

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

Васильев Александр Иванович

Москва, 2023

Содержание

Введение	4
1 Аналитическая часть	5
1.1 Постановка задачи.	5
1.2 Описание используемых методов машинного обучения	6
1.2.1 Линейная регрессия LinearRegression().....	6
1.2.2 Регрессор дерева решений DecisionTreeRegressor().....	7
1.2.3 Регрессор случайного леса RandomForestRegressor().....	8
1.2.4 Гребневая регрессия Ridge()	9
1.2.5 Регрессия Лассо Lasso()	9
1.2.6 Эластичная сеть ElasticNet().....	10
1.2.7 Регрессия опорных векторов SVR()	10
1.2.8 Стохастический градиентный спуск SGDRegressor().....	11
1.2.9 Регрессия k-ближайших соседей KNeighborsRegressor()	12
1.3 Нейронная сеть.....	13
1.4 Разведочный анализ данных	15
2 Практическая часть.....	20
2.1 Предобработка данных	20
2.2 Разработка и обучение модели.....	24
2.2.1 Модель машинного обучения для параметра «Модуль упругости при растяжении, ГПа»	26

2.2.2 Модель машинного обучения для параметра «Прочность при растяжении, МПа»	32
2.3 Разработка нейронной сети.....	39
2.4 Разработка приложения	44
2.5 Создание репозитория	47
Заключение	48
Библиографический список	49
Приложение 1	52

Введение

Темой выпускной квалификационной работы является прогнозирование конечных свойств новых материалов (композиционных материалов).

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).

Актуальность: 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.

Предоставлен набор данных (датасет) о свойствах компонентов композиционных материалов. Датасет состоит из двух таблиц в формате Excel (.xlsx).

Необходимо решить следующие задачи:

- 1) объединить файлы датасета по индексу (тип объединения INNER);
- 2) провести разведочный анализ и предобработку данных;
- 3) обучить алгоритм машинного обучения, который будет прогнозировать значения следующих параметров:
 - а) «модуль упругости при растяжении, ГПа»;
 - б) «прочность при растяжении, МПа»;
- 4) написать нейронную сеть, которая будет рекомендовать параметр «соотношение матрица-наполнитель»;
- 5) написать приложение, которое будет выдавать прогноз, полученный в задачах 2 и 3;
- 6) создать репозиторий в GitHub и разместить там код исследования.

Данные получены из реальных производственных задач Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

При разделении датасета на обучающую и тестирующую выборки выделим 70% данных на обучение моделей, на остальных проведем тестирование.

1.2 Описание используемых методов машинного обучения

Прогнозирование неизвестной величины на основании одной или нескольких других случайных величин является типичной задачей регрессии. Следовательно, мы будем применять регрессионные модели обучения.

В данной работе автор использует следующие методы машинного обучения:

- 1) линейная регрессия `LinearRegression()`;
- 2) регрессор дерева решений `DecisionTreeRegressor()`;
- 3) регрессор случайного леса `RandomForestRegressor()`;
- 4) гребневая регрессия `Ridge()`;
- 5) регрессия Лассо `Lasso()`;
- 6) эластичная сеть `ElasticNet()`;
- 7) регрессия опорных векторов `SVR()`;
- 8) стохастический градиентный спуск `SGDRegressor()`;
- 9) регрессия k-ближайших соседей `KNeighborsRegressor()`.

1.2.1 Линейная регрессия `LinearRegression()`

Линейная регрессия – это модель линейной зависимости одной (зависимой) переменной от другой или нескольких.

Линейная регрессия подгоняет линейную модель с коэффициентами $w=(w_1, \dots, w_p)$ к минимизации остаточной суммы квадрата между наблюдаемым целевым признаком в наборе данных и предсказанным целевым признаком по линейной аппроксимации.

Преимущества:

- скорость и простота реализации;
- легкость понимания и интерпретации;
- имеет меньшую сложность по сравнению с другими алгоритмами.

Недостатки:

- моделирует только прямые линейные зависимости;

- требует прямую связь между зависимыми и независимыми переменными;
- выбросы оказывают огромное влияние, а границы линейны.

1.2.2 Регрессор дерева решений `DecisionTreeRegressor()`

Дерево решений — это непараметрический метод обучения с учителем, используемый для классификации и регрессии. Цель - создать модель, которая предсказывает значение целевой переменной путем изучения простых правил принятия решений, выведенных из характеристик данных.

Преимущества деревьев решений:

- легкость понимания и интерпретации;
- несложная подготовка данных;
- стоимость прогнозирования данных логарифмически зависит от количества точек данных, используемых для обучения дерева;
- обработка числовых и категориальных данных;
- способен обрабатывать проблемы с несколькими выводами;
- использование модели белого ящика, когда условия легко объясняются булевой логикой;
- возможность проверки модели с помощью статистических тестов;

Недостатки деревьев решений:

- риск переобучения (могут создаваться слишком сложные деревья, которые плохо обобщают данные);
- деревья решений могут быть нестабильными (небольшие отклонения в данных могут привести к созданию совершенно другого дерева);
- предсказания деревьев решений являются кусочно-постоянными приближениями. Поэтому они не очень подходят для экстраполяции;
- известно, что проблема обучения оптимального дерева решений является NP-полной при нескольких аспектах оптимальности и даже для простых концепций. Следовательно, практические алгоритмы

обучения дерева решений основаны на эвристических алгоритмах, таких как жадный алгоритм, где локально оптимальные решения принимаются на каждом узле. Такие алгоритмы не могут гарантировать возврат глобально оптимального дерева решений.;

- есть моменты, препятствующие обучению, например, «исключающее ИЛИ», четность или мультиплексор;
- могут создаваться предвзятые деревья, если некоторые классы доминируют. Рекомендуется проводить балансировку датасета перед применением дерева решений.

1.2.3 Регрессор случайного леса `RandomForestRegressor()`

Случайный лес — это мета-алгоритм, который применяет ансамбль деревьев решений для различных подвыборок датасета и использует усреднение для повышения точности прогнозирования и контроля переобучения. Размер подвыборки определяется параметром `max_samples`, если `bootstrap=True` (по умолчанию), в противном случае для построения каждого дерева используется весь набор данных.

Установка по умолчанию значений параметров, контролирующих размер деревьев (например, `max_depth`, `min_samples_leaf` и т. д.), приводит к максимальному ветвлению алгоритма. Для экономии вычислительных ресурсов и памяти, рекомендуется настраивать сложность и размер деревьев, устанавливая значения этих параметров.

Признаки случайным образом перемешиваются при каждом ветвлении. Следовательно, наилучший найденный сплит может различаться даже при одинаковых обучающих данных. Следует фиксировать параметр `random_state`.

Преимущества:

- эффективная обработка данных с большим числом признаков и классов;
- нечувствительность к масштабированию значений признаков;

- одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки. Существуют методы построения деревьев по данным с пропущенными значениями признаков;
- методы оценивания значимости отдельных признаков в модели;
- внутренняя оценка способности модели к обобщению (тест по неотобраным образцам);
- высокая параллелизуемость и масштабируемость.

Недостатки:

- большой размер получающихся моделей.

1.2.4 Гребневая регрессия Ridge()

Гребневая регрессия — это вариация линейной регрессии, похожая на регрессию Lasso. Она применяет сжатие и хорошо работает для данных, которые демонстрируют сильную мультиколлинеарность.

Отличие от Lasso в том, что гребневая регрессия использует регуляризацию L2, которая взвешивает ошибки по их квадрату, чтобы сильнее наказывать за более значительные ошибки.

1.2.5 Регрессия Лассо Lasso()

Регрессия Лассо — это вариация линейной регрессии, адаптированная для данных, имеющих сильную корреляцию признаков друг с другом.

Лассо использует сжатие коэффициентов (shrinkage) и этим пытается уменьшить сложность данных, искривляя пространство, на котором они лежат.

В этом процессе лассо автоматически помогает устранить или исказить сильно коррелированные и избыточные функции в методе с низкой дисперсией.

Регрессия лассо использует регуляризацию L1, то есть взвешивает ошибки по их абсолютному значению.

Лассо оценивает разреженные коэффициенты. Это полезно в некоторых контекстах из-за своей тенденции отдавать предпочтение решениям с меньшим

количеством ненулевых коэффициентов, эффективно уменьшая количество функций, от которых зависит данное решение. По этой причине лассо и его варианты являются фундаментальными для области сжатого зондирования. При определенных условиях он может восстановить точный набор ненулевых коэффициентов.

1.2.6 Эластичная сеть ElasticNet()

Эластичная сеть ElasticNet() — это вариация линейной регрессии, которая использует комбинированную регуляризацию L1 и L2 (сочетает в себе свойства как регрессии Ridge, так и регрессии Lasso).

Этот метод минимизирует сложность регрессионной модели (величины и числа коэффициентов регрессии), штрафует модель, используя как L2-норму, так и L1-норму.

1.2.7 Регрессия опорных векторов SVR()

Регрессия опорных векторов — это вариация метода опорных векторов (SVM), которую можно использовать для выполнения как линейной, так и нелинейной регрессии. Цель применения регрессии опорных векторов — убедиться, что ошибки не превышают пороговое значение. В SVR мы помещаем как можно больше экземпляров между строками, ограничивая при этом нарушение полей.

В основе метода лежит поиск гиперплоскости, при которой риск в многомерном пространстве будет минимальным. По сравнению с традиционной регрессионной моделью SVR оценивает коэффициенты путем минимизации квадратичных потерь.

Преимущества:

- задача выпуклого квадратичного программирования хорошо изучена и имеет единственное решение;

- метод опорных векторов эквивалентен двухслойной нейронной сети, где число нейронов на скрытом слое определяется автоматически как число опорных векторов;
- принцип оптимальной разделяющей гиперплоскости приводит к максимизации ширины разделяющей полосы, а следовательно, к более уверенной классификации.

Недостатки классического SVM:

- неустойчивость к шуму: выбросы в исходных данных становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости;
- не описаны общие методы построения ядер и спрямляющих пространств, наиболее подходящих для конкретной задачи;
- нет отбора признаков;
- необходимо подбирать константу C при помощи кросс-валидации.

1.2.8 Стохастический градиентный спуск `SGDRegressor()`

Стохастический градиентный спуск (SGD) — это эффективный метод подгонки линейных классификаторов и регрессоров под выпуклые функции потерь.

Стохастический градиентный спуск представляет собой итерационный метод оптимизации целевой функции с подходящими свойствами гладкости. Его можно рассматривать как стохастическое приближение оптимизации градиентного спуска, поскольку он заменяет фактический градиент (рассчитанный на основе всего набора данных) его оценкой (рассчитанной на основе случайно выбранного подмножества данных). Особенно в задачах многомерной оптимизации это снижает очень высокую вычислительную нагрузку, обеспечивая более быстрые итерации в обмен на более низкую скорость сходимости.

1.2.9 Регрессия k-ближайших соседей KNeighborsRegressor()

Регрессия на основе соседей может использоваться в тех случаях, когда метки данных являются непрерывными, а не дискретными переменными. Метка, присвоенная точке запроса, вычисляется на основе среднего значения меток ее ближайших соседей. Базовая регрессия ближайших соседей использует одинаковые веса: то есть каждая точка в локальной окрестности вносит одинаковый вклад в классификацию точки запроса. При некоторых обстоятельствах может быть выгодно взвешивать точки таким образом, чтобы близлежащие точки вносили больший вклад в регрессию, чем удаленные точки.

Преимущества:

- алгоритм прост и легко реализуем;
- не чувствителен к выбросам;
- нет необходимости строить модель, настраивать несколько параметров или делать дополнительные допущения;

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 .

1.3 Нейронная сеть

Искусственная нейронная сеть – это математическая модель (её программное или аппаратное воплощение), созданная по принципу организации биологических нейронных сетей (человеческий мозг).

Искусственная нейронная сеть представляет собой систему соединённых между собой простых процессоров (искусственных нейронов). Каждый процессор этой сети имеет дело только с сигналами, которые он получает, и сигналами, которые он посылает другим процессорам.

Технически обучение заключается в нахождении коэффициентов связей между нейронами. Сигналы суммируются с учётом значимости (веса) каждого входа. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными и выходными сигналами.

В процессе обучения участвуют также такие факторы, как смещение (дополнительный вход для нейрона, который всегда равен 1) и функция активации, которая определяет выходное значение нейрона. Последняя используется для того, чтобы ввести нелинейность в нейронную сеть.

Способности нейронной сети к прогнозированию напрямую следуют из её способности к обобщению и выделению скрытых зависимостей между входными и выходными данными.

Для построения нейронной сети возможно использовать библиотеки «PyTorch» и «Tensorflow». «PyTorch» проста в изучении, может вычислять на тензорах благодаря ускорению графического процессора. Но у «TensorFlow» больше инструментов для полноценной работы, и он чаще используется для нейронных сетей.

Распространённым видом модели («TensorFlow») является стек слоев – последовательная модель `Sequential()`. Это линейный набор слоев нейронной сети.

В качестве слоя модели можно использовать полносвязный слой `Dense()`. Этот слой обрабатывает каждый элемент предыдущего слоя, выполняя

матричное перемножение этих элементов со своими весами. Полученные данные затем отправляются на следующий слой.

Преимущества полносвязных слоёв:

- возможность изучения любой функции, которая может быть представлена в виде математической модели;
- нейроны полносвязных слоев могут адаптироваться к любым входным данным, обучаясь выделять важные признаки и игнорировать незначительные;
- универсальность, они могут использоваться в различных типах нейронных сетей;
- полносвязные слои просты в использовании и реализации.

К недостаткам полносвязных слоёв относится их способность переобучаться, если есть много параметров и мало данных.

После каждого слоя `Dense()` применяют вспомогательный слой `Dropout` в качестве метода регуляризации модели с целью предотвращения переобучения.

В данной работе автор использует последовательную модель `Sequential()` из библиотеки «Tensorflow» с пятью полносвязными слоями `Dense()`, четырьмя вспомогательными слоями `Dropout`. Во входном и внутренних слоях используется функция активации «relu», в выходном слое – «linear».

1.4 Разведочный анализ данных

Разведочный анализ данных (Exploratory Data Analysis) – первый шаг в обработке данных.

Сырые, необработанные данные ненадежны. В них могут быть различные аномалии и выбросы, данные могут быть искажены. Зачастую данные просто отсутствуют. Все это может привести к неадекватным результатам алгоритмов машинного обучения.

Предварительное изучение, обработка и очистка данных должны проводиться до того, как набор данных будет использоваться для обучения модели.

Разведочный анализ данных означает исследование датасета с целью определения его основных характеристик, наличия аномалий, пропусков данных, взаимосвязей между признаками, изучение данных для получения из них практической информации.

В качестве инструментов анализа и визуализации данных используются оценка статистических характеристик датасета; гистограммы распределения признаков; диаграммы «ящик с усами»; попарные графики рассеяния точек; квантиль-квантильный график; тепловая карта взаимной корреляции признаков; проверка наличия пропусков и дубликатов.

Нами получен датасет из двух файлов: X_bp.xlsx (10 признаков, 1023 строки) и X_nup.xlsx (3 признака, 1040 строк).

Объединим данные файлы по индексу (тип объединения INNER) с помощью следующей команды:

```
df = X_bp.join(X_nup, how='inner')
```

В результате объединения получен датафрейм, в котором содержится 13 признаков и 1023 строки (Рисунок 1).

```

1 [10]: df.info()

<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, C_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)
memory usage: 111.9 KB

```

Рисунок 1 – Общая информация о датафрейме

На рисунке 1 видно, что все переменные содержат значения float64, за исключением признака «Угол нашивки, град», у которого целочисленный тип int64, качественные характеристики отсутствуют; пропусков нет.

На рисунке 2 представлено содержание уникальных значений в датасете.

```

In [12]: df.nunique()

Out[12]: Соотношение матрица-наполнитель          1014
          Плотность, кг/м3                          1013
          модуль упругости, ГПа                     1020
          Количество отвердителя, м.%              1005
          Содержание эпоксидных групп,%_2          1004
          Температура вспышки, C_2                 1003
          Поверхностная плотность, г/м2            1004
          Модуль упругости при растяжении, ГПа     1004
          Прочность при растяжении, МПа            1004
          Потребление смолы, г/м2                  1003
          Угол нашивки, град                       2
          Шаг нашивки                              989
          Плотность нашивки                         988
          dtype: int64

In [13]: df['Угол нашивки, град'].unique()

Out[13]: array([ 0, 90])

```

Рисунок 2 – Уникальные значения

На рисунке 2 видно, что признак «Угол нашивки, град» имеет два уникальных значения: 0 и 90. Этот признак будет рассматриваться нами как категориальный, и в дальнейшем подвергнется обработке методом LabelEncoder().

На Рисунке 3 видно, что данные в датафрейме имеют разный масштаб. В дальнейшем будем проводить масштабирование данных.

```
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, C_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 3 – Основные статистические характеристики данных

Далее из гистограмм распределения параметров (Рисунок 4) видим, что признаки имеют нормальное распределение, за исключением признака «Угол нашивки, град».

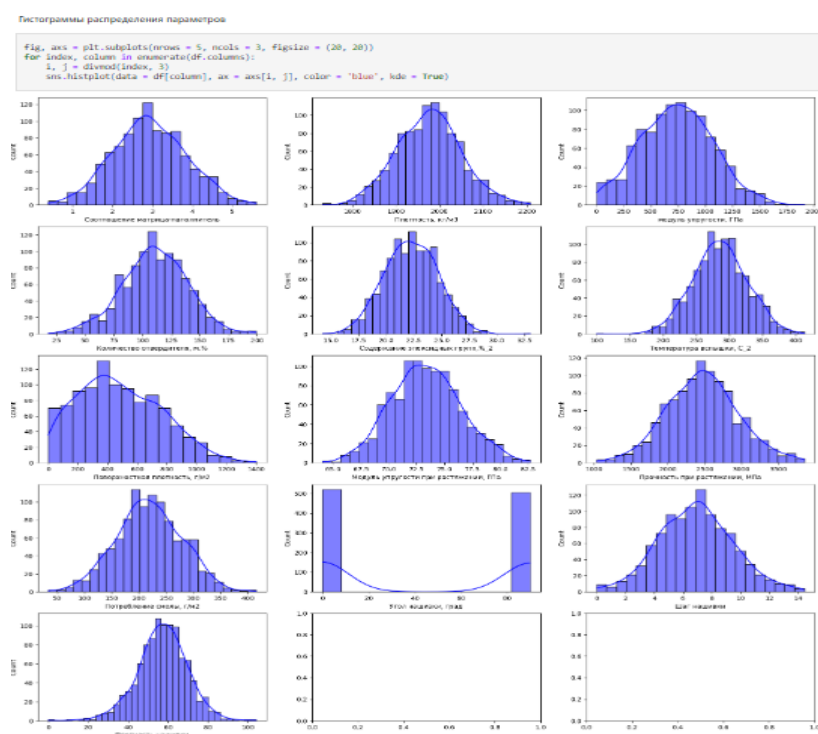


Рисунок 4 – Гистограмма распределения параметров

Попарные графики рассеяния (Рисунок 5) и график боксплот («ящик с усами»), построенный для датасета, показывают, что имеются выбросы.

Попарные графики рассеяния (Рисунок 5) и тепловая карта (Рисунок 6) демонстрируют низкую (околонулевую) корреляцию признаков.

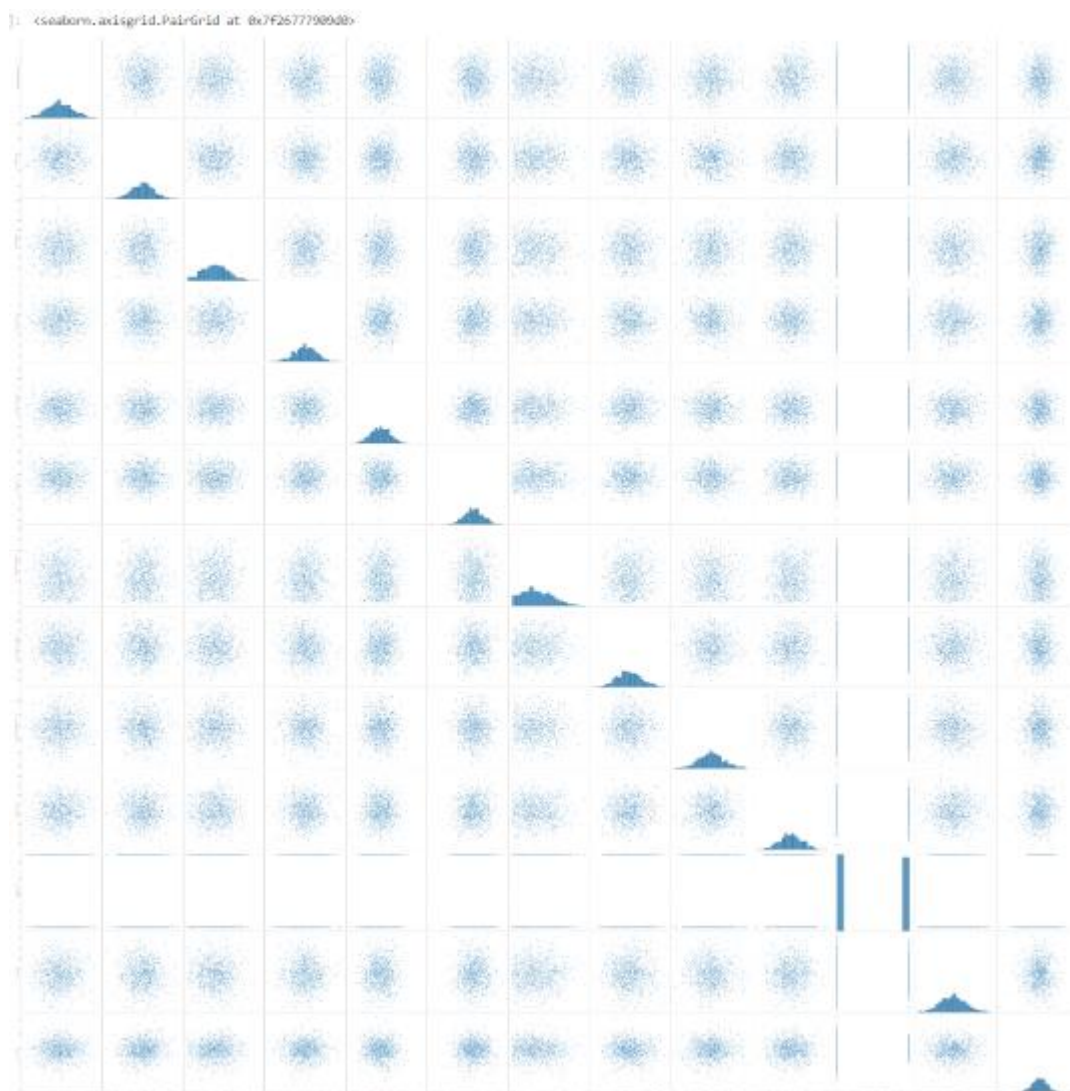


Рисунок 5 - Попарные графики рассеяния

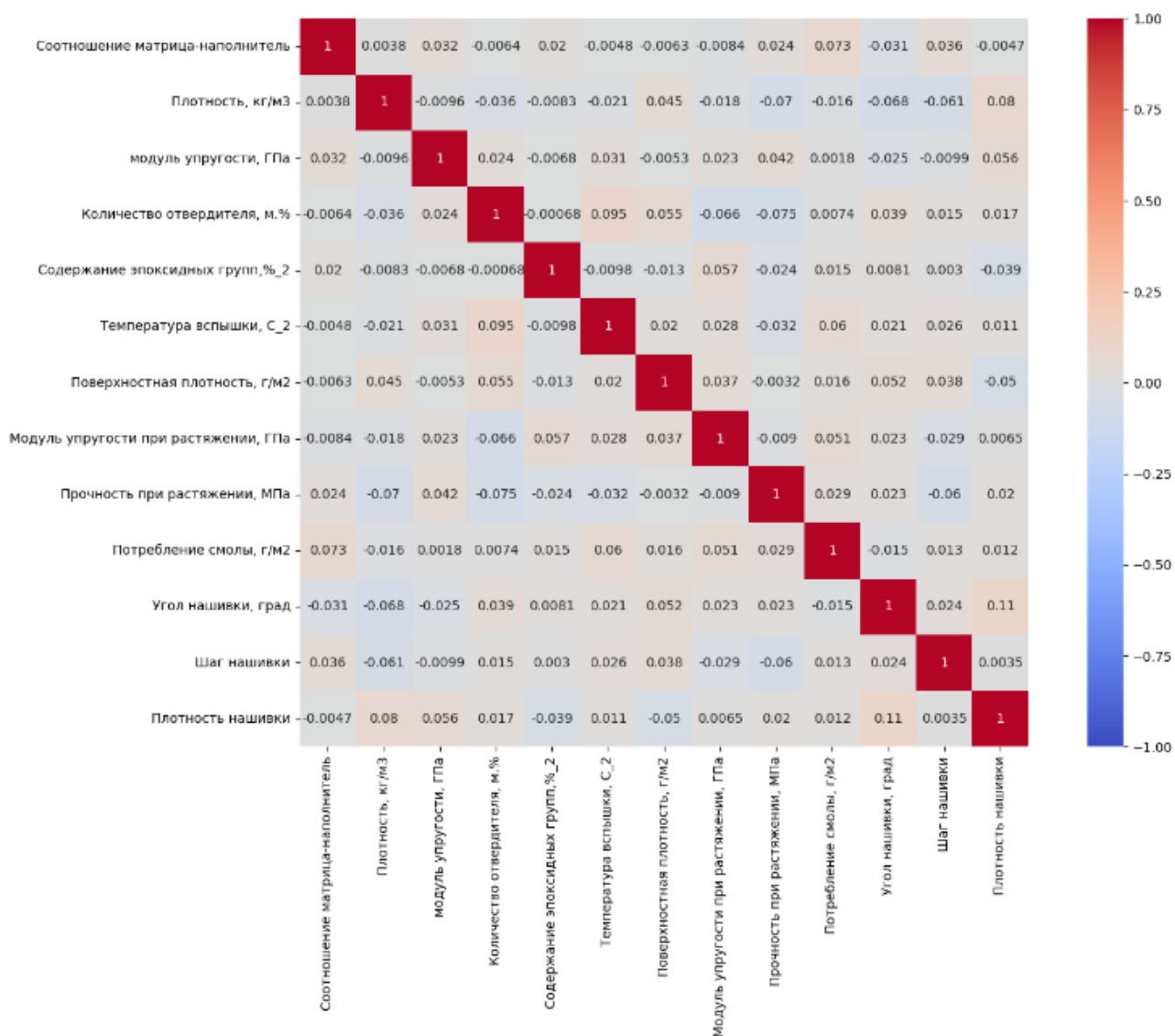


Рисунок 6 - Тепловая карта корреляции

2 Практическая часть

2.1 Предобработка данных

После разведочного анализа данных нужно провести их предобработку для того, чтобы модели машинного обучения могли наиболее эффективно обучаться.

Подготовка данных состоит из нескольких этапов:

- обработка недостающих данных;
- удаление данных, не имеющих статистической значимости, и признаков, находящихся в прямой линейной зависимости;
- кодирование категориальных признаков;
- поиск и удаление выбросов (аномалий);
- разделение датасета на тренировочную и тестирующую части;
- нормализация или стандартизация данных.

Разведочный анализ данных показал, что пустых значений нет, удалять данные, не имеющие статистической значимости, и признаки, находящихся в прямой линейной зависимости не нужно.

Параметры соответствуют нормальному распределению, и не коррелируют друг с другом (корреляция близка к 0).

Но есть категориальные признаки (Рисунок 2), выбросы (Рисунок 5), а также масштаб некоторых признаков различается (Рисунок 3).

Начнем с кодирования категориальных данных (Рисунок 7).

```
Кодирование категориальных данных

: # Ранее были обнаружены два уникальных значения в параметре "Угол нашивки, град" - 0 и 90. Закодируем их с помощью LabelEncoder.

le = LabelEncoder()
df['Угол нашивки, град'] = le.fit_transform(df['Угол нашивки, град'])

: # Посмотрим, что получилось

df['Угол нашивки, град'].unique()

: array([0, 1], dtype=int64)
```

Рисунок 7 – Кодирование категориальных данных

В результате кодирования с помощью метода LabelEncoder() мы получили признак «Угол нашивки, град» со значениями 0 и 1.

Далее приступим к удалению выбросов.

Попарные графики рассеяния (Рисунок 5) и диаграммы "ящик с усами", выполненный в Jupyter Notebook, показали, что во всех параметрах, за исключением "Угол нашивки, град", есть выбросы.

Для подсчета количества выбросов используем стандартизированную оценку (z-score) и межквартильное расстояние (IQR).

Удаление выбросов

```
# Диаграмма "ящик с усами" выше показал, что во всех параметрах, за исключением "Угол нашивки, град", есть выбросы.  
# Для подсчета количества выбросов используем стандартизированную оценку (z-score) и межквартильное расстояние (IQR).  
  
z_score = 0  
iqr = 0  
  
for column in df:  
  
    z = stats.zscore(df[column])  
    z = z.abs() > 3  
    print(column, ' , z-score: ', z.sum())  
    z_score += z.sum()  
  
    q1 = df[column].quantile(0.25)  
    q3 = df[column].quantile(0.75)  
    iqr_1 = q3 - q1  
    lower = q1 - 1.5 * iqr_1  
    upper = q3 + 1.5 * iqr_1  
    iqr_r = (df[column] <= lower) | (df[column] >= upper)  
    print(column, ' , IQR: ', iqr_r.sum())  
    iqr += iqr_r.sum()  
  
print('Стандартизированная оценка (z-score), выбросов:', z_score)  
print('Межквартильное расстояние (IQR), выбросов:', iqr)
```

Соотношение матрица-наполнитель , z-score: 0
Соотношение матрица-наполнитель , IQR: 6
Плотность, кг/м3 , z-score: 3
Плотность, кг/м3 , IQR: 9
модуль упругости, ГПа , z-score: 2
модуль упругости, ГПа , IQR: 2
Количество отвердителя, м.%, z-score: 2
Количество отвердителя, м.%, IQR: 14
Содержание эпоксидных групп,%_2 , z-score: 2
Содержание эпоксидных групп,%_2 , IQR: 2
Температура вспышки, C_2 , z-score: 3
Температура вспышки, C_2 , IQR: 8
Поверхностная плотность, г/м2 , z-score: 2
Поверхностная плотность, г/м2 , IQR: 2
Модуль упругости при растяжении, ГПа , z-score: 1
Модуль упругости при растяжении, ГПа , IQR: 6
Прочность при растяжении, МПа , z-score: 0
Прочность при растяжении, МПа , IQR: 11
Потребление смолы, г/м2 , z-score: 3
Потребление смолы, г/м2 , IQR: 8
Угол нашивки, град , z-score: 0
Угол нашивки, град , IQR: 0
Шаг нашивки , z-score: 0
Шаг нашивки , IQR: 4
Плотность нашивки , z-score: 7
Плотность нашивки , IQR: 21
Стандартизированная оценка (z-score), выбросов: 25
Межквартильное расстояние (IQR), выбросов: 93

Рисунок 8 - Подсчет количества выбросов

На рисунке 8 мы видим, что метод z-score выдал 25 выбросов, а метод IQR подсчитал 93 выброса.

Так как у нас относительно небольшой датасет, наша задача оставить как можно больше данных. Поэтому удалим выбросы методом z-score.

```
# Удалим выбросы по методу z-score.

z_score_clean = pd.DataFrame(index = df.index)
for column in df:
    z = stats.zscore(df[column])
    z_score_clean[column] = z.abs() > 3
df_clean = df[z_score_clean.sum(axis=1)==0]

# Посмотрим форму очищенного датасета.
df_clean.shape

(999, 13)
```

Рисунок 9 – Удаление выбросов

В результате удаления выбросов мы получили датасет из 13 признаков и 999 строк (Рисунок 9).

Далее идет масштабирование данных. Применим метод StandardScaler().

Рекомендуется проводить масштабирование данных после их разделения на обучающую (train) и тестовую (test) выборки во избежание их взаимного влияния друг на друга. А пока определим метод и функции для отображения данных до и после масштабирования (Рисунок 10).

```
scaler = StandardScaler()

# Создадим функцию для отображения данных до и после масштабирования.
def statistics_before(data1):
    print('До масштабирования')
    return data1.describe().loc[['min', 'max', 'mean', 'std'], :].style.format(precision=6)

def statistics_after(data2):
    print('После масштабирования')
    return pd.DataFrame(data2.describe().loc[['min', 'max', 'mean', 'std'], :].style.format(precision=6))
```

Рисунок 10 - Масштабирование

С помощью методов машинного обучения нам необходимо спрогнозировать два признака: «Модуль упругости при растяжении, ГПа» и

«Прочность при растяжении, МПа». Следовательно, нам понадобятся два комплекта целевых данных. Входные данные могут быть общими для обоих признаков.

Проведём разбиение датасета на входные (X) и целевые (y) данные в двух вариантах, для каждого целевого параметра.

```
# Проведём разбиение датасета на входные (X) и целевые (y) данные в двух вариантах, для каждого целевого параметра.

# Целевой параметр - "Модуль упругости при растяжении, ГПа"
y1 = df_clean.iloc[:, df_clean.columns == 'Модуль упругости при растяжении, ГПа']

# Целевой параметр - "Прочность при растяжении, МПа"
y2 = df_clean.iloc[:, df_clean.columns == 'Прочность при растяжении, МПа']

# Входные параметры
X = df_clean.drop(columns=['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа'])

# Посмотрим форму полученных данных.
print(y1.shape)
print(y2.shape)
print(X.shape)
```

(999, 1)
(999, 1)
(999, 11)

Рисунок 11 – Разбиение данных на X и y

На рисунке 11 показано, что в результате разбиения мы получили три комплекта данных:

- целевые данные y1 для признака «Модуль упругости при растяжении, ГПа» (1 признак, 999 строк);
- целевые данные y2 для признака «Прочность при растяжении, МПа» (1 признак, 999 строк);
- входные данные X (11 признаков, 999 строк)

2.2 Разработка и обучение модели

Ранее в пункте 1.2 говорилось, что прогнозирование неизвестной величины на основании других величин — это задача регрессии.

Следовательно, мы будем применять регрессионные модели обучения.

Модели возьмем из библиотеки `scikit-learn`.

Для начала определим список методов машинного обучения для каждого признака.

```
# Выберем модели регрессии для Модуля упругости при растяжении, ГПа

models_1 = {
    'LinearRegression': LinearRegression(),
    'DecisionTreeRegressor': DecisionTreeRegressor(random_state=42),
    'RandomForestRegressor': RandomForestRegressor(random_state=42),
    'Ridge': Ridge(),
    'Lasso': Lasso(),
    'ElasticNet': ElasticNet(),
    'SVR': SVR(kernel='linear'),
    'SGDRegressor': SGDRegressor(),
    'KNeighborsRegressor': KNeighborsRegressor()
}

# Выберем модели регрессии для Прочности при растяжении, МПа

models_2 = {
    'LinearRegression': LinearRegression(),
    'DecisionTreeRegressor': DecisionTreeRegressor(random_state=42),
    'RandomForestRegressor': RandomForestRegressor(random_state=42),
    'Ridge': Ridge(),
    'Lasso': Lasso(),
    'ElasticNet': ElasticNet(),
    'SVR': SVR(kernel='linear'),
    'KNeighborsRegressor': KNeighborsRegressor()
}
```

Рисунок 12 – Выбор моделей

Указанные на рисунке 12 модели мы применим к нашим данным, подберем гиперпараметры, рассчитаем показатели эффективности и выберем лучшую из них.

В качестве показателей эффективности моделей используем метрики регрессии из библиотеки `scikit-learn`: `r2`, `max_error`, `neg_mean_absolute_error`, `neg_root_mean_squared_error`.

По общему правилу, метрики, заканчивающиеся на error (ошибка), тем лучше, чем они ниже. А метрики, заканчивающиеся на score, а также r2, наоборот, тем лучше, чем они выше.

В задачах регрессии же библиотека Scikit-learn использует `neg_mean_absolute_error`, `neg_root_mean_squared_error`, являющиеся противоположностью MAE и MSE. Тем самым упрощая отслеживание метрик, подчиняя их правилу «чем выше, тем лучше». Более высокие возвращаемые значения метрик лучше, чем более низкие возвращаемые значения.

Определим вспомогательные функции:

```
# Определим функцию для вывода метрик по каждой модели.  
# Используем кросс-валидацию для статистической устойчивости результатов.  
  
def models_metrics (models, x, y):  
    models_metrics_table = pd.DataFrame()  
    cv = KFold(10, shuffle = True, random_state = 42)  
    scoring = ['max_error',  
              'neg_mean_absolute_error',  
              'neg_root_mean_squared_error',  
              'r2']  
    for model_name, model in models.items():  
        scores = cross_validate(model, x, y, cv = cv, scoring = scoring)  
        models_metrics_table.loc[model_name, 'max_error'] = scores['test_max_error'].mean()  
        models_metrics_table.loc[model_name, 'MAE'] = scores['test_neg_mean_absolute_error'].mean()  
        models_metrics_table.loc[model_name, 'RMSE'] = scores['test_neg_root_mean_squared_error'].mean()  
        models_metrics_table.loc[model_name, 'R2'] = scores['test_r2'].mean()  
    return models_metrics_table
```

Рисунок 13 – Функция для вывода метрик

```
# Определим функцию для поиска гиперпараметров по сетке.  
# RMSE – одна из самых популярных метрик в задаче регрессии.  
# Но использовать RMSE для сравнения моделей на выборках с большим количеством выбросов может быть неудобно.  
# В таких случаях прибегают к метрике MAE.  
  
def grid_search(model, parameters, x, y):  
    pd.set_option("max_colwidth", 200)  
    grid_search_table = pd.DataFrame()  
    cv = KFold(10, shuffle=True, random_state=42)  
    scoring = 'neg_mean_absolute_error'  
    searcher = GridSearchCV(model, parameters, cv=cv, scoring=scoring)  
    searcher.fit(x, y)  
    grid_search_table.loc[:, 'parameters'] = searcher.cv_results_['params']  
    grid_search_table.loc[:, 'MAE'] = searcher.cv_results_['mean_test_score']  
    grid_search_table.loc[:, 'rank'] = searcher.cv_results_['rank_test_score']  
    return grid_search_table, searcher.best_estimator_
```

Рисунок 14 – Функция для поиска гиперпараметров по сетке

```
#Определим функцию для расчета метрик выбранной модели.
def best_model_metrics(model, y, y_pred):
    best_model_metrics_table = pd.DataFrame()
    best_model_metrics_table.loc[model, 'max_error'] = metrics.max_error(y, y_pred) * -1
    best_model_metrics_table.loc[model, 'MAE'] = metrics.mean_absolute_error(y, y_pred) * -1
    best_model_metrics_table.loc[model, 'RMSE'] = metrics.mean_squared_error(y, y_pred, squared=False) * -1
    best_model_metrics_table.loc[model, 'R2'] = metrics.r2_score(y, y_pred)
    return best_model_metrics_table
```

Рисунок 15 – Функция для расчета метрик выбранной модели

2.2.1 Модель машинного обучения для параметра «Модуль упругости при растяжении, ГПа»

Разделим данные на обучающую и тестовую выборки.

```
# Разделим данные на train и test
X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X, y1, test_size = 0.30, random_state = 42)
```

Рисунок 16 – Деление датасета для модуля упругости при растяжении

Масштабируем данные.

```
# масштабируем обучающую выборку
X_train_scaled_1 = scaler.fit_transform(X_train_1)

# используем среднее арифметическое и СКО обучающей выборки для масштабирования тестовых данных
X_test_scaled_1 = scaler.transform(X_test_1)
```

Рисунок 17 – Масштабирование для модуля упругости при растяжении

Масштабировать целевые данные не будем, но преобразуем их в одномерный массив.

```
# Преобразуем целевые данные в одномерный массив
y_train_1 = y_train_1.values.ravel()
y_test_1 = y_test_1.values.ravel()
```

Рисунок 18 – Целевые данные 1

Сравним выбранные модели с параметрами по умолчанию (функция указана на рисунке 13).

```
table_1 = models_metrics(models_1, X_train_scaled_1, y_train_1)
table_1
```

	max_error	MAE	RMSE	R2
LinearRegression	-7.754544	-2.518390	-3.140033	-0.050320
DecisionTreeRegressor	-11.837542	-3.799079	-4.746385	-1.437139
RandomForestRegressor	-7.907961	-2.554298	-3.199981	-0.091714
Ridge	-7.753969	-2.518327	-3.139936	-0.050255
Lasso	-7.687178	-2.506801	-3.114610	-0.033390
ElasticNet	-7.687178	-2.506801	-3.114610	-0.033390
SVR	-8.007060	-2.523153	-3.160472	-0.066064
SGDRegressor	-7.748267	-2.517246	-3.137990	-0.048817
KNeighborsRegressor	-8.613586	-2.741288	-3.440714	-0.269092

Рисунок 19 – Модели до поиска гиперпараметров

Проведем поиск оптимальных гиперпараметров по каждой модели.

Введем переменную, куда будут сохраняться модели с оптимальными параметрами.

```
better_models_1 = {}
```

Рисунок 20

Модель LinearRegression() - поиск гиперпараметров

```
parameters_1 = {'fit_intercept': [True, False],
                'positive': [True, False]}

search, smodel = grid_search(LinearRegression(), parameters_1, X_train_scaled_1, y_train_1)
better_models_1[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 21 – Линейная регрессия 1

Модель DecisionTreeRegressor() - поиск гиперпараметров

```
parameters_1 = [{'random_state': [42],
                  'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
                  'splitter': ['best', 'random'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': ['sqrt', 'log2']},
                 {'random_state': [42],
                  'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
                  'splitter': ['best', 'random'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': range(1, 12)},
                 {'random_state': [42],
                  'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
                  'splitter': ['best', 'random'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': [None]}]

search, smodel = grid_search(DecisionTreeRegressor(), parameters_1, X_train_scaled_1, y_train_1)
better_models_1[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 22 - Регрессор дерева решений 1

Модель RandomForestRegressor() - поиск гиперпараметров

```
parameters_1 = [{'random_state': [42],
                  'n_estimators': [200],
                  'criterion': ['squared_error', 'absolute_error', 'friedman_mse', 'poisson'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': ['sqrt', 'log2'],
                  'bootstrap': [True, False]},
                 {'random_state': [42],
                  'n_estimators': [200],
                  'criterion': ['squared_error', 'absolute_error', 'friedman_mse', 'poisson'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': range(1, 12),
                  'bootstrap': [True, False]},
                 {'random_state': [42],
                  'n_estimators': [200],
                  'criterion': ['squared_error', 'absolute_error', 'friedman_mse', 'poisson'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': [None],
                  'bootstrap': [True, False]}]

search, smodel = grid_search(RandomForestRegressor(), parameters_1, X_train_scaled_1, y_train_1)
better_models_1[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 23 - Регрессор случайного леса 1

Модель Ridge() - поиск гиперпараметров

```
parameters_1 = [{'alpha': range(1, 111, 10),  
                 'fit_intercept': [True, False],  
                 'solver': ['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg']},  
                {'alpha': range(1, 101, 10),  
                 'fit_intercept': [True, False],  
                 'solver': ['lbfgs'],  
                 'positive': [True]},  
                {'alpha': range(1, 101, 10),  
                 'fit_intercept': [True, False],  
                 'solver': ['sag', 'saga'],  
                 'random_state': [42]}]  
  
search, smodel = grid_search(Ridge(), parameters_1, X_train_scaled_1, y_train_1)  
better_models_1[str(smodel)] = smodel  
search[search['rank']==1]
```

Рисунок 24 – Гребневая регрессия 1

Модель Lasso() – поиск гиперпараметров

```
parameters_1 = {'alpha': range(1, 111, 10),  
                'fit_intercept': [True, False],  
                'positive': [True, False]}  
  
search, smodel = grid_search(Lasso(), parameters_1, X_train_scaled_1, y_train_1)  
better_models_1[str(smodel)] = smodel  
search[search['rank']==1]
```

Рисунок 25 – Регрессия Лассо 1

Модель ElasticNet() – поиск гиперпараметров

```
parameters_1 = {'alpha': range(1, 111, 10),  
                'l1_ratio': [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],  
                'fit_intercept': [True, False],  
                'positive': [True, False],  
                'tol': [4e+03]}  
  
search, smodel = grid_search(ElasticNet(), parameters_1, X_train_scaled_1, y_train_1)  
better_models_1[str(smodel)] = smodel  
search[search['rank']==1]
```

Рисунок 26 – Эластичная сеть 1

Модель SVR() – поиск гиперпараметров

```
parameters_1 = [{'kernel': ['linear'],
                  'C': range(1, 111, 10)}, # по умолчанию 1.0
                 {'kernel': ['poly'],
                  'C': range(1, 111, 10),
                  'degree': range(2, 5),
                  'gamma': ['auto', 'scale'],
                  'coef0': [0, 1]},
                 {'kernel': ['rbf'],
                  'C': range(1, 111, 10),
                  'gamma': ['auto', 'scale']}],
                 {'kernel': ['sigmoid'],
                  'C': range(1, 111, 10),
                  'gamma': ['auto', 'scale'],
                  'coef0': [0, 1]}]

search, smodel = grid_search(SVR(), parameters_1, X_train_scaled_1, y_train_1)
better_models_1[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 27 – Регрессия опорных векторов 1

Модель SGDRegressor() – поиск гиперпараметров

```
parameters_1 = [{'loss': ['squared_error', 'huber', 'epsilon_insensitive', 'squared_epsilon_insensitive'],
                  'penalty': ['l2', 'l1', None],
                  'alpha': range(1, 111, 10),
                  'fit_intercept': [True, False],
                  'random_state': [42],
                  'learning_rate': ['constant', 'optimal', 'invscaling', 'adaptive'],
                  'max_iter': [10000000]},
                 {'loss': ['squared_error', 'huber', 'epsilon_insensitive', 'squared_epsilon_insensitive'],
                  'penalty': ['elasticnet'],
                  'alpha': range(1, 111, 10),
                  'l1_ratio': [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0], # 0 <= l1_ratio <= 1
                  'fit_intercept': [True, False],
                  'random_state': [42],
                  'learning_rate': ['constant', 'optimal', 'invscaling', 'adaptive'],
                  'max_iter': [10000000]}]

search, smodel = grid_search(SGDRegressor(), parameters_1, X_train_scaled_1, y_train_1)
better_models_1[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 28 – Стохастический градиентный спуск 1

Модель KNeighborsRegressor () – поиск гиперпараметров

```
parameters_1 = {'n_neighbors': range(5, 31, 2),
                 'weights': ['uniform', 'distance'],
                 'algorithm': ['ball_tree', 'kd_tree', 'brute', 'auto'],
                 'p': [1, 2, 3]}

search, smodel = grid_search(KNeighborsRegressor(), parameters_1, X_train_scaled_1, y_train_1)
better_models_1[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 29 – Регрессия k-ближайших соседей 1

Сравним модели для параметра "Модуль упругости при растяжении, ГПа" после подбора гиперпараметров (функция указана на рисунке 13).

```
table_1_improved = models_metrics(better_models_1, X_train_scaled_1, y_train_1)
table_1_improved
```

	max_error	MAE	RMSE	R2
LinearRegression(positive=True)	-7.709031	-2.505067	-3.127999	-0.041764
DecisionTreeRegressor(max_depth=1, max_features=2, random_state=42)	-7.837971	-2.498890	-3.117294	-0.035270
RandomForestRegressor(criterion='absolute_error', max_depth=3, max_features=1, n_estimators=200, random_state=42)	-7.723437	-2.502208	-3.117585	-0.036191
Ridge(alpha=91, positive=True, solver='lbfgs')	-7.693051	-2.503336	-3.123893	-0.039065
Lasso(alpha=1, positive=True)	-7.687178	-2.506801	-3.114610	-0.033390
ElasticNet(alpha=1, l1_ratio=0.0, positive=True, tol=4000.0)	-7.662191	-2.502964	-3.116174	-0.034063
SVR(C=1, coef0=0, gamma='auto', kernel='sigmoid')	-8.003318	-2.516191	-3.176419	-0.075434
SGDRegressor(alpha=1, l1_ratio=0.2, learning_rate='constant', max_iter=10000000, penalty='elasticnet', random_state=42)	-7.684682	-2.501460	-3.104073	-0.026146
KNeighborsRegressor(algorithm='ball_tree', n_neighbors=23, p=1, weights='distance')	-7.892295	-2.543740	-3.168132	-0.071450

Рисунок 30 – Модели после поиска гиперпараметров

В результате подбора гиперпараметров по сетке модели были улучшены, хотя и незначительно.

Метрики моделей имеют отрицательные значения. Следуя принципу "чем выше, тем лучше", выберем модель, метрики которой близки к нулю. В нашем случае это SGDRegressor.

Обучим выбранную модель и спрогнозируем целевой параметр.

```
best_model_1 = SGDRegressor(alpha=1, l1_ratio=0.2, learning_rate='constant', max_iter=10000000, penalty='elasticnet', random_state=42)
best_model_1.fit(X_train_scaled_1, y_train_1)
y1_pred = best_model_1.predict(X_test_scaled_1)
```

Рисунок 31 – Прогноз 1

Качество прогнозирования выбранной модели (функция указана на рисунке 15)

```
best_model_1_metrics = best_model_metrics('SGDRegressor', y_test_1, y1_pred)
best_model_1_metrics
```

	max_error	MAE	RMSE	R2
SGDRegressor	-9.40674	-2.482018	-3.078749	-0.020254

Рисунок 32 – Модель для модуля упругости при растяжении

2.2.2 Модель машинного обучения для параметра «Прочность при растяжении, МПа»

Разделим данные на обучающую и тестовую выборки.

```
# Разделим данные на train и test
X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(X, y2, test_size = 0.30, random_state = 42)
```

Рисунок 33 – Деление датасета для Прочности при растяжении

Масштабируем данные.

```
# масштабируем обучающую выборку
X_train_scaled_2 = scaler.fit_transform(X_train_2)

# используем среднее арифметическое и СКО обучающей выборки для масштабирования тестовых данных
X_test_scaled_2 = scaler.transform(X_test_2)
```

Рисунок 34 – Масштабирование для Прочности при растяжении

Масштабировать целевые данные не будем, но преобразуем их в одномерный массив.

```
# Преобразуем целевые данные в одномерный массив
y_train_2 = y_train_2.values.ravel()
y_test_2 = y_test_2.values.ravel()
```

Рисунок 35 – Целевые данные 2

Сравним выбранные модели с параметрами по умолчанию (функция указана на рисунке 13).


```
table_2 = models_metrics(models_2, X_train_scaled_2, y_train_2)
table_2
```

	max_error	MAE	RMSE	R2
LinearRegression	-1278.790633	-389.140981	-489.393761	-0.023017
DecisionTreeRegressor	-1853.212070	-565.241047	-705.793418	-1.187462
RandomForestRegressor	-1295.309292	-397.583551	-496.374281	-0.056399
Ridge	-1278.701017	-389.121962	-489.376075	-0.022937
Lasso	-1276.501200	-388.572626	-488.798449	-0.020419
ElasticNet	-1260.882923	-385.862788	-486.452377	-0.009559
SVR	-1256.587914	-386.793138	-487.032580	-0.010721
KNeighborsRegressor	-1404.589902	-422.401615	-529.352566	-0.201183

Рисунок 36 – Модели до поиска гиперпараметров

Проведем поиск оптимальных гиперпараметров по каждой модели.

Введем переменную, куда будут сохраняться модели с оптимальными параметрами.

```
better_models_2 = {}
```

Рисунок 37

Модель LinearRegression() - поиск гиперпараметров

```
parameters_2 = {'fit_intercept': [True, False],
                'positive': [True, False]}

search, smodel = grid_search(LinearRegression(), parameters_2, X_train_scaled_2, y_train_2)
better_models_2[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 38 – Линейная регрессия 2

Модель DecisionTreeRegressor() - поиск гиперпараметров

```
parameters_2 = [{'random_state': [42],
                  'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
                  'splitter': ['best', 'random'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': ['sqrt', 'log2']},
                {'random_state': [42],
                  'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
                  'splitter': ['best', 'random'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': range(1, 12)},
                {'random_state': [42],
                  'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
                  'splitter': ['best', 'random'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': [None]}]

search, smodel = grid_search(DecisionTreeRegressor(), parameters_2, X_train_scaled_2, y_train_2)
better_models_2[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 39 - Регрессор дерева решений 2

Модель RandomForestRegressor() - поиск гиперпараметров

```
parameters_2 = [{'random_state': [42],
                  'n_estimators': [200],
                  'criterion': ['squared_error', 'absolute_error', 'friedman_mse', 'poisson'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': ['sqrt', 'log2'],
                  'bootstrap': [True, False]},
                {'random_state': [42],
                  'n_estimators': [200],
                  'criterion': ['squared_error', 'absolute_error', 'friedman_mse', 'poisson'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': range(1, 12),
                  'bootstrap': [True, False]},
                {'random_state': [42],
                  'n_estimators': [200],
                  'criterion': ['squared_error', 'absolute_error', 'friedman_mse', 'poisson'],
                  'max_depth': [1, 2, 3, None],
                  'max_features': [None],
                  'bootstrap': [True, False]}]

search, smodel = grid_search(RandomForestRegressor(), parameters_2, X_train_scaled_2, y_train_2)
better_models_2[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 40 - Регрессор случайного леса 2

Модель Ridge() - поиск гиперпараметров

```
parameters_2 = [{'alpha': range(1, 111, 10),
                  'fit_intercept': [True, False],
                  'solver': ['auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg']},
                 {'alpha': range(1, 101, 10),
                  'fit_intercept': [True, False],
                  'solver': ['lbfgs'],
                  'positive': [True]},
                 {'alpha': range(1, 101, 10),
                  'fit_intercept': [True, False],
                  'solver': ['sag', 'saga'],
                  'random_state': [42]}]

search, smodel = grid_search(Ridge(), parameters_2, X_train_scaled_2, y_train_2)
better_models_2[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 41 – Гребневая регрессия 2

Модель Lasso() – поиск гиперпараметров

```
parameters_2 = {'alpha': range(1, 111, 10),
                 'fit_intercept': [True, False],
                 'positive': [True, False]}

search, smodel = grid_search(Lasso(), parameters_2, X_train_scaled_2, y_train_2)
better_models_2[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 42 – Регрессия Лассо 2

Модель ElasticNet() – поиск гиперпараметров

```
parameters_2 = {'alpha': range(1, 111, 10),
                 'l1_ratio': [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0],
                 'fit_intercept': [True, False],
                 'positive': [True, False],
                 'tol': [4e+03]}

search, smodel = grid_search(ElasticNet(), parameters_2, X_train_scaled_2, y_train_2)
better_models_2[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 43 – Эластичная сеть 2

Модель SVR() – поиск гиперпараметров

```
parameters_2 = [{'kernel': ['linear'],
                  'C': range(1, 111, 10)}, # по умолчанию 1.0
                 {'kernel': ['poly'],
                  'C': range(1, 111, 10),
                  'degree': range(2, 5),
                  'gamma': ['auto', 'scale'],
                  'coef0': [0, 1]},
                 {'kernel': ['rbf'],
                  'C': range(1, 111, 10),
                  'gamma': ['auto', 'scale']}],
                 {'kernel': ['sigmoid'],
                  'C': range(1, 111, 10),
                  'gamma': ['auto', 'scale'],
                  'coef0': [0, 1]}}

search, smodel = grid_search(SVR(), parameters_2, X_train_scaled_2, y_train_2)
better_models_2[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 44 – Регрессия опорных векторов 2

Модель KNeighborsRegressor () – поиск гиперпараметров

```
parameters_2 = {'n_neighbors': range(5, 31, 2),
                 'weights': ['uniform', 'distance'],
                 'algorithm': ['ball_tree', 'kd_tree', 'brute', 'auto'],
                 'p': [1, 2, 3]}

search, smodel = grid_search(KNeighborsRegressor(), parameters_2, X_train_scaled_2, y_train_2)
better_models_2[str(smodel)] = smodel
search[search['rank']==1]
```

Рисунок 45 – Регрессия k-ближайших соседей 2

Сравним модели для параметра "Прочность при растяжении, МПа" после подбора гиперпараметров (функция указана на рисунке 13).

```
table_2_improved = models_metrics(better_models_2, X_train_scaled_2, y_train_2)
table_2_improved
```

	max_error	MAE	RMSE	R2
LinearRegression(positive=True)	-1258.054267	-385.924801	-488.176644	-0.014997
DecisionTreeRegressor(criterion='absolute_error', max_depth=3, max_features=7, random_state=42, splitter='random')	-1267.015475	-381.101592	-482.397665	0.008114
RandomForestRegressor(criterion='poisson', max_depth=2, max_features=2, n_estimators=200, random_state=42)	-1247.264367	-383.937358	-484.824258	-0.002294
Ridge(alpha=91, positive=True, solver='lbfgs')	-1255.130626	-385.688933	-487.804637	-0.013487
Lasso(alpha=21)	-1250.154999	-383.695991	-484.101871	0.000730
ElasticNet(alpha=21, l1_ratio=1.0, tol=4000.0)	-1249.856166	-383.695406	-484.127785	0.000621
SVR(C=11, coef0=0, degree=4, kernel='poly')	-1224.265261	-383.879245	-485.013673	-0.001935
KNeighborsRegressor(algorithm='ball_tree', n_neighbors=27, p=1)	-1256.245922	-386.941168	-486.491225	-0.009783

Рисунок 46 – Модели после поиска гиперпараметров

В результате подбора гиперпараметров по сетке модели были улучшены. Метрики моделей имеют отрицательные значения, за исключением r2 у трех моделей. Следуя принципу "чем выше, тем лучше", выберем модель, MAE и RMSE которой близки к нулю, а r2 выше остальных. В нашем случае это DecisionTreeRegressor.

Обучим выбранную модель и спрогнозируем целевой параметр.

```
best_model_2 = DecisionTreeRegressor(criterion='absolute_error', max_depth=3, max_features=7, random_state=42, splitter='random')
best_model_2.fit(X_train_scaled_2, y_train_2)
y2_pred = best_model_2.predict(X_test_scaled_2)
```

Рисунок 47 – Прогноз 2

Качество прогнозирования выбранной модели (функция указана на рисунке 15)

```
best_model_2_metrics = best_model_metrics('DecisionTreeRegressor', y_test_2, y2_pred)
best_model_2_metrics
```

	max_error	MAE	RMSE	R2
DecisionTreeRegressor	-1468.095589	-380.172727	-477.100938	-0.010834

Рисунок 48 – Модель для прочности при растяжении

Сохраним модели.

```
pickle.dump(best_model_1, open('best_model_1.pkl', 'wb'))  
pickle.dump(best_model_2, open('best_model_2.pkl', 'wb'))
```

Рисунок 49 – Сохранение моделей

2.3 Разработка нейронной сети

Для построения нейронной сети будем использовать библиотеку Keras.

Используем датасет, предобработанный выше в п.2.1. Поскольку нейронная сеть будет прогнозировать новый признак, нам необходимо будет провести разбиение данных на X и y заново.

Проведем разбиение датасета на входные (X) и целевые (y) данные.

```
# Целевой параметр - "Соотношение матрица-наполнитель"
y = df_clean.iloc[:, df_clean.columns == 'Соотношение матрица-наполнитель']

# Входные параметры
X = df_clean.drop(columns=['Соотношение матрица-наполнитель'])

# Посмотрим форму полученных данных.
print(y.shape)
print(X.shape)
```

(999, 1)
(999, 12)

Рисунок 50 – Входные и целевые данные для Н.С.

Разделим датасет на обучающую и тестовую выборки.

```
# Разделим данные на обучающую и тестовую выборку
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 42)
```

Рисунок 51 – Выборки train и test для Н.С.

Масштабируем данные с помощью метода StandardScaler(). Метод указан на рисунке 10.

```
# масштабируем обучающую выборку
X_train_scaled = scaler.fit_transform(X_train)

# используем среднее арифметическое и СКО обучающей выборки для масштабирования тестовых данных
X_test_scaled = scaler.transform(X_test)
```

Рисунок 52 – Масштабирование данных для Н.С.

Применять масштабирование к целевым данным не будем, но преобразуем их в массив.

```
# Преобразуем целевые данные в numpy массив
y_train_z = y_train.values
y_test_z = y_test.values
```

Рисунок 53 – Целевые данные Н.С.

Определим количество признаков на вход нейронной сети.

```
X.shape[1]
```

12

Рисунок 54 – Вход Н.С.

Определим архитектуру нейронной сети.

```
model = keras.Sequential()

model.add(keras.layers.Dense(128, input_dim = 12, activation = 'relu'))
model.add(keras.layers.Dropout(0.2))

model.add(keras.layers.Dense(64, input_dim = 12, activation = 'relu'))
model.add(keras.layers.Dropout(0.2))

model.add(keras.layers.Dense(32, activation = 'relu'))
model.add(keras.layers.Dropout(0.2))

model.add(keras.layers.Dense(16, input_dim = 12, activation = 'relu'))
model.add(keras.layers.Dropout(0.2))

model.add(keras.layers.Dense(1, activation = 'linear'))
```

Рисунок 55 – Архитектура Н.С.

Посмотрим информацию о нашей нейронной сети.


```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	1664
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2080
dropout_2 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 16)	528
dropout_3 (Dropout)	(None, 16)	0
dense_4 (Dense)	(None, 1)	17
Total params: 12,545		
Trainable params: 12,545		
Non-trainable params: 0		

Рисунок 56 – Информация о Н.С.

Проведем компиляцию и обучение модели.

Применим метод EarlyStopping, который останавливает процесс обучения, когда контролируемая метрика перестает улучшаться.

```
# Компиляция
```

```
model.compile(loss = keras.losses.MeanSquaredError(), optimizer = keras.optimizers.Adam(),  
              metrics = [keras.metrics.MeanAbsoluteError()])
```

```
# метод EarlyStopping для остановки обучения, как только производительность модели перестанет улучшаться.
```

```
earlystopping = keras.callbacks.EarlyStopping(monitor='val_loss', min_delta=0, patience=10,  
                                              verbose=0, mode='auto', restore_best_weights=True)
```

```
# Обучение
```

```
history = model.fit(X_train_scaled, y_train_z, batch_size = 150, epochs = 1000, validation_split=0.2,  
                   callbacks=[earlystopping], verbose = 1)
```

Рисунок 57 – Компиляция и обучение модели Н.С.

Посмотрим на график обучения (рисунок 58).

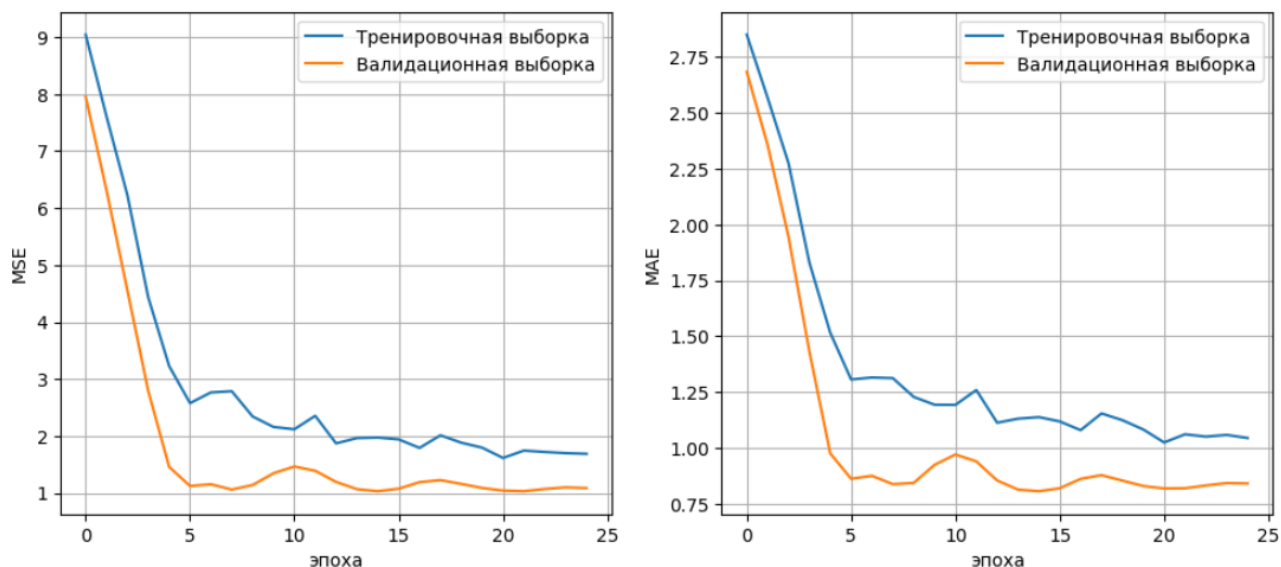


Рисунок 58 – График обучения Н.С.

На рисунке 58 видно, что значения метрик MSE и MAE уменьшаются по мере обучения нейронной сети.

Проверим метрики MSE и MAE, применив модель на тестовой выборке (рисунок 59). В итоге, MSE составляет 1.08, а MAE составляет 0.83.

```
# MSE и MAE на тестовой выборке
```

```
print(f'MSE и MAE на тестовой выборке: {model.evaluate(X_test_scaled, y_test_z,  
                                                    batch_size = 50, callbacks=[earlystopping])}')'
```

```
6/6 [=====] - 0s 8ms/step - loss: 1.0781 - mean_absolute_error: 0.8279  
MSE и MAE на тестовой выборке: [1.0780550241470337, 0.827873706817627]
```

Рисунок 59 - MSE и MAE на тестовой выборке (Н.С.)

Далее спрогнозируем целевой признак на тестовой выборке.

```
# Прогноз
```

```
y_pred = model.predict(X_test_scaled)
```

```
10/10 [=====] - 0s 5ms/step
```

Рисунок 60 – Прогноз Н.С.

Сравним прогноз с реальными данными.

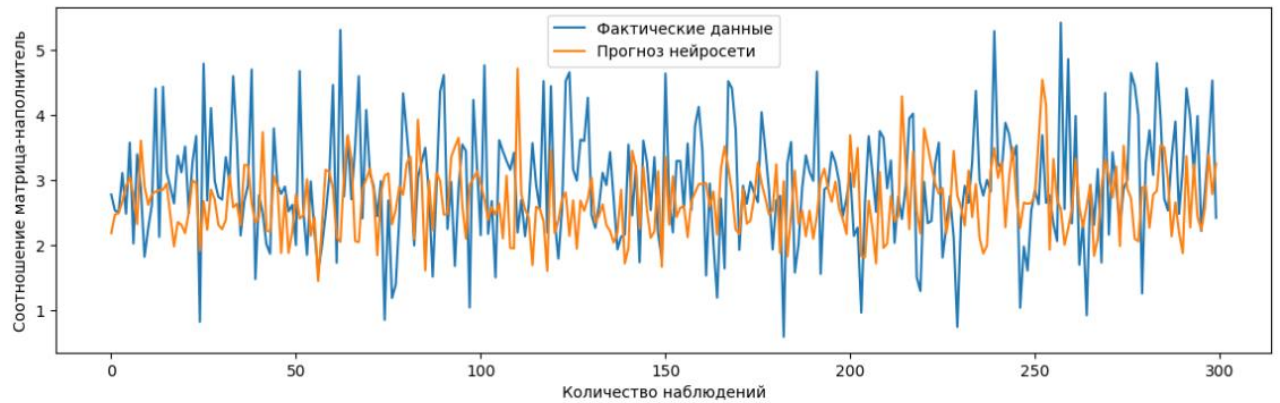


Рисунок 61 – Сравнение прогноза Н.С. с реальными данными

Сохраним модель.

```
model.save("models/ANN")
```

INFO:tensorflow:Assets written to: models/ANN/assets

Рисунок 62 – Сохранение Н.С.

2.4 Разработка приложения

Разработаем приложение для применения наших моделей на практике. Это web-приложение на основе библиотеки Flask. Исходный код представлен в Приложении 1.

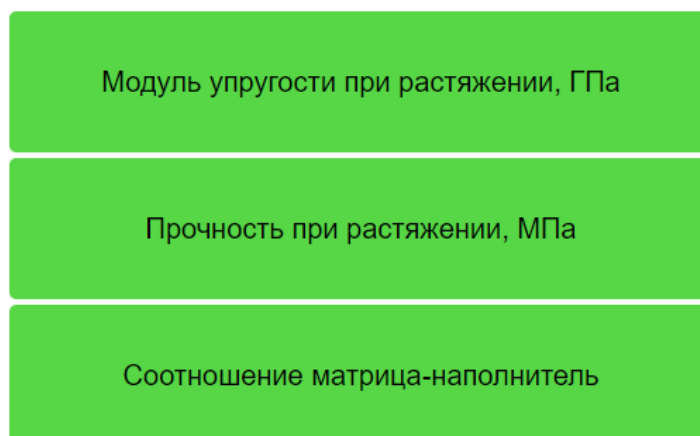
Приложение запускается локально в браузере из среды Python, для этого необходимо активировать файл «Application_VasilevAI_VKR_MGTU.py».

Web-приложение выполнено по многостраничной схеме.

На стартовой странице (рисунок 63) выбирается признак для расчета.

Прогнозирование конечных свойств новых материалов (композиционных материалов)

Выберите параметр для расчёта:



Модуль упругости при растяжении, ГПа

Прочность при растяжении, МПа

Соотношение матрица-наполнитель

Рисунок 63 – Стартовая страница web-приложения (index.html)

После выбора признака пользователь перенаправляется на соответствующую страницу (рисунки 64, 65 и 66).

Назад

Модуль упругости при растяжении, ГПа

Введите значение

Введите Соотношение матрица-наполнитель	<input type="text"/>
Введите Плотность, кг/м ³	<input type="text"/>
Введите Модуль упругости, ГПа	<input type="text"/>
Введите Количество отвердителя, м.%	<input type="text"/>
Введите Содержание эпоксидных групп, % ₂	<input type="text"/>
Введите Температура вспышки, С ₂	<input type="text"/>
Введите Поверхностная плотность, г/м ²	<input type="text"/>
Введите Потребление смолы, г/м ²	<input type="text"/>
Введите Угол нашивки, град	<input type="text"/>
Введите Шаг нашивки	<input type="text"/>
Введите Плотность нашивки	<input type="text"/>

Рисунок 64 – Страница для расчета Модуля упругости при растяжении
(main_1.html)

Назад

Прочность при растяжении, МПа

Введите значение

Введите Соотношение матрица-наполнитель	<input type="text"/>
Введите Плотность, кг/м ³	<input type="text"/>
Введите Модуль упругости, ГПа	<input type="text"/>
Введите Количество отвердителя, м.%	<input type="text"/>
Введите Содержание эпоксидных групп, % ₂	<input type="text"/>
Введите Температура вспышки, С ₂	<input type="text"/>
Введите Поверхностная плотность, г/м ²	<input type="text"/>
Введите Потребление смолы, г/м ²	<input type="text"/>
Введите Угол нашивки, град	<input type="text"/>
Введите Шаг нашивки	<input type="text"/>
Введите Плотность нашивки	<input type="text"/>

Рисунок 65 – Страница для расчета Прочности при растяжении
(main_2.html)

Назад

Соотношение матрица-наполнитель

Введите значение	
Введите Плотность, кг/м3	<input type="text"/>
Введите Модуль упругости, ГПа	<input type="text"/>
Введите Количество отвердителя, м.%	<input type="text"/>
Введите Содержание эпоксидных групп, %_2	<input type="text"/>
Введите Температура вспышки, С_2	<input type="text"/>
Введите Поверхностная плотность, г/м2	<input type="text"/>
Введите Модуль упругости при растяжении, ГПа	<input type="text"/>
Введите Прочность при растяжении, МПа	<input type="text"/>
Введите Потребление смолы, г/м2	<input type="text"/>
Введите Угол нашивки, град	<input type="text"/>
Введите Шаг нашивки	<input type="text"/>
Введите Плотность нашивки	<input type="text"/>
<input type="button" value="Рассчитать"/> <input type="button" value="Сбросить"/>	

Рисунок 66 – Страница для расчета Соотношения матрица-наполнитель
(main_3.html)

Для расчета необходимо ввести данные в соответствующие поля и нажать на кнопку «Рассчитать» в нижней части формы ввода данных.

Кнопка «Сбросить» в нижней части формы ввода данных очищает поля.

Результат расчета отображается ниже формы ввода данных.

Для возврата на стартовую страницу нужно нажать на кнопку «Назад» в верхнем левом углу.

2.5 Создание репозитория

Файлы исследования и приложение размещены в репозитории: <https://github.com/Alvas01/BMSTU> (рисунок 67).

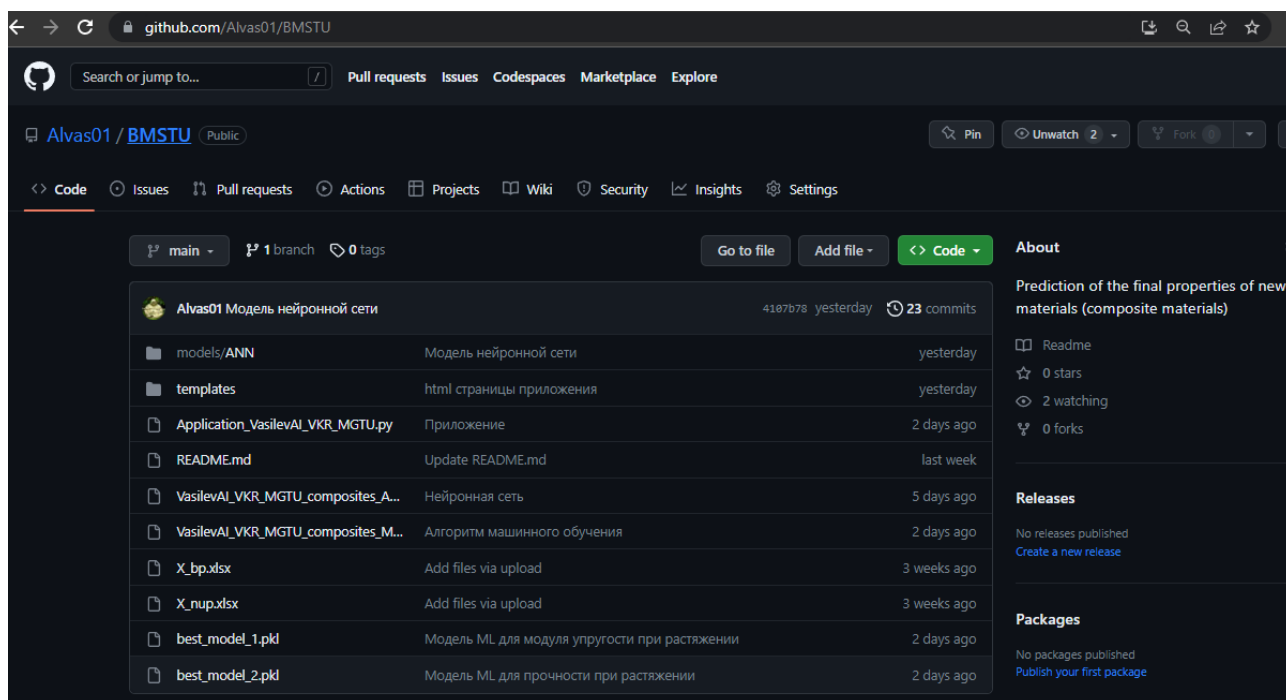


Рисунок 67 – Страница репозитория на GitHub

Заключение

В ходе данной работы мы пытались спрогнозировать значения признаков «Модуль упругости при растяжении, ГПа», «Прочность при растяжении, МПа» и «Соотношение матрица-наполнитель».

Для выполнения поставленной задачи мы выбрали методы машинного обучения путем подбора гиперпараметров и разработали искусственную нейронную сеть.

Примененные нами методы машинного обучения и нейронная сеть показали невысокую эффективность.

Разведочный анализ данных показал низкую (околонулевую) корреляцию признаков в датасете. Думаю, что это стало причиной низкой эффективности моделей, которые строят свою работу на выявлении взаимосвязей в данных.

Предполагаю, что в предоставленных данных имеются скрытые недостатки. Для повышения эффективности моделей необходима дополнительная информация и, возможно, более тонкая и трудоемкая настройка моделей.

В рамках данного исследования были проведены следующие работы:

- 1) разведочный анализ и предобработка данных.
- 2) обучение моделей для прогноза признаков «Модуль упругости при растяжении, ГПа» и «Прочность при растяжении, МПа». Модели были выбраны путем подбора гиперпараметров по сетке.
- 3) создана нейронная сеть, рекомендуемая соотношение матрица-наполнитель.
- 4) разработано приложение на основе выбранных моделей машинного обучения и нейронной сети.
- 5) создан репозиторий Github.

Библиографический список

- 1 Андерсон, Карл Аналитическая культура. От сбора данных до бизнес-результатов / Карл Андерсон ; пер. с англ. Юлии Константиновой ; [науч. ред. Руслан Салахияев]. — М. : Манн, Иванов и Фербер, 2017. — 336 с.
- 2 Билл Любанович. Простой Python. Современный стиль программирования. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O'Reilly»).
- 3 Бизли Д. Python. Подробный справочник: учебное пособие. — Пер. с англ. — СПб.: Символ-Плюс, 2010. — 864 с., ил.
- 4 Гафаров, Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учеб. пособие /Ф.М. Гафаров, А.Ф. Галимянов. — Казань: Издательство Казанского университета, 2018. — 121 с.
- 5 Грас Д. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.
- 6 Джулли, Пал: Библиотека Keras - инструмент глубокого обучения / пер. с англ. А. А. Слинкин.- ДМК Пресс, 2017. — 249 с.
- 7 Жерон, Орельен. Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем. Пер. с англ. - СПб.: ООО "Альфа-книга": 2018. - 688 с.: ил.
- 8 Плас Дж. Вандер, Python для сложных задач: наука о данных и машинное обучение. Санкт-Петербург: Питер, 2018, 576 с.
- 9 Рассел С., Норвиг П. Искусственный интеллект: современный подход, 2-е изд.. : Пер. с англ. - М. : Издательский дом “Вильямс”, 2007. - 1408 с.
- 10 Роббинс, Дженнифер. HTML5: карманный справочник, 5-е издание.: Пер. с англ. - М.: ООО «И.Д. Вильямс»: 2015. - 192 с.: ил.

- 11 С. Николенко, А. Кадури, Е. Архангельская Глубокое обучение. Погружение в мир нейронных сетей. - СПб.: Питер. - 2020. - 480 с.
- 12 Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru. 2020. - 412 с.: ил.
- 13 Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.
- 14 Скиена, Стивен С. С42 Наука о данных: учебный курс.: Пер. с англ. - СПб.: ООО "Диалектика", 2020. - 544 с. : ил.
- 15 Траск Эндрю. Грожаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.: ил.
- 16 Документация по библиотеке keras: – Режим доступа: <https://keras.io/api/> (дата обращения: 04.2023).
- 17 Документация по библиотеке matplotlib: – Режим доступа: <https://matplotlib.org/stable/users/index.html> (дата обращения: 04.2023)
- 18 Документация по библиотеке numpy: – Режим доступа: <https://numpy.org/doc/1.22/user/index.html#user> (дата обращения: 04.2023)
- 19 Документация по библиотеке pandas: – Режим доступа: https://pandas.pydata.org/docs/user_guide/index.html#user-guide (дата обращения: 04.2023).
- 20 Документация по библиотеке scikit-learn: – Режим доступа: https://scikit-learn.org/stable/user_guide.html (дата обращения: 04.2023).
- 21 Документация по библиотеке seaborn: – Режим доступа: <https://seaborn.pydata.org/tutorial.html> (дата обращения: 04.2023).
- 22 Документация по библиотеке Tensorflow: – Режим доступа: <https://www.tensorflow.org/overview> (дата обращения: 04.2023).
- 23 Документация по языку программирования python: – Режим доступа: <https://docs.python.org/3.8/index.html> (дата обращения: 04.2023).

24 Руководство по быстрому старту в flask: – Режим доступа: <https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html> (дата обращения: 04.2023).

25 Мега-Учебник Flask, Часть 1: «Привет, Мир!»: – Режим доступа: <https://habr.com/ru/post/193242/> (дата обращения: 04.2023).

26 Веб фреймворк Flask в Python: – Режим доступа: <https://docs-python.ru/packages/web-frejmwork-flask-python/> (дата обращения: 09.12.2022).

Приложение 1

Листинг web-приложения:

```
import flask
import tensorflow as tf
import pickle

app = flask.Flask(__name__, template_folder='templates')

@app.route('/', methods = ['POST', 'GET'])
def index():
    return flask.render_template('index.html')

@app.route('/main_1', methods = ['POST', 'GET'])
def main_1():
    result_1 = "
    if flask.request.method == 'GET':
        return flask.render_template('main_1.html')
    if flask.request.method == 'POST':
        param_list = ('Соотношение матрица-наполнитель', 'Плотность, кг/м3',
'модуль упругости, ГПа', 'Количество отвердителя, м.%',
        'Содержание эпоксидных групп,%_2', 'Температура вспышки, С_2',
'Поверхностная плотность, г/м2',
        'Потребление смолы, г/м2', 'Угол нашивки, град', 'Шаг нашивки',
'Плотность нашивки')
        params = []
        for i in param_list:
            param = flask.request.form.get(i)
            params.append(param)
        params = [float(i.replace(',', '.')) for i in params]
```

```

with open('best_model_1.pkl', 'rb') as f:
    model = pickle.load(f)

pred = model.predict([params])

result_1 = f'Модуль упругости при растяжении, ГПа: {pred}'
return flask.render_template('main_1.html', result=result_1)

@app.route('/main_2', methods = ['POST', 'GET'])
def main_2():
    result_2 = "
    if flask.request.method == 'GET':
        return flask.render_template('main_2.html')
    if flask.request.method == 'POST':
        param_list = ('Соотношение матрица-наполнитель', 'Плотность, кг/м3',
        'модуль упругости, ГПа', 'Количество отвердителя, м.%',
        'Содержание эпоксидных групп,%_2', 'Температура вспышки, С_2',
        'Поверхностная плотность, г/м2',
        'Потребление смолы, г/м2', 'Угол нашивки, град', 'Шаг нашивки',
        'Плотность нашивки')
        params = []
        for i in param_list:
            param = flask.request.form.get(i)
            params.append(param)
        params = [float(i.replace(',', '.')) for i in params]

        with open('best_model_2.pkl', 'rb') as f:
            model = pickle.load(f)

```

```

pred = model.predict([params])

result_2 = f'Прочность при растяжении, МПа: {pred}'
return flask.render_template('main_2.html', result=result_2)

@app.route('/main_3', methods = ['POST', 'GET'])
def main_3():
    result_3 = "
    if flask.request.method == 'GET':
        return flask.render_template('main_3.html')
    if flask.request.method == 'POST':
        param_list = ('Плотность, кг/м3', 'модуль упругости, ГПа', 'Количество
отвердителя, м.%',
            'Содержание эпоксидных групп,%_2', 'Температура вспышки, С_2',
            'Поверхностная плотность, г/м2',
            'Модуль упругости при растяжении, ГПа', 'Прочность при
растяжении, МПа', 'Потребление смолы, г/м2',
            'Угол нашивки, град', 'Шаг нашивки', 'Плотность нашивки')
        params = []
        for i in param_list:
            param = flask.request.form.get(i)
            params.append(param)
        params = [float(i.replace(',', '.')) for i in params]

        model = tf.keras.models.load_model('models/ANN')
        pred = model.predict([params])

        result_3 = f'Соотношение матрица-наполнитель: {pred}'
        return flask.render_template('main_3.html', result=result_3)

```

```
if __name__ == '__main__':  
    app.run()
```

Листинг web-страницы index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Главная</title>
    <style>
      p {
        font-size: 200%;
        align: center;
      }
      button {
        appearance: none;
        border: 0;
        border-radius: 5px;
        background: #57d746;
        color: #000;
        padding: 20px 10px;
        font-size: 20px;
        height: 100px;
        width: 500px;
      }
      button:hover {
        background: #66ff00;
      }
    </style>
  </head>
  <body>
```


<div>

<p align="center">

Прогнозирование конечных свойств новых материалов (композиционных материалов)

</br>

</p>

<p align="center">

Выберите параметр для расчёта:

</br>

</p>

</div>

<table align="center">

<tr align="right" width = 500px>

<td>

<button>Модуль упругости при растяжении, ГПа</button>

</td>

</tr>

<tr align="right" width = 500px>

<td>

<button>Прочность при растяжении, МПа</button>

</td>

</tr>

<tr align="right" width = 500px>

<td>

```
<a href="{ {url_for('main_3') }}">
  <button>Соотношение матрица-наполнитель</button>
</a>
</td>
</tr>
</table>
</body>
</html>
```

Листинг web-страницы main_1.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Модуль упругости при растяжении, ГПа</title>
    <style>
      div_1 {
        font-size: 200%;
      }
      form {
        margin: auto;
        width: 35%;
        background-color:#afafaf;
        width:600px;
        margin:1;
        align: center;
      }
      .button, button {
        appearance: none;
        border: 1;
        border-radius: 5px;
        background: #fff;
        color: #000;
        padding: 8px 16px;
        font-size: 16px;
      }
      .button:hover {
```

```

        background: #57d746;
    }
    button:hover {
        background: #57d746;
    }
    .result {
        margin: auto;
        width: 35%;
        border: 1px solid #ccc;
        background-color: #dbdbdb;
        width: 600px;
        margin: 1;
        align: center;
    }
</style>
</head>
<body>
    <a href="{ {url_for('index')}} ">
        <button>Назад</button>
    </a>
    <div_1>
        <p align="center">
            Модуль упругости при растяжении, ГПа
        </br>
        </p>
    </div_1>
    <form action="{ {url_for('main_1')}}" method="POST">
        <fieldset>
            <legend>Введите значение</legend>
            <table align="center">

```

```

<tr>
  <td align="right">
    <label for="Соотношение матрица-наполнитель">Введите
Соотношение матрица-наполнитель</label>
    <input type="text" name="Соотношение матрица-наполнитель"
required>
  </td>
</tr>
<tr>
  <td align="right">
    <label for="Плотность, кг/м3">Введите Плотность, кг/м3</label>
    <input type="text" name="Плотность, кг/м3" required>
  </td>
</tr>
<tr>
  <td align="right">
    <label for="модуль упругости, ГПа">Введите Модуль упругости,
ГПа</label>
    <input type="text" name="модуль упругости, ГПа" required>
  </td>
</tr>
<tr>
  <td align="right">
    <label for="Количество отвердителя, м.%">Введите Количество
отвердителя, м.%</label>
    <input type="text" name="Количество отвердителя, м.%" required>
  </td>
</tr>
<tr>
  <td align="right">

```

```

        <label for="Содержание эпоксидных групп,%_2">Введите
Содержание эпоксидных групп,%_2</label>
        <input type="text" name="Содержание эпоксидных групп,%_2"
required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Температура вспышки, C_2">Введите Температура
вспышки, C_2</label>
        <input type="text" name="Температура вспышки, C_2" required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Поверхностная плотность, г/м2">Введите
Поверхностная плотность, г/м2</label>
        <input type="text" name="Поверхностная плотность, г/м2" required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Потребление смолы, г/м2">Введите Потребление
смолы, г/м2</label>
        <input type="text" name="Потребление смолы, г/м2" required>
    </td>
</tr>
<tr>
    <td align="right">

```

```

        <label for="Угол нашивки, град">Введите Угол нашивки,
град</label>
        <input type="text" name="Угол нашивки, град" min='0' max='90'
step='1' required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Шаг нашивки">Введите Шаг нашивки</label>
        <input type="text" name="Шаг нашивки" required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Плотность нашивки">Введите Плотность
нашивки</label>
        <input type="text" name="Плотность нашивки" required>
    </td>
</tr>
</table>
<table align="center">
    <td align="center">
        <input type="submit" value="Рассчитать" class="button">
        <input type="reset" value="Сбросить" class="button">
    </td>
</table>
</fieldset>
</form>
<div class="result" align="center">
    { % if result % }

```

```
<br>
<p style="font-size:50px">Прогнозное значение параметра</p>
<p><b style="font-size:50px">{ { result } }</b></p>
{% endif %}
</div>
</body>
</html>
```


Листинг web-страницы main_2.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Прочность при растяжении, МПа</title>
    <style>
      div_1 {
        font-size: 200%;
      }
      form {
        margin: auto;
        width: 35%;
        background-color:#afafaf;
        width:600px;
        margin:1;
        align: center;
      }
      .button, button {
        appearance: none;
        border: 1;
        border-radius: 5px;
        background: #fff;
        color: #000;
        padding: 8px 16px;
        font-size: 16px;
      }
      .button:hover {
```

```

        background: #57d746;
    }
    button:hover {
        background: #57d746;
    }
    .result {
        margin: auto;
        width: 35%;
        border: 1px solid #ccc;
        background-color: #dbdbdb;
        width: 600px;
        margin: 1;
        align: center;
    }
</style>
</head>
<body>
    <a href="{ {url_for('index')}} ">
        <button>Назад</button>
    </a>
    <div_1>
        <p align="center">
            Прочность при растяжении, МПа
        </br>
        </p>
    </div_1>
    <form action="{ {url_for('main_2')}} " method="POST">
        <fieldset>
            <legend>Введите значение</legend>
            <table align="center">

```

```

<tr>
  <td align="right">
    <label for="Соотношение матрица-наполнитель">Введите
Соотношение матрица-наполнитель</label>
    <input type="text" name="Соотношение матрица-наполнитель"
required>
  </td>
</tr>
<tr>
  <td align="right">
    <label for="Плотность, кг/м3">Введите Плотность, кг/м3</label>
    <input type="text" name="Плотность, кг/м3" required>
  </td>
</tr>
<tr>
  <td align="right">
    <label for="модуль упругости, ГПа">Введите Модуль упругости,
ГПа</label>
    <input type="text" name="модуль упругости, ГПа" required>
  </td>
</tr>
<tr>
  <td align="right">
    <label for="Количество отвердителя, м.%">Введите Количество
отвердителя, м.%</label>
    <input type="text" name="Количество отвердителя, м.%" required>
  </td>
</tr>
<tr>
  <td align="right">

```

	<div><label for="Содержание эпоксидных групп,%_2">Введите Содержание эпоксидных групп,%_2</label> <div><input type="text" name="Содержание эпоксидных групп,%_2" required></div></div>
	</td>
	</tr>
	<tr>
	<td align="right"> <label for="Температура вспышки, C_2">Введите Температура вспышки, C_2</label> <input type="text" name="Температура вспышки, C_2" required>
	</td>
	</tr>
	<tr>
	<td align="right"> <label for="Поверхностная плотность, г/м2">Введите Поверхностная плотность, г/м2</label> <input type="text" name="Поверхностная плотность, г/м2" required>
	<td>
	</tr>
	<tr>
	<td align="right"> <label for="Потребление смолы, г/м2">Введите Потребление смолы, г/м2</label> <input type="text" name="Потребление смолы, г/м2" required>
	</td>
	</tr>
	<tr>
	<td align="right">

```

        <label for="Угол нашивки, град">Введите Угол нашивки,
град</label>
        <input type="text" name="Угол нашивки, град" min='0' max='90'
step='1' required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Шаг нашивки">Введите Шаг нашивки</label>
        <input type="text" name="Шаг нашивки" required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Плотность нашивки">Введите Плотность
нашивки</label>
        <input type="text" name="Плотность нашивки" required>
    </td>
</tr>
</table>
<table align="center">
    <td align="center">
        <input type="submit" value="Рассчитать" class="button">
        <input type="reset" value="Сбросить" class="button">
    </td>
</table>
</fieldset>
</form>
<div class="result" align="center">
    { % if result % }

```

```
<br>
<p style="font-size:50px">Прогнозное значение параметра</p>
<p><b style="font-size:50px">{ { result } }</b></p>
{% endif %}
</div>
</body>
</html>
```

Листинг web-страницы main_3.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Соотношение матрица-наполнитель</title>
    <style>
      div_1 {
        font-size: 200%;
      }
      form {
        margin: auto;
        width: 35%;
        background-color:#afafaf;
        width:600px;
        margin:1;
        align: center;
      }
      .button, button {
        appearance: none;
        border: 1;
        border-radius: 5px;
        background: #fff;
        color: #000;
        padding: 8px 16px;
        font-size: 16px;
      }
      .button:hover {
```

```

        background: #57d746;
    }
    button:hover {
        background: #57d746;
    }
    .result {
        margin: auto;
        width: 35%;
        border: 1px solid #ccc;
        background-color: #dbdbdb;
        width: 600px;
        margin: 1;
        align: center;
    }
</style>
</head>
<body>
    <a href="{ {url_for('index')}} ">
        <button>Назад</button>
    </a>
    <div_1>
        <p align="center">
            Соотношение матрица-наполнитель
        </br>
        </p>
    </div_1>
    <form action="{ {url_for('main_3')}} " method="POST">
        <fieldset>
            <legend>Введите значение</legend>
            <table align="center">

```



```

<tr>
  <td align="right">
    <label for="Плотность, кг/м3">Введите Плотность, кг/м3</label>
    <input type="text" name="Плотность, кг/м3" required>
  </td>
</tr>
<tr>
  <td align="right">
    <label for="модуль упругости, ГПа">Введите Модуль упругости,
ГПа</label>
    <input type="text" name="модуль упругости, ГПа" required>
  </td>
</tr>
<tr>
  <td align="right">
    <label for="Количество отвердителя, м.%">Введите Количество
отвердителя, м.%</label>
    <input type="text" name="Количество отвердителя, м.% " required>
  </td>
</tr>
<tr>
  <td align="right">
    <label for="Содержание эпоксидных групп,%_2">Введите
Содержание эпоксидных групп,%_2</label>
    <input type="text" name="Содержание эпоксидных групп,%_2"
required>
  </td>
</tr>
<tr>
  <td align="right">

```

```

        <label for="Температура вспышки, C_2">Введите Температура
вспышки, C_2</label>
        <input type="text" name="Температура вспышки, C_2" required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Поверхностная плотность, г/м2">Введите
Поверхностная плотность, г/м2</label>
        <input type="text" name="Поверхностная плотность, г/м2" required>
    <td>
</tr>
<tr>
    <td align="right">
        <label for="Модуль упругости при растяжении, ГПа">Введите
Модуль упругости при растяжении, ГПа</label>
        <input type="text" name="Модуль упругости при растяжении, ГПа"
required>
    <td>
</tr>
<tr>
    <td align="right">
        <label for="Прочность при растяжении, МПа">Введите Прочность
при растяжении, МПа</label>
        <input type="text" name="Прочность при растяжении, МПа"
required>
    <td>
</tr>
<tr>
    <td align="right">

```

```

        <label for="Потребление смолы, г/м2">Введите Потребление
смолы, г/м2</label>
        <input type="text" name="Потребление смолы, г/м2" required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Угол нашивки, град">Введите Угол нашивки,
град</label>
        <input type="text" name="Угол нашивки, град" min='0' max='90'
step='1' required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Шаг нашивки">Введите Шаг нашивки</label>
        <input type="text" name="Шаг нашивки" required>
    </td>
</tr>
<tr>
    <td align="right">
        <label for="Плотность нашивки">Введите Плотность
нашивки</label>
        <input type="text" name="Плотность нашивки" required>
    </td>
</tr>
</table>
<table align="center">
    <td align="center">
        <input type="submit" value="Рассчитать" class="button">

```

```

        <input type="reset" value="Сбросить" class="button">
    </td>
</table>
</fieldset>
</form>
<div class="result" align="center">
    { % if result % }
    <br>
    <p style="font-size:50px">Прогнозное значение параметра</p>
    <p><b style="font-size:50px">{ { result } }</b></p>
    { % endif % }
</div>
</body>
</html>

```