпус­ные изделия автомобилей, самолетов, товары широкого потребления и др. Применение ПКМ часто не только снижает вес изделия, но и уде­шевляет его производство, например, позволяет изготавливать детали сложной формы за одну операцию.

Для производства изделий из ПКМ используют термореактивные и термопластичные пластики. Термореактивные пластики более тепло­стойкие, чем большинство термопластов. Обычно в качестве терморе­активных матриц для КМ используют полиэфирные и эпоксидные смолы и их разновидности, однако в настоящее время все больший интерес вызывает класс смол, называемых полиимидными, которые могут вы­держивать длительный нагрев до температур, выше 300°С.

Выбор матричного материала определяет также и способ изготов­ления изделия из КМ. При использовании термореактивных связующих после изготовления изделия из КМ необходимо провести процесс отверждения, т.е. обеспечить условия для пространственной сшивки мо­лекул полимера - выдержать изделие из КМ при высокой температуре и давлении в течение нескольких часов.

Использование термопластичных материалов ускоряет процесс из­готовления, т.к. получение КМ в этом случае требует лишь сравнитель­но короткого нагрева, достаточного для размягчения пластика. Темпе­ратура плавления наиболее перспективных термопластов настолько велика, что по теплостойкости они превосходят термореактивы, напри­мер, полиэфиркетон плавится при 334°С.

Датасет представляет собой два файла excel: X\_bp.xlsx и X\_nup.xlsx. После объединения этих двух таблиц получили новый датасет, содержащий 13 признаков и 1023 строки.

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1023 entries, 0 to 1022

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Соотношение матрица-наполнитель 1023 non-null float64

1 Плотность, кг/м3 1023 non-null float64

2 модуль упругости, ГПа 1023 non-null float64

3 Количество отвердителя, м.% 1023 non-null float64

4 Содержание эпоксидных групп,%\_2 1023 non-null float64

5 Температура вспышки, С\_2 1023 non-null float64

6 Поверхностная плотность, г/м2 1023 non-null float64

7 Модуль упругости при растяжении, ГПа 1023 non-null float64

8 Прочность при растяжении, МПа 1023 non-null float64

9 Потребление смолы, г/м2 1023 non-null float64

10 Угол нашивки, град 1023 non-null int64

11 Шаг нашивки 1023 non-null float64

12 Плотность нашивки 1023 non-null float64

dtypes: float64(12), int64(1)

Пропущенных данных в датасете нет.

**1.2. Описание используемых методов**

Для решения поставленной задачи предполагается исследование следующих методов:

1. Метод ближайших соседей (KNN);
2. Линейная регрессия;
3. Градиентный бустинг;
4. Случайный лес.

Алгоритм K-ближайших соседей (KNN) – это тип управляемого алгоритма ML, который может использоваться как для классификации, так и для задач прогнозирования регрессии.

Ближайшие соседи – одна из простейших предсказательных моделей. В данной модели не делается никаких математических допущений, а также не требуется какой-либо тяжелый вычислительный механизм.

Алгоритм находит расстояния между запросом и всеми примерами в данных, выбирая определенное количество примеров (**k**), наиболее близких к запросу, затем голосует за наиболее часто встречающуюся метку (в случае задачи классификации) или усредняет метки (в случае задачи регрессии).

Преимущества:

* Алгоритм прост и легко реализуем.
* Не чувствителен к выбросам.
* Нет необходимости строить модель, настраивать несколько параметров или делать дополнительные допущения.
* Алгоритм универсален. Его можно использовать для обоих типов задач: классификации и регрессии.

Недостатки:

* Алгоритм работает значительно медленнее при увеличении объема выборки, предикторов или независимых переменных.
* Из аргумента выше следуют большие вычислительные затраты во время выполнения.
* Всегда нужно определять оптимальное значение **k**.

2. Линейная регрессия

Линейная регрессия — один из самых простых и популярных алгоритмов машинного обучения. Это статистический метод, который используется для предиктивного анализа. Линейная регрессия делает прогнозы для непрерывных/реальных или числовых переменных, таких как продажи, зарплата, возраст, цена продукта и т. д.

Алгоритм линейной регрессии показывает линейную связь между зависимой (y) и одной или несколькими независимыми (y) переменными, поэтому называется линейной регрессией. Поскольку линейная регрессия показывает линейную зависимость, это означает, что она находит, как значение зависимой переменной изменяется в соответствии со значением независимой переменной.

Модель линейной регрессии представляет собой наклонную прямую линию, представляющую взаимосвязь между переменными.

Модель линейной регрессии помогает прогнозировать значение зависимой переменной, а также может помочь объяснить, насколько точен прогноз. Это определяется значениями параметров R-квадрат и p-значение. Значение R-квадрат указывает, какая часть вариации зависимой переменной может быть объяснена независимой переменной, а p-значение объясняет, насколько надежно это объяснение. Значения R-квадрата варьируются между 0 и 1. Значение 0,8 означает, что независимая переменная может объяснить 80 процентов вариации наблюдаемых значений зависимой переменной. Значение 1 означает, что можно сделать идеальный прогноз, что редко встречается на практике. Значение 0 означает, что независимая переменная совсем не помогает в прогнозировании зависимой переменной. Используя p-значение, вы можете проверить, насколько сильно независимая переменная влияет на зависимую по сравнению с 0.

Главный недостаток линейной регрессии состоит в том, что она может моделировать только прямые линейные зависимости, в то время как часто возникает необходимость создания модели других типов отношений между данными.  К счастью, существуют простые способы отображения данных, между которыми нет линейной зависимости, с помощью линейной регрессии. Первый способ – **преобразование исходных данных.** Вместо того, чтобы использовать переменные для создания модели в их исходном виде, на практике часто приходится использовать различные преобразования, такие как определение логарифма значений или возведение в степень. Даже если между значениями нет прямой линейной зависимости, она может существовать между логарифмами этих значений. В таком случае модель покажет, что при увеличении зависимой переменной на 1% целевая переменная увеличится на X%.

Другой способ предполагает включение в модель **квадратных членов.** Для этого данные возводятся в квадрат и добавляются в уравнение как дополнительная переменная. Еще одна технология преобразования данных предполагает учет **взаимодействия предикторов,** когда исходные переменные рассматриваются в комбинациях.

Модели линейной регрессии очень популярны в различных сферах исследований благодаря быстроте их создания и простоте интерпретации. Благодаря возможности преобразования данных, они могут быть использованы для моделирования широкого спектра зависимостей. Благодаря простоте формы, по сравнению с нейронными сетями, их статистические параметры легко поддаются анализу и сравнению, что позволяет извлекать из них ценную информацию.  Линейная регрессия используется не только в прогностических целях; она также показала свою эффективность в описании систем.

3. Градиентный бустинг

Градиентный бустинг – это продвинутый алгоритм машинного обучения для решения задач классификации и регрессии. Он строит предсказание в виде ансамбля слабых предсказывающих моделей, которыми в основном являются деревья решений. Из нескольких слабых моделей в итоге собирается одну, но уже эффективную. Общая идея алгоритма – последовательное применение предиктора (предсказателя) таким образом, что каждая последующая модель сводит ошибку предыдущей к минимуму.

Параметры алгоритма

* loss – [функция ошибки](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_%D0%BE%D1%88%D0%B8%D0%B1%D0%BE%D0%BA) для минимизации.
* criterion – критерий выбора расщепления, Mean Absolute Error (MAE) или Mean Squared Error (MSE). Используется только при построении деревьев.
* init – какой алгоритм мы будем использовать в качестве главного (именно его и улучшает техника бустинга).
* learning\_rate – скорость обучения.
* n\_estimators – число итераций в бустинге. Чем больше, тем лучше качество, однако слишком большой увеличение данного параметра может привести к ухудшению производительности и переобучению.
* min\_samples\_split – минимальное число объектов, при котором происходит расщепление. С данным параметром мы можем избежать переобучение.
* min\_samples\_leaf – минимальное число объектов в листе (узле). При увеличении данного параметра качество модели на обучении падает, в то время как время построения модели сокращается. Меньшие значения стоит выбирать для менее сбалансированных выборок.
* max\_depth – максимальная глубина дерева. Используется для того, чтобы исключить возможность переобучения.
* max\_features – количество признаков, учитываемых алгоритмом для построения расщепления в дереве.
* max\_leaf\_nodes : Максимальное число верхних точек в дереве. При наличии данного параметра max\_depth будет игнорироваться.

Алгоритм градиентного бустинга может использоваться при следующих условиях:

* Наличие большого количества наблюдений (ближайшее сходство) в тренировочной выборке данных.
* Количество признаков меньше количества наблюдений в обучающих данных. Бустинг хорошо работает, когда данные содержат смесь числовых и категориальных признаков или только числовые признаки.
* Когда необходимо рассмотреть метрики производительности модели.

Бустинг не следует использовать:

* В задачах распознавания изображений и компьютерного зрения (CV – Computer Vision).
* В обработке и понимании естественного языка (NLP – Natural Language Processing).
* Когда число обучающих выборок значительно меньше, чем число признаков (фич).

Плюсы алгоритма:

* Алгоритм работает с любыми функциями потерь.
* Предсказания в среднем лучше, чем у других алгоритмов.
* Самостоятельно справляется с пропущенными данными.

Минусы алгоритма:

* Алгоритм крайне чувствителен к выбросам и при их наличии будет тратить огромное количество ресурсов на эти моменты. Однако, стоит отметить, что использование Mean Absolute Error (MAE) вместо Mean Squared Error (MSE) значительно снижает влияние выбросов на вашу модель (выбор функции в параметре criterion).
* Ваша модель будет склонна к переобучению при слишком большом количестве деревьев. Данная проблема присутствует в любом алгоритме, связанном с деревьями, и справляется правильной настройкой параметра n\_estimators.
* Вычисления могут занять много времени. Поэтому, если у вас большой набор данных, всегда составляйте правильный размер выборки и не забывайте правильно настроить параметр min\_samples\_leaf.

4.Случайный лес

Алгоритм случайного леса (Random Forest) — универсальный алгоритм машинного обучения, суть которого состоит в использовании ансамбля решающих деревьев. Само по себе решающее дерево предоставляет крайне невысокое качество классификации, но из-за большого их количества результат значительно улучшается. Также это один из немногих алгоритмов, который можно использовать в абсолютном большинстве задач.

Благодаря своей гибкости Random Forest применяется для решения практически любых проблем в области машинного обучения. Сюда относятся классификации (RandomForestClassifier) и регрессии (RandomForestRegressor), а также более сложные задачи, вроде отбора признаков, поиска выбросов/аномалий и кластеризации.

Необходимые параметры алгоритма:

- Число деревьев – n\_estimators

Чем больше деревьев, тем лучше качество. Стоит отметить, что время настройки и работы Random Forest будут пропорционально увеличиваться, что может сказаться на производительности.

Часто при большом увеличении n\_estimators качество на обучающей выборке может даже доходить до 100%, в то время как качество на тесте выходит на [асимптоту](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D0%B8%D0%BC%D0%BF%D1%82%D0%BE%D1%82%D0%B0), что сигнализирует о переобучении нашей модели. Лучший способ избежать этого – прикинуть, сколько деревьев вам достаточно, зафиксировав момент, когда качество теста еще не становится стабильно-неизменным.

- Критерий расщепления – criterion

Также один из самых важных параметров для построения, но без значительной возможности выбора. В библиотеке sklearn для задач классификации реализованы критерии gini и entropy. Они соответствуют классическим критериям расщепления: джини и энтропии.

В свою очередь, для задач регрессии реализованы два критерия (mse и mae), которые являются функциями ошибок Mean Square Error и Mean Absolute Error соответственно. Практически во всех задачах используется критерий mse.

Простой метод перебора поможет выбрать, что использовать для решения конкретной проблемы.

- Число признаков для выбора расщепления – max\_features

При увеличении max\_features увеличивается время построения леса, а деревья становятся похожими друг на друга. В задачах классификации он по умолчанию равен sqrt(n), в задачах регрессии – n/3. Является одним из самых важных параметров в алгоритме.

- Минимальное число объектов для расщепления – min\_samples\_split

Второстепенный по своему значению параметр, его можно оставить в состоянии по умолчанию.

-Ограничение числа объектов в листьях – min\_samples\_leaf

Аналогично с min\_samples\_split, но при увеличении данного параметра качество модели на обучении падает, в то время как время построения модели сокращается.

**- Максимальная глубина деревьев – max\_depth**

Чем меньше максимальная глубина, тем быстрее строится и работает алгоритм случайного дерева.

При увеличении глубины резко возрастает качество как на обучении модели, так и на ее тестировании. Если у вас есть возможность и время для построения глубоких деревьев, то рекомендуется использовать максимальное значение данного параметра.

Неглубокие деревья рекомендуется использовать в задачах со значительным количеством шумовых объектов (выбросов).

Преимущества алгоритма:

* Высокая точность предсказания, которая сравнима с результатами градиентного бустинга.
* Не требует тщательной настройки параметров, хорошо работает из коробки.
* Практически не чувствителен к выбросам в данных из-за случайного семплирования (random sample).
* Не чувствителен к масштабированию и к другим монотонным преобразованиям значений признаков.
* Редко переобучается. На практике добавление деревьев только улучшает композицию.
* В случае наличия проблемы переобучения, она преодолевается путем усреднения или объединения результатов различных деревьев решений.
* Способен эффективно обрабатывать данные с большим числом признаков и классов.
* Хорошо работает с пропущенными данными – сохраняет хорошую точность даже при их наличии.
* Одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки
* Высокая параллелизуемость и масштабируемость.

## **Недостатки алгоритма**

## Для реализации алгоритма случайного дерева требуется значительный объем вычислительных ресурсов.

* Большой размер моделей.
* Построение случайного леса отнимает больше времени, чем деревья решений или линейные алгоритмы.
* Алгоритм склонен к переобучению на зашумленных данных.
* Нет формальных выводов, таких как p-values, которые используются для оценки важности переменных.
* В отличие от более простых алгоритмов, результаты случайного леса сложнее интерпретировать.
* Когда в выборке очень много разреженных признаков, таких как тексты или наборы слов (bag of words), алгоритм работает хуже, чем линейные методы.
* В отличие от линейной регрессии, Random Forest не обладает возможностью [экстраполяции](https://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D1%82%D1%80%D0%B0%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D1%8F). Это можно считать и плюсом, так как в случае выбросов не будет экстремальных значений.
* Если данные содержат группы признаков с корреляцией, которые имеют схожую значимость для меток, то предпочтение отдается небольшим группам перед большими, что ведет к недообучению.
* Процесс прогнозирования с использованием случайных лесов очень трудоемкий по сравнению с другими алгоритмами.

**1.3. Разведочный анализ данных**

Разведочный анализ — это предварительный анализ данных с целью выявления наиболее общих зависимостей, закономерностей и тенденций, характера и свойств анализируемых данных, законов распределения анализируемых величин. Применяется для нахождения связей между переменными в ситуациях, когда отсутствуют (или недостаточны) априорные представления о природе этих связей.

Как правило, при разведочном анализе учитывается и сравнивается большое число [признаков](https://wiki.loginom.ru/articles/attribute.html), а для поиска закономерностей используются самые разные методы.

Термин «разведочный анализ» был впервые введен математиком из Принстонского университета [Дж. Тьюки](https://ru.wikipedia.org/wiki/%D0%A2%D1%8C%D1%8E%D0%BA%D0%B8,_%D0%94%D0%B6%D0%BE%D0%BD). Он также сформулировал основные цели данного анализа:

* Максимальное «проникновение» в данные.
* Выявление основных структур.
* Выбор наиболее важных переменных.
* [Обнаружение отклонений](https://wiki.loginom.ru/articles/deviation-detection.html) и [аномалий](https://wiki.loginom.ru/articles/outlier.html).
* Проверка основных [гипотез](https://wiki.loginom.ru/articles/hypothesis.html) (предположений).
* Разработка начальных моделей.

Результаты разведочного анализа не используются для выработки управленческих решений. Их назначение — помощь в разработке наилучшей стратегии углубленного анализа, выдвижение гипотез, уточнение особенностей применения тех или иных математических методов и моделей. Без разведочного анализа углубленный анализ данных будет производиться практически «вслепую».

К основным методам разведочного анализа относится процедура анализа распределений переменных, [корреляционный анализ](https://wiki.loginom.ru/articles/correlation-analysis.html) c целью поиска [коэффициентов](https://wiki.loginom.ru/articles/correlation-coefficient.html), превосходящих по величине определенные пороговые значения, [факторный анализ](https://wiki.loginom.ru/articles/factorial-analysis.html), [дискриминантный анализ](https://wiki.loginom.ru/articles/linear-discriminant-analysis.html), многомерное шкалирование, визуальный анализ [гистограмм](https://wiki.loginom.ru/articles/histogram.html) и т.д.

Предварительное исследование данных может служить лишь первым этапом в процессе их анализа, и пока результаты не подтверждены на других [выборках](https://wiki.loginom.ru/articles/sample.html) или на независимом множестве данных, их следует воспринимать самое большее как гипотезу. Если результаты разведочного анализа говорят в пользу некоторой модели, то ее правильность можно затем проверить, применив к новым данным.

Разведочный анализ данных представленного датасета.

На первом этапе следует найти и удалить пропущенные значения. В представленном датасете пропущенных значений не оказалось, что проиллюстрировано тепловой картой.

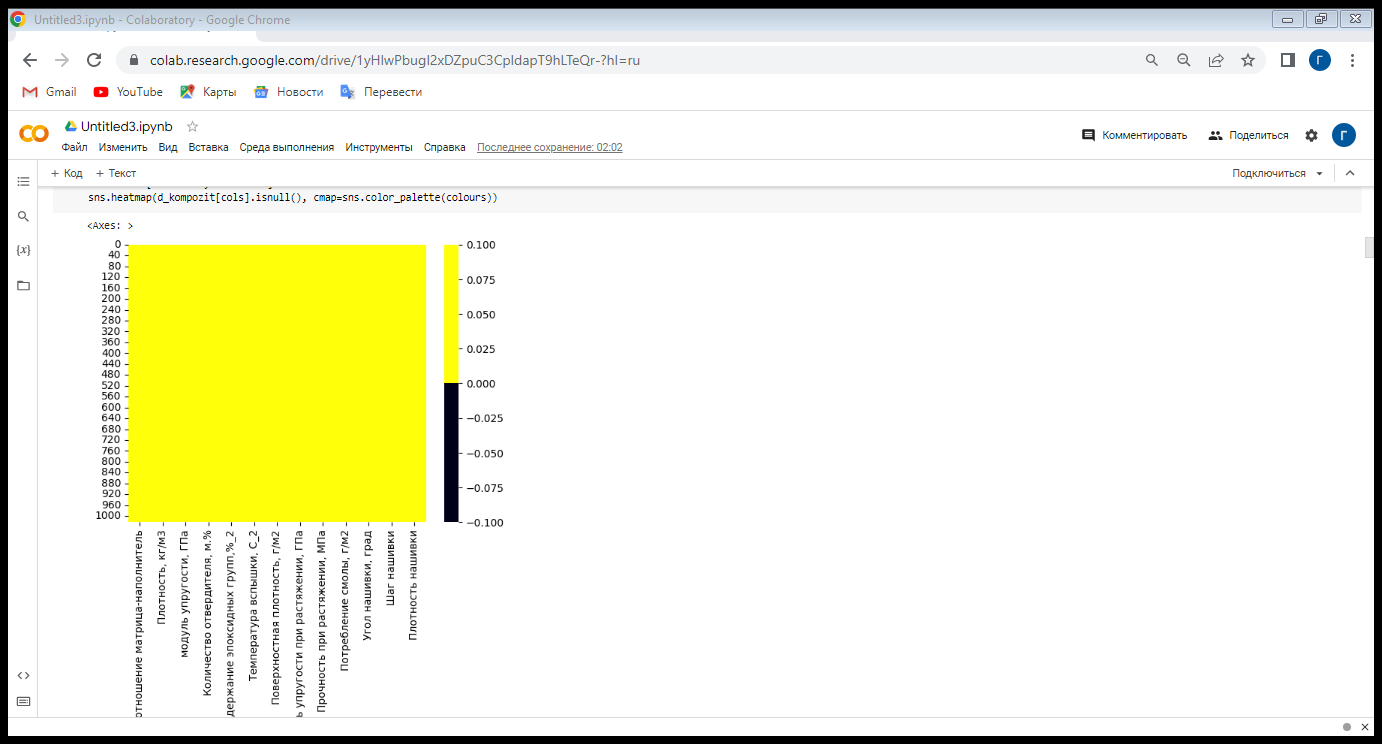


Рис. 1. Тепловая карта пропущенных значений (черный цвет-пропущенные значения)

Важнейшим этапом разведочного анализа данных является выявление и устранение выбросов. В Приложении 1 представлены гистограммы распределения, в Приложении 2 – «ящик с усами».

Для определения выбросов в данной работе используется метод z-оценки и метод межквартильного диапазона. В первом было обнаружено 24 выброса. Во втором случае был обнаружен 1 выброс. Количество выбросов незначительно, поэтому их можно удалить.

Далее исследуется, есть ли какая-либо корреляция (зависимость) между отдельными функциями (столбцами), которые нам необходимо учитывать. Некоторые модели, такие как Наи́вный ба́йесовский классифика́тор, используют допущение об отсутствии корреляции между отдельными характеристиками, поэтому этот шаг имеет решающее значение.

Результатом метода corr()является таблица с большим количеством чисел, которые представляют, насколько хорошо связаны отношения между двумя столбцами. Число варьируется от -1 до 1.

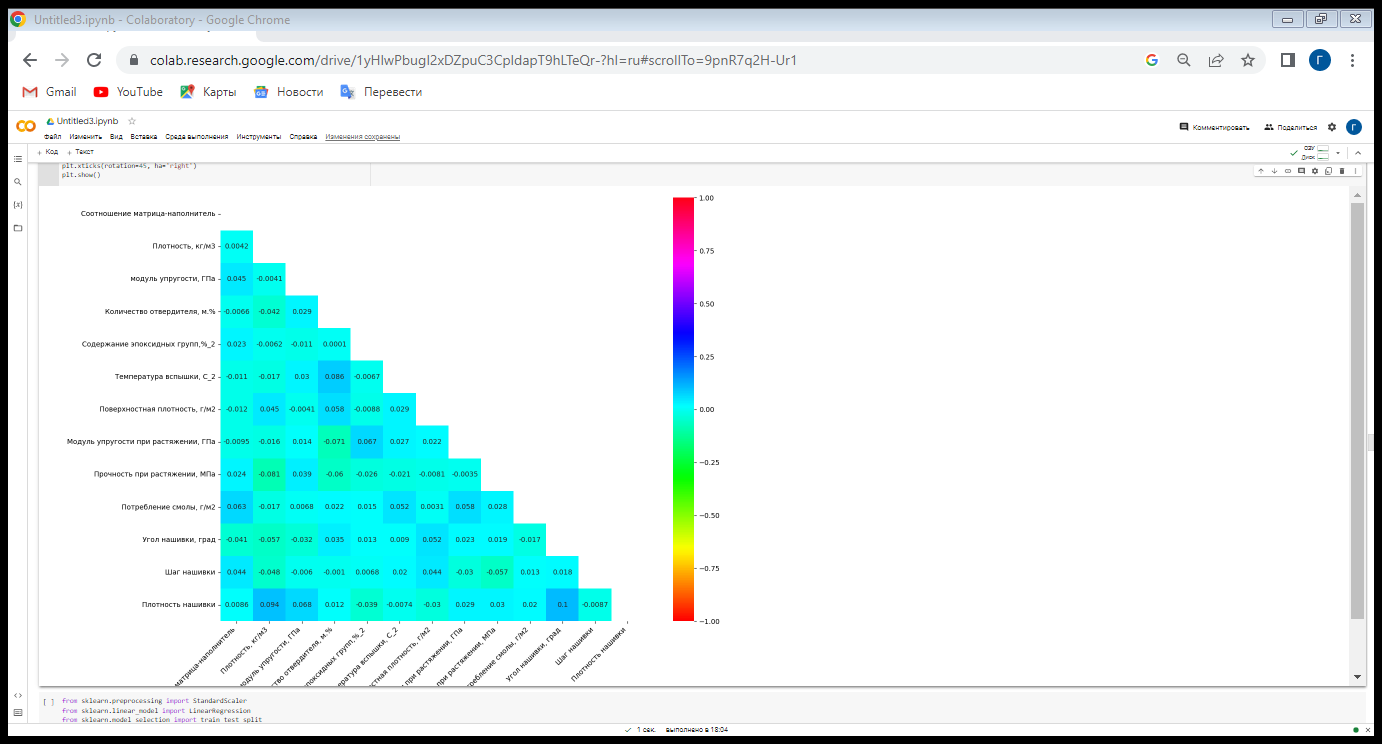
1 означает, что существует отношение 1 к 1 (идеальная корреляция), и для этого набора данных каждый раз, когда значение в первом столбце увеличивалось, другое значение также увеличивалось.

0,9 также является хорошим соотношением, и если вы увеличите одно значение, то, вероятно, увеличится и другое.

-0,9 будет таким же хорошим соотношением, как и 0,9, но если вы увеличите одно значение, другое, вероятно, уменьшится.

02 означает, что если одно значение повышается, это не означает, что другое будет расти.

Линейная зависимость между анализируемыми признаками в исследуемом датасете отсутствует, что иллюстрирует матрица корреляции.



**Рис.2. Матрица корреляции**

Мультиколлинеарность — корреляция независимых переменных, которая затрудняет оценку и анализ общего результата. Когда независимые переменные коррелируют друг с другом, говорят о возникновении мультиколлинеарности.

В машинном обучении мультиколлинеарность может стать причиной переобучаемости модели, что приведет к неверному результату. Кроме того, избыточные коэффициенты увеличивают сложность модели машинного обучения, а значит, время ее тренировки возрастает. Еще мультиколлинеарность факторов плоха тем, что математическая модель регрессии содержит избыточные переменные, а это значит:

* осложняется интерпретация параметров множественной регрессии как величин действия факторов, параметры регрессии теряют смысл и следует рассматривать другие переменные;
* оценки параметров ненадежны – получаются большие стандартные ошибки, которые меняются с изменением объема наблюдений, что делает модель регрессии непригодной для прогнозирования.

Для оценки мультиколлинеарности используется матрица парных коэффициентов корреляции, у которой необходимо вычислить определитель. При этом возможны следующие ситуации:

* у совсем не коррелирующих факторов матрица парных коэффициентов корреляции единичная, у которой все элементы вне ее главной диагонали равны нулю;
* если между факторами определилась абсолютно линейная зависимость и все коэффициенты корреляции равняются единице, то определитель такой матрицы равен нулю;
* чем определитель меньше (ближе к нулю), тем сильнее мультиколлинеарность факторов и ненадежнее результаты множественной регрессии;
* чем определитель ближе к единице, тем меньше мультиколлинеарность факторов.

Две переменных коллинеарны, когда они находятся между собой в линейной зависимости, если коэффициент корреляции более 0,7. Чтобы избавиться от мультиколлинеарности, необходимо исключить из модели один из факторов. Например, в эконометрике исключается фактор, который при сильной связи с результатом имеет наибольшую тесноту связи с другими переменными.

**2. Практическая часть**

* 1. **Обучение алгоритма машинного обучения**

Очищенный от выбросов разделен на обучающую и тестовую выборку (999, 300).

Алгоритмы машинного обучения, как правило, работают лучше или сходятся быстрее, когда различные функции (переменные) имеют меньший масштаб. Поэтому перед обучением на них моделей машинного обучения данные обычно нормализуются.

Нормализация также делает процесс обучения менее чувствительным к масштабу функций. Это приводит к улучшению коэффициентов после тренировки.

Этот процесс повышения пригодности функций для обучения путем изменения масштаба называется масштабированием функций.

Одним из способов нормализации данных в Sklearn является [MinMaxScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler).

Это более популярный выбор для нормализации наборов данных.

scaler = preprocessing.MinMaxScaler()

columns = data.columns

data\_norm = scaler.fit\_transform(np.array(data))

data\_norm = pd.DataFrame(data\_norm, columns=columns)

data\_norm.head()

В результате получена следующая таблица:

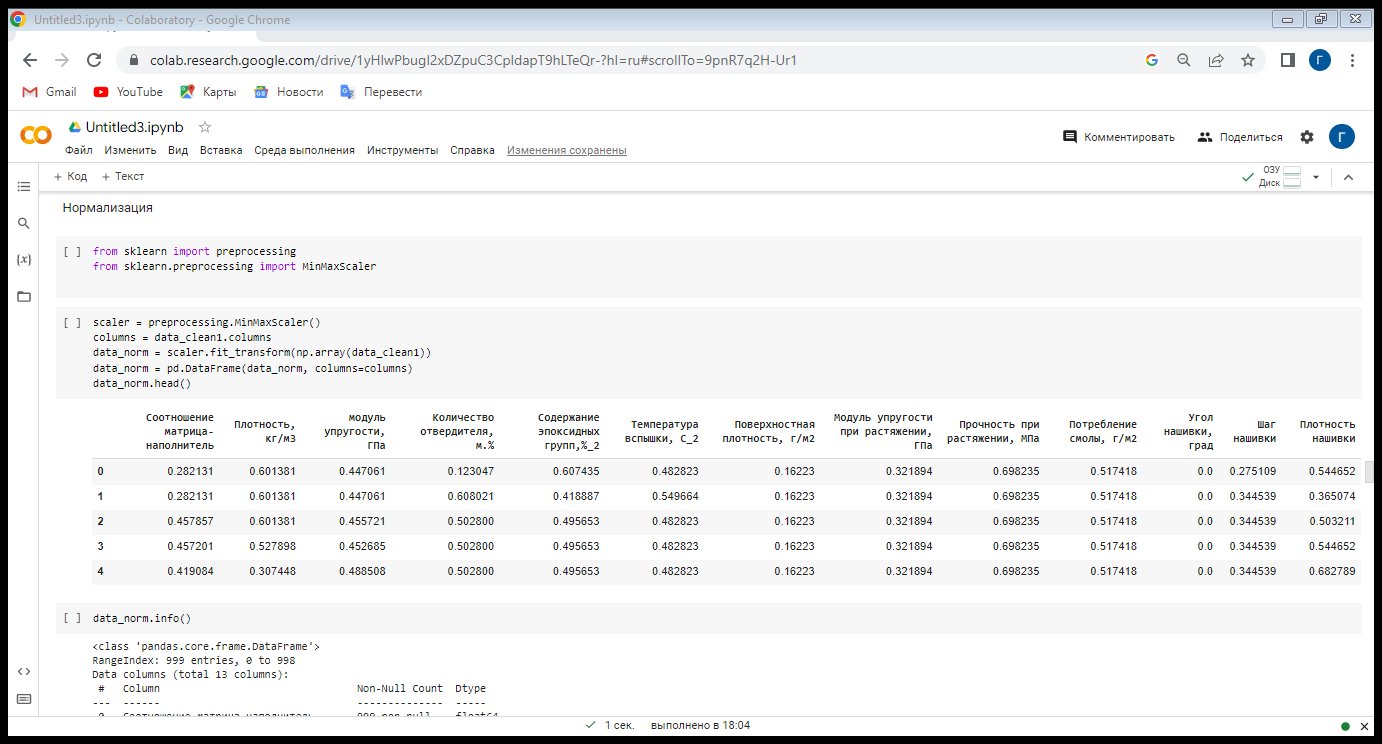


Рис.3. Датасет после нормализации данных

Необходимо обучить алгоритм машинного обучения, который будет определять значения:

* Модуль упругости при растяжении, ГПа
* Прочность при растяжении, Мпа

Для каждого параметра будем использовать следующие модели:

1. Метод ближайших соседей;
2. Случайный лес;
3. Линейная регрессия;
4. Градиентный бустинг.

Особенности каждой модели, ее преимущества и недостатки рассмотрены в первой части данной работы.

Метод ближайших соседей для Модуля упругости при растяжении показал следующие результаты:

MSE: 11.913689834066206

MAE: 2.7735225664891776

R2: -0.2823483783385832

Линейная регрессия:

linear\_train linear\_test

R2 0.023781 -0.011552

mae 381.888430 380.397595

mse 232279.347113 227787.171261

rmse 481.953677 477.270543

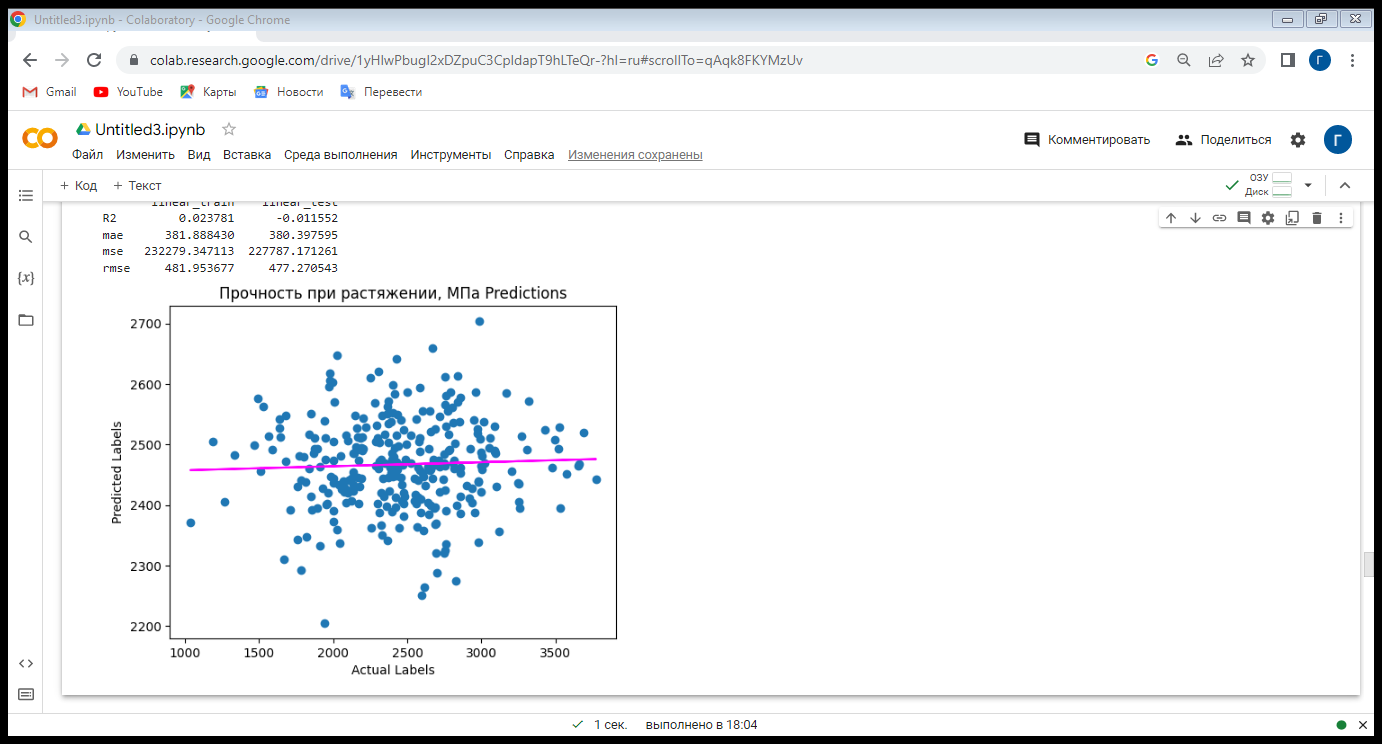


Рис. 3. График линейной регрессии

R2-score случайного леса составил -0.03. R2 градиентного бустинга: -0.9.

Таким образом, R2 обученных моделей для модуля упругости при растяжении принимает отрицательные значения.

Отрицательный коэффициент детерминации указывает на то, что увеличение одной переменной соответствует уменьшению другой переменной. Модели не имеют прогностической ценности.

Используем те же модели для определения модуля прочности при растяжении.

Результат метода ближайших соседей:

MSE: 227787.17126110577

MAE: 380.3975950693701

R2: -0.011552320567429275

Коэффициент детерминации градиентного бустинга составил -0.038.

R2-score случайного леса: -0.03.

Результат обучения модели линейной регрессии:

linear\_train linear\_test

R2 0.023781 -0.011552

mae 381.888430 380.397595

mse 232279.347113 227787.171261

rmse 481.953677 477.270543

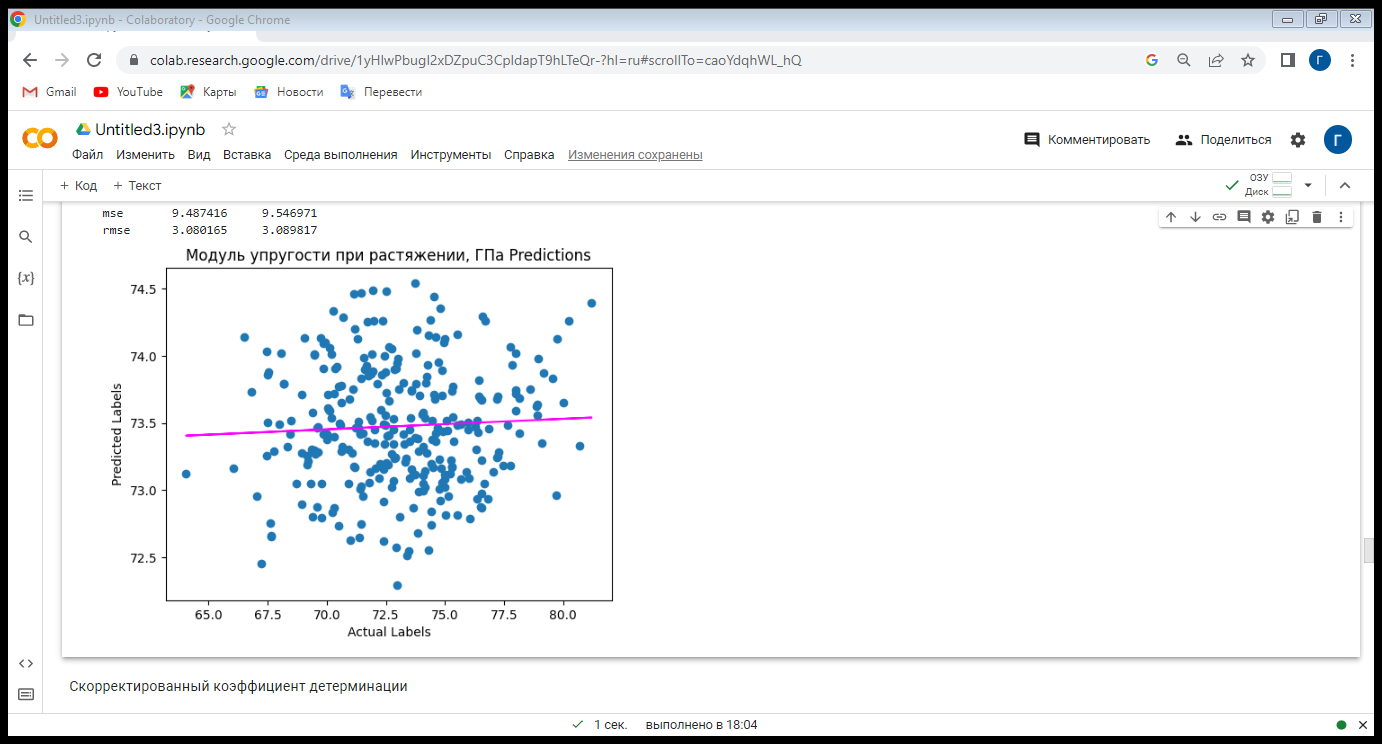


Рис.4. Линейная регрессия

Таким образом, коэффициент детерминации отрицательный, ни одна из обученных моделей не работает.

* 1. **Нейронная сеть, которая рекомендует соотношение матрица-наполнитель**

Нейронная сеть построена на Keras. Sequential – последовательная модель.

model.add(Dense(128, activation='tanh', input\_shape=(X\_train.shape[1],)))

model.add(Dense(64, activation='tanh'))

model.add(Dense(1, activation='sigmoid'))

Активационные функции — гиперболический тангенс и сигмоида.

Сигмоида стремиться привести значения к одной из сторон кривой (например, к верхнему при х=2 и нижнему при х=-2). Такое поведение позволяет находить четкие границы при предсказании.

Другое преимущество сигмоиды над линейной функцией заключается в следующем. В первом случае имеем фиксированный диапазон значений функции — [0,1], тогда как линейная функция изменяется в пределах (-inf, inf). Такое свойство сигмоиды очень полезно, так как не приводит к ошибкам в случае больших значений активации.

Гиперболический тангенс очень похож на сигмоиду **- это скорректированная сигмоидная функция.**

Результат:

Mean Squared Error: 4.899943090898793

Mean Absolute Error: 2.024359294279085

R-squared Score: -4.990564477075089.

Результат получился неудовлетворительным, поэтому целесообразно использовать другие функции активации, а также изменить количество слоев.

model = Sequential()

model.add(Dense(512, activation='softmax', input\_shape=(X\_train.shape[1],)))

model.add(Dense(256, activation='softmax'))

model.add(Dense(1, activation='sigmoid'))

Кроме того, увеличено количество эпох – 150.

Результат:

Mean Squared Error: 0.03028229869573374

Mean Absolute Error: 0.13569690740445164

R-squared Score: -0.0019887514237220127.

Результат лучше, чем предыдущий, однако R2 отрицательный.

model = Sequential()

model.add(Dense(64, activation='tanh', input\_shape=(X\_train.shape[1],)))

model.add(Dense(128, activation='tanh'))

model.add(Dense(1, activation='sigmoid')).

Количество эпох 100.

Результат:

Mean Squared Error: 0.030128028473482248

Mean Absolute Error: 0.13534187770458747

R-squared Score: 0.0031157827111786274.

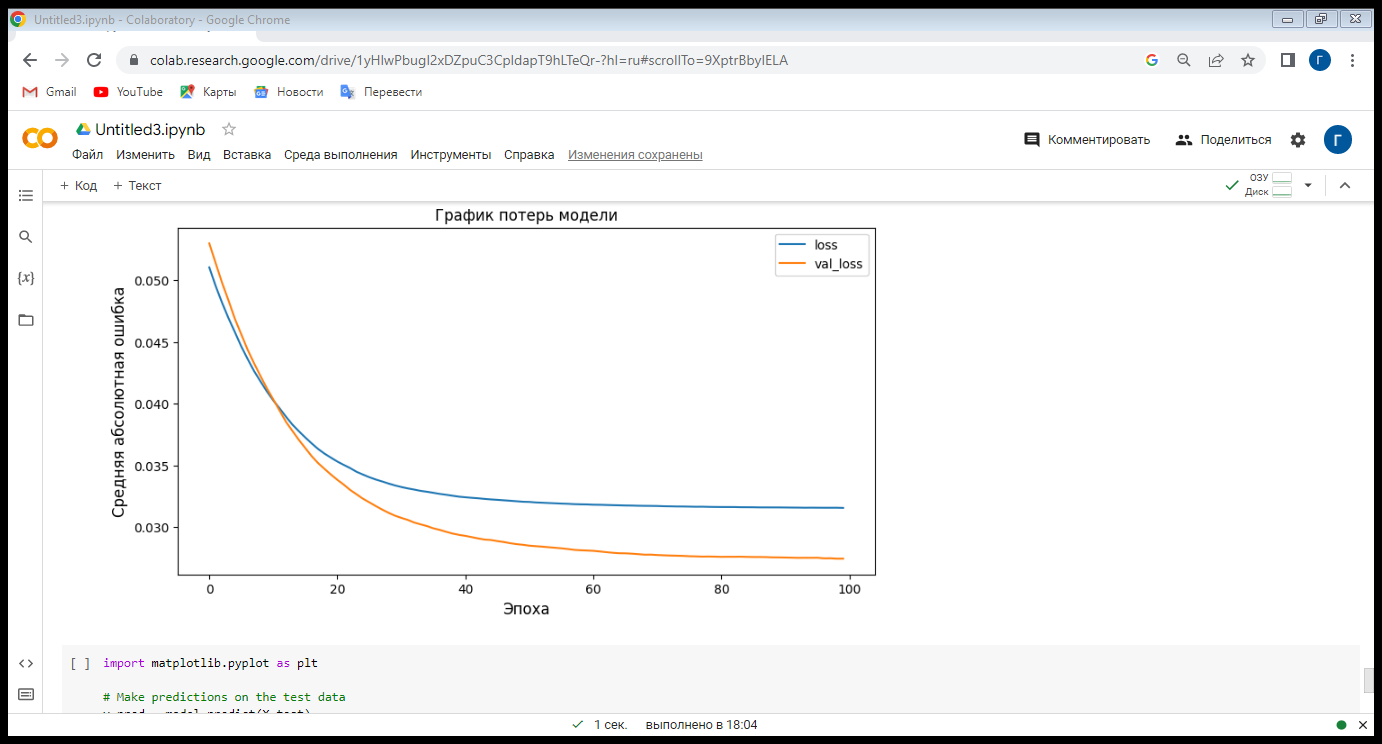
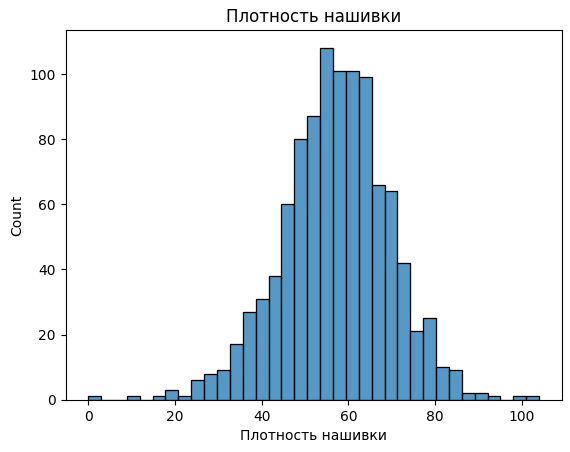
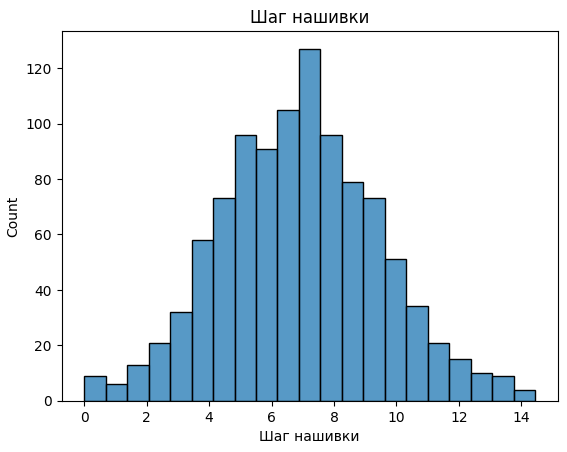
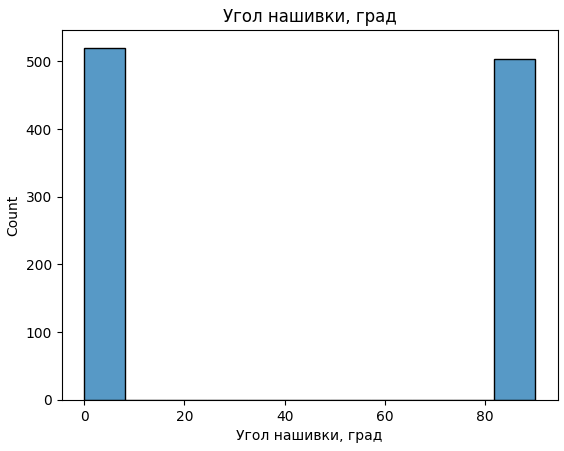
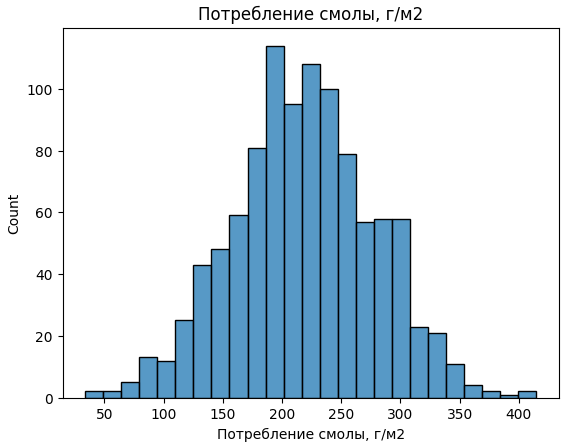
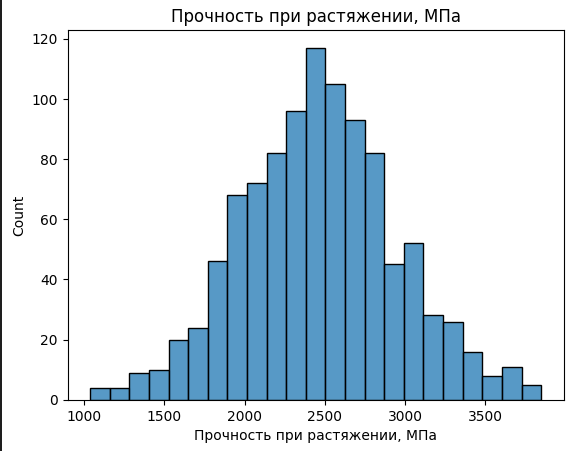
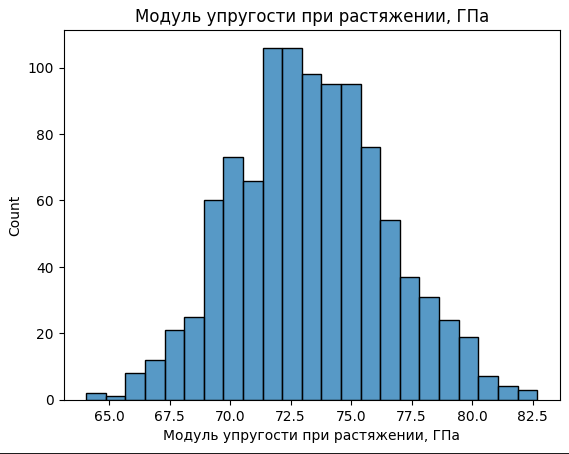
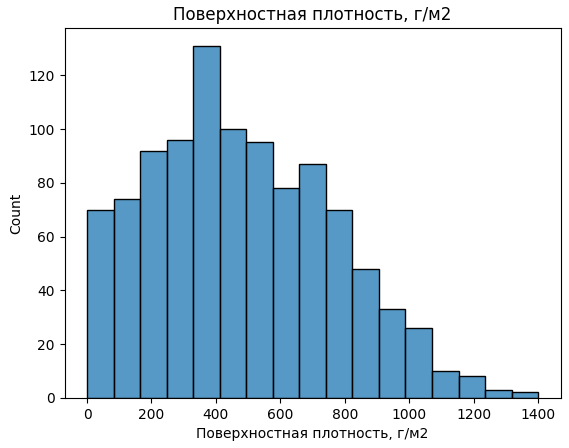
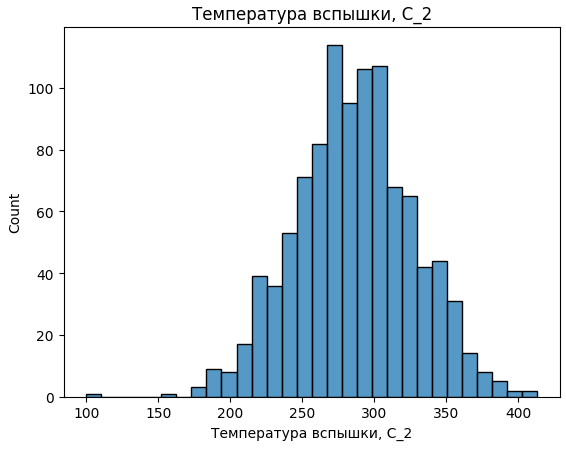
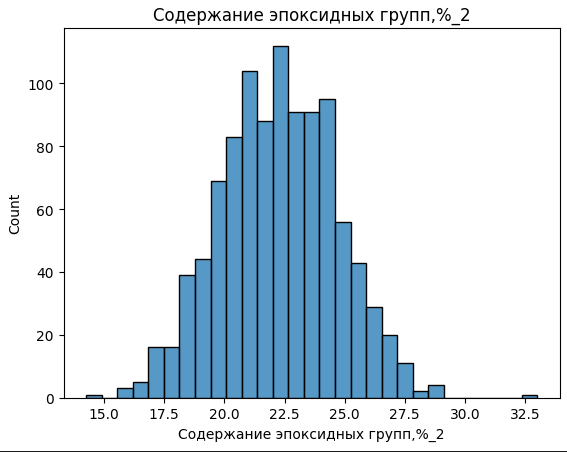
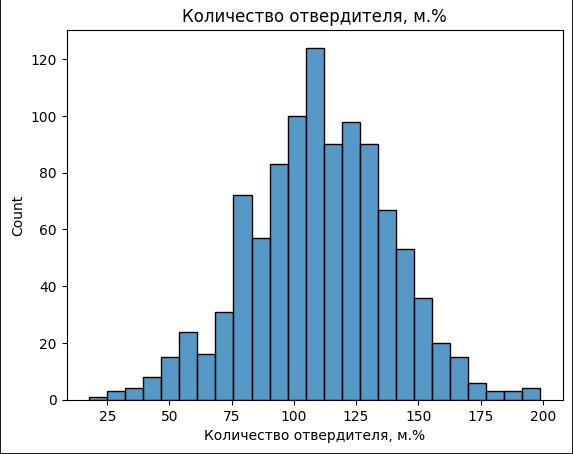
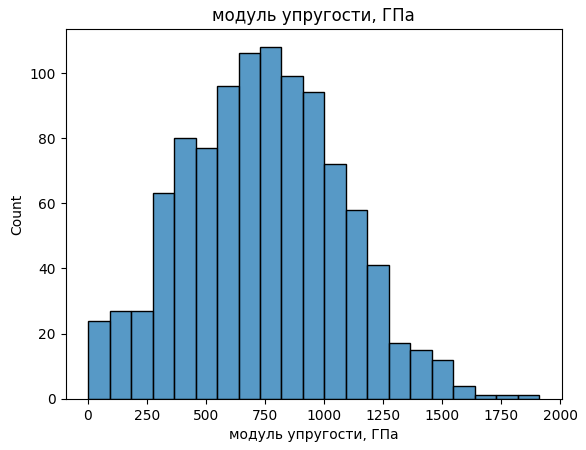
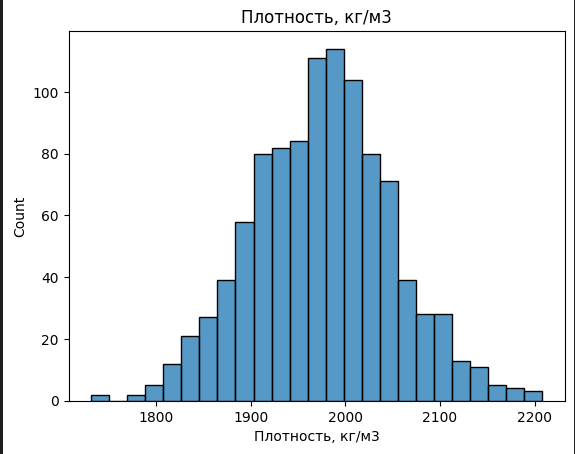
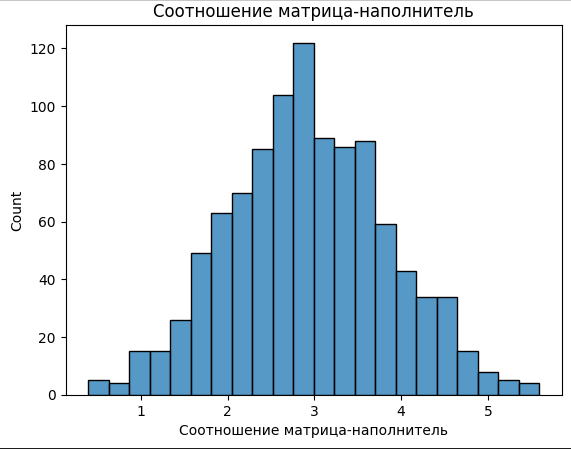


Рис.5. График потерь модели

Приложение 1



Приложение 2

