

**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

Григорьев Сергей Алексеевич

Москва, 2023

## Оглавление

Введение.....	4
1. Аналитическая часть.....	7
1.1. Постановка задачи.....	7
1.2. Описание используемых методов.....	8
1.2.1. Линейная регрессия.....	8
1.2.2. Метод К-ближайших соседей.....	9
1.2.3. Случайный лес.....	10
1.2.4. Метод опорных векторов.....	11
1.2.5. Многослойный персепtron.....	12
1.2.6. Статистическая линейная регрессия.....	13
1.2.7. Нейронная сеть.....	14
1.2.7.1. Понятие нейронной сети.....	14
1.2.7.2. Выбор параметров нейронной сети.....	16
1.3. Описание используемых библиотек.....	19
1.3.1. Подготовка среды для проекта ENV.....	19
1.3.2. Используемые библиотеки Python.....	20
1.4. Методы разведочного анализа данных.....	21
1.4.1. Разведочный анализ данных.....	21
1.4.2. Использованные в работе регрессионные методы.....	22
1.4.3. Метрики оценки.....	23
2. Практическая часть.....	24
2.1. Блок № 1 Разведочный анализ данных.....	25
2.1.1. Подготовка к исследовательской части.....	25
2.1.2. Первичный исследовательский (разведочный) анализ данных.....	31
2.1.3. Описательная статистика.....	33
2.1.4. Графическая часть статистического анализа.....	37
2.1.5. Глубокие исследования числовых значений статистического анализа.....	41
2.1.6. Выводы по Блоку № 1.....	47

2.2. Блок № 2 Корреляционный анализ данных.....	49
2.2.1. Исходные данные для Блока № 2.....	49
2.2.2. Общая корреляционная матрица.....	51
2.2.3. Гипотеза по сокращению числа выборки значений.....	55
2.2.4. Выбор параметров (фитч) для целевых переменных при анализе регрессионных моделей.....	56
2.2.5. Выводы по Блоку № 2.....	59
2.3. Блок № 3 Создание регрессионных моделей для прогноза ‘модуля упругости при растяжении’ и ‘прочности при растяжении’ .....	60
2.3.1. Исходные данные для работы Блока № 3.....	60
2.3.2. Создание регрессионных моделей.....	62
2.3.3 Тестирование регрессионных моделей.....	65
2.3.4. Выводы по Блоку № 3.....	72
2.4. Блок № 4 Создание нейронной сети, которая будет рекомендовать «соотношение матрица-наполнитель».....	73
2.4.1. Исходные данные для работы Блока № 4.....	73
2.4.2. Создание нейронной сети.....	74
2.4.2. Выводы по Блоку № 4.....	81
2.5. Разработка приложения.....	82
2.6. Создание удалённого репозитория и загрузка результатов работы на него.....	85
Заключение.....	87
Список используемой литературы и веб ресурсы.....	92
Приложения.....	95
Приложение А.....	95
Приложение Б (в PDF версии).....	96

## 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.

Композиционные материалы представляют собой многофазные системы, которые состоят из двух и более компонентов, сохраняющих индивидуальность (структуру и свойства) своего вещества в составе композита. Компонент, непрерывный в объеме композита, является матрицей или связующим элементом. Упрочняющие или армирующие элементы распределены в матрице в определенном порядке. Наполненные композиты содержат в матрице наполнители – дисперсные частицы неорганических и органических веществ, которые могут находиться в любой фазе. Разработка новых материалов имеет жизненно важное значение для решения насущных проблем в разных промышленных областях. Пространство гипотетических материалов, подлежащих рассмотрению, невероятно велико, и только небольшая часть возможных соединений может быть проверена экспериментально.

Уровень сложности разработки композитов очень высок из-за огромного количества компонентов и условий обработки, поскольку изменение одного фактора может иметь множество непредвиденных эффектов из-за их взаимосвязи. Традиционный подход к разработке новых материалов происходит в режиме «проб и ошибок», что приводит к проблемам низкой эффективности, высокой стоимости и неустойчивости полученных результатов.

Между тем, многочисленные экспериментальные и вычислительные испытания накапливают огромные объемы многомерных, сложных и недостаточно изученных данных, которые могут скрывать важные правила «структуро-

свойства». Машинное обучение может помочь выявить подобные связи, а также предоставить возможность тестировать и оптимизировать несколько показателей одновременно, тем самым ускоряя процесс обнаружения и оптимизации конструирования материалов, спрогнозировав состав с желаемыми характеристиками и уникальными свойствами.

Внедрение композиционных материалов обусловлено стремлением использовать их преимущества по сравнению с традиционно используемыми металлами и сплавами.

Для достижения определенных характеристик требуется большое количество различных комбинированных тестов, что делает насущной задачу прогнозирования успешного решения, снижающего затраты на разработку новых материалов и затраты на рабочую силу. 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. Работа основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана)

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

В процессе исследовательской работы были разработаны несколько моделей, способные с высокой вероятностью прогнозировать модули упругости при растяжении и прочности при растяжении, а также была создана нейронная сеть, которая предлагает расчет соотношение «матрица - наполнитель» на основе вводимых данных.

Для визуализации работы модели предсказания соотношение «матрица - наполнитель» было создано пользовательское веб - приложение на фреймворке Flask.

## 1. Аналитическая часть

### 1.1. Постановка задачи

Цель работы - разработать регрессионные модели для прогноза «Модуля упругости при растяжении» и «Прочности при растяжении» и написать нейронную сеть для прогнозирования «Соотношение матрица-наполнитель».

Полный текст задания по настоящей работе приведен в Приложении А.

Для решения поставленных задач были использованы методы машинного обучения.

### 1.2. Описание используемых методов

Машинное обучение (англ. machine learning, ML) — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение за счёт применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, математического анализа, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.

Постановка задачи по прогнозированию ряда конечных свойств получаемых композиционных материалов решается задачей регрессии.

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

#### 1.2.1. Линейная регрессия

Линейная регрессия (LinearRegression) - модель зависимости переменной  $x$  от одной или нескольких других переменных (факторов, регрессоров, незави-

симых переменных) с линейной функцией зависимости. Линейная регрессия относится к задаче определения «линии наилучшего соответствия» через набор точек данных и стала простым предшественником нелинейных методов, которые используют для обучения нейронных сетей.

Модель линейной регрессии является часто используемой и наиболее изученной. Это один из самых простых и эффективных инструментов статистического моделирования. А именно изучены свойства оценок параметров, получаемых различными методами при предположениях о вероятностных характеристиках факторов, и случайных ошибок модели. Предельные (асимптотические) свойства оценок нелинейных моделей также выводятся исходя из аппроксимации последних линейными моделями. Более важное значение имеет линейность по параметрам, чем линейность по факторам модели. Она определяет зависимость переменных с помощью линии наилучшего соответствия. Модель регрессии создаёт несколько метрик. R-квадрат, или коэффициент детерминации, позволяет измерить, насколько модель может объяснить дисперсию данных. Если R-квадрат равен 1, это значит, что модель описывает все данные. Если же R-квадрат равен 0,5, модель объясняет лишь 50 процентов дисперсии данных. Оставшиеся отклонения не имеют объяснения. Чем ближе R-квадрат к единице, тем лучше.

**Преимущества:** хорошо изучены; очень быстрые, могут работать на очень больших выборках; практически вне конкуренции, когда признаков очень много (от сотен тысяч и более), и они разреженные (хотя есть еще факторизационные машины); коэффициенты перед признаками могут интерпретироваться (при условии, что признаки масштабированы; легко интерпретируем; имеет меньшую сложность по сравнению с другими алгоритмами).

**Недостатки:** плохо работают в задачах, в которых зависимость ответов от признаков сложная, нелинейная; моделирует только прямые линейные зави-

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

### 1.2.2. Метод К-ближайших соседей

Метод (kNN - k Nearest Neighbours) ищет ближайшие объекты с известными значениями целевой переменной и основывается на хранении данных в памяти для сравнения с новыми элементами. 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:** прост в реализации и понимании полученных результатов; имеет низкую чувствительность к выбросам; не требует построения модели; допускает настройку нескольких параметров; позволяет делать дополнительные допущения; универсален; находит лучшее решение из возможных; решает задачи небольшой размерности.

**Недостатки:** замедляется с ростом объёма данных; не создаёт правил; не обобщает предыдущий опыт; основывается на всем массиве доступных исторических данных; невозможно сказать, на каком основании строятся ответы; сложно выбрать близость метрики; имеет высокую зависимость результатов классификации от выбранной метрики; полностью перебирает всю обучающую выборку при распознавании; имеет вычислительную трудоёмкость.

### 1.2.3. Случайный лес

Случайный лес (RandomForest) - алгоритм машинного обучения, предложенный Лео Брейманом и Адель Катлер, заключающийся в использовании комитета (ансамбля) решающих деревьев. Алгоритм сочетает в себе две основ-

ные идеи: метод бэггинга Бреймана, и метод случайных подпространств, предложенный Тин Кам Хо. Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого ансамбля решающих деревьев, каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим.

**Преимущества:** способность эффективно обрабатывать данные с большим числом признаков и классов; нечувствительность к масштабированию (и вообще к любым монотонным преобразованиям) значений признаков; одинаково хорошо обрабатываются как непрерывные, так и дискретные признаки; существуют методы построения деревьев по данным с пропущенными значениями признаков; существуют методы оценивания значимости отдельных признаков в модели; внутренняя оценка способности модели к обобщению (тест по неотобранным образцам); высокая параллелизуемость и масштабируемость.

**Недостатки:** большой размер получающихся моделей; требуется большой объем **памяти** для хранения модели; построение занимает много времени; сложно интерпретируемый; не обладает возможностью экстраполяции; может недо обучаться; трудоёмко прогнозируемый; иногда работает хуже, чем линейные методы.

#### 1.2.4. Метод опорных векторов

Метод опорных векторов (support vector machine, SVM) — один из наиболее популярных методов машинного обучения. Он создает гиперплоскость или набор гиперплоскостей в многомерном пространстве, которые могут быть использованы для решения задач классификации и регрессии.

Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Интуитивно, хорошее разде-

ление достигается за счет гиперплоскости, которая имеет самое большое расстояние до ближайшей точки обучающей выборке любого класса. Максимально близкие объекты разных классов определяют опорные вектора. Если в исходном пространстве объекты линейно неразделимы, то выполняется переход в пространство большей размерности.

Эффективность метода опорных векторов зависит от выбора ядра, параметров ядра и параметра для регуляризации.

**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 увереной классификации; при правильной работе модели, построенной на тестовом множестве, вполне возможно применение данного метода на реальных данных; достаточно небольшого набора данных.

**Недостатки:** 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 С при помощи кросс-валидации; подходит только для решения задач с 2 классами; параметры модели сложно интерпретировать.

### 1.2.5. Многослойный персепtron.

Многослойный персептрон (MLPRegressor) — это алгоритм обучения с учителем, который изучает функцию обучением на наборе данных. Это искус-

ственная нейронная сеть, имеющая 3 или более слоёв персепtronов. Эти слои - один входной слой, 1 или более скрытых слоёв и один выходной слой персепtronов.

Преимущества: построение сложных разделяющих поверхностей; возможность осуществления любого отображения входных векторов в выходные; легко обобщает входные данные; не требует распределения входных векторов; изучает нелинейные модели.

Недостатки: имеет невыпуклую функцию потерь; разные инициализации случайных весов могут привести к разной точности проверки; требует настройки ряда гиперпараметров; чувствителен к масштабированию функций.

### **1.2.6. Статистическая линейная регрессия.**

Статистическая линейная регрессия sm.OLS() – statsmodels regression linear\_model - регрессия методом наименьших квадратов

Остатки (residuals) — это то же самое, что и ошибка, т.е. отклонение предсказания от истинного значения.

R-squared — это коэффициент детерминации.

F-statistic - критерий Фишера проверки гипотезы о равенстве дисперсий.

В данном случае сравнивается модель с нулевыми параметрами (кроме интерсепта) с текущей.

Prob. (F-statistic) - вероятность того, что коэффициенты при всех переменных равны нулю.

Log-likelihood - насколько хорошо данные описываются моделью.

AIC BIC как критерий для отбора модели, который определяет на сколько сильно модель переобучена.

Чем меньше значение - тем лучше модель.

$P > |t|$ . Отдельные р-значения говорят нам, является ли каждая предикторная переменная статистически значимой.

Intercept - все переменные обнуляются и подбирается константа, наилучшим образом описывающая данные. Значение У при X=0

std err - дисперсия коэффициента по точкам данных.

t-stat - статистика t-Стьюарта, чем больше — тем лучше измерен коэффициент (чем меньше стандартное отклонение, тем лучше коэффициент описывает зависимость). Обычно t-критерий позволяет нам оценить важность различных предикторов, предполагая, что остатки модели нормально распределены около нуля. Если остатки не ведут себя таким образом, то это говорит о наличии некоторой нелинейности между переменными и о том, что их t-тесты не следует использовать для оценки важности отдельных предикторов.

## 1.2.7. Нейронная сеть

### 1.2.7.1. Понятие нейронной сети

Нейронная сеть - математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Это понятие возникло при изучении процессов, протекающих в мозге, и при попытке смоделировать эти процессы. Первой такой попыткой были нейронные сети У. Маккалока и У. Питтса. После разработки алгоритмов обучения получаемые модели стали использовать в практических целях: в задачах прогнозирования, для распознавания образов, в задачах управления и др.

Нейронная сеть представляет собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Такие процессоры обычно довольно просты (особенно в сравнении с процессорами, используемыми в персональных компьютерах). Каждый процессор подобной

сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посыпает другим процессорам. И, тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие по отдельности простые процессоры вместе способны выполнять довольно сложные задачи.

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке, а также неполных и/или «зашумленных», частично искажённых данных.

Входные данные передаются в эти нейроны в виде линейной комбинации со множеством переменных. Значение, умножаемое на каждую функциональную переменную, называется весом. Затем к этой линейной комбинации применяется нелинейность, что даёт нейронной сети возможность моделировать сложные нелинейные отношения.

Технически обучение заключается в нахождении коэффициентов связей между нейронами. У нейрона есть функция активации, которая определяет выходное значение нейрона. Она используется для того, чтобы ввести нелинейность в нейронную сеть.

У полносвязной нейросети выход каждого нейрона подается на вход всем нейронам следующего слоя. У нейросети имеется:

- входной слой — его размер соответствует входным параметрам;
- скрытые слои — их количество и размерность определяют специалист;
- выходной слой — его размер соответствует выходным параметрам.

Прямое распространение – это процесс передачи входных значений в нейронную сеть и получения выходных данных, которые называются прогнозируемым значением.

Прогнозируемое значение сравниваем с фактическим с помощью функции потери. В методе обратного распространения ошибки градиенты (производные значений ошибок) вычисляются по значениям весов в направлении, обратном прямому распространению сигналов. Значение градиента вычитают из значения веса, чтобы уменьшить значение ошибки. Таким образом происходит процесс обучения. Обновляются веса каждого соединения, чтобы функция потерь минимизировалась.

Нейронные сети применяются для решения задач регрессии, классификации, распознавания образов и речи, компьютерного зрения и других. На настоящий момент это самый мощный, гибкий и широко применяемый инструмент в машинном обучении.

### 1.2.7.2. Выбор параметров нейронной сети.

#### Количество слоев.

Элементы нейронной сети Входной слой: этот слой принимает входные функции. Он предоставляет информацию из внешнего мира в сеть, на этом уровне вычисления не выполняются, узлы здесь просто передают информацию (функции) скрытому слою.

Скрытый слой / слои — это слой не подвергается воздействию внешнего мира, он являются частью абстракции, предоставляемой любой нейронной сетью. Скрытый слой выполняет все виды вычислений над объектами, введенными через входной слой, и передает результат на выходной слой.

Выходной уровень: этот уровень выводит информацию, полученную сетью, во внешний мир.

В настоящее время нет никаких жестких правил ни для выбора количества скрытых слоев, ни для выбора количества нейронов в них. Хотя существуют ограничения, помогающие принимать решения. Критерии выбора количества слоев.

Если функция определена на конечном множестве точек, непрерывна и определена на компактной области, то 3-ехслойный перцептрон способен ее аппроксимировать.

Остальные функции, которым могут быть обучены нейронные сети, могут быть аппроксимированы 4-ехслойным перцетроном. Таким образом, теоретически максимальное количество слоев, которое необходимо – четыре, или два скрытых слоя.

### **Количество нейронов в скрытых слоях.**

Слишком малое количество – и сеть не сможет обучиться. Слишком большое повлечет за собой увеличение времени обучения сети до фактически нереального значения. Также это может привести к переобученности сети (overfitting), проявляющейся в том, что сеть будет прекрасно работать на обучающей выборке, но очень плохо на входных примерах, не входящих в нее.

Это происходит из-за того, что сеть будет обладать избыточными способностями к обучению и наряду со значительными для данной задачи факторами будет учитывать черты, характерные лишь для данной обучающей выборки.

Однако, существуют эвристические правила выбора количества нейронов в скрытых слоях. Одним из таких правил является правило геометрической пирамиды (geometric pyramid rule).

Для 3-ех слойного перцептрана, по этому правилу, число нейронов скрытого слоя вычисляется по следующей формуле:  $k = \sqrt{n*m}$  где  $k$  – число нейронов в скрытом слое,  $n$  – число нейронов во входном слое;  $m$  – число нейронов в выходном слое.

Для 4-ехслойного перцептрана число нейронов вычисляется несколько сложнее:  $r = \text{sqrt}^3(n/m)$  ( кубический корень) где  $k1 = m * r2$  - число нейронов в первом скрытом слое;  $k2 = m * r$  - число нейронов во втором скрытом слое.

### Выбор Активационных функций.

Функция активации решает, следует ли активировать нейрон или нет, путем вычисления взвешенной суммы и дальнейшего добавления к ней смещения. Целью функции активации является введение нелинейности в выходной сигнал нейрона.

Варианты функции активации Линейная функция Использование: Функция линейной активации используется только в одном месте, т.е. на выходном уровне.

Функция RELU это означает выпрямленную линейную единицу. Это наиболее широко используемая функция активации. В основном реализованы в скрытых слоях нейронной сети. Природа: - нелинейная, что означает, что мы можем легко распространять ошибки в обратном направлении и иметь несколько слоев нейронов, активируемых функцией ReLU. Использует: - ReLU менее дорогостоящий в вычислительном отношении, чем tanh и sigmoid, поскольку он включает в себя более простые математические операции. Одновременно активируется только несколько нейронов, что делает сеть разреженой, что делает ее эффективной и простой для вычислений. Проще говоря, RELU обучается намного быстрее, чем sigmoid и функция Tanh.

Функция активации SELU масштабированная экспоненциальная линейная единица или функцию активации SELU масштабируемые экспоненциальные линейные единицы" (SELU) Эта функция активации, которая обладает свойствами самонормализации, гарантирует, что все выходные данные будут нормализованы без явного добавления слоя нормализации в вашу модель. Что лучше, так это

то, что его можно использовать относительно легко и что он обеспечивает адекватные результаты, важно:

При использовании SELU необходимо использовать инициализатор LecunNormal() - kernel\_initializer='lecun\_normal' и использовать Alpha Dropout().

Масштабируемая экспоненциальная линейная единица или функция активации SELU может использоваться для объединения эффектов RELU и пакетной нормализации, что устраняет необходимость в использовании BatchNorm.

**Преимущества:** высокая точность классификации; сильная способность параллельной распределенной обработки, сильная распределенная память и способность к обучению; он обладает высокой устойчивостью и отказоустойчивостью к шумовым нервам и может полностью аппроксимировать сложные нелинейные отношения; с функцией ассоциативной памяти.

**Недостатки:** нейронные сети требуют большого количества параметров, таких как топология сети, начальные значения весов и пороговых значений; невозможно наблюдать процесс обучения между ними, а полученные результаты трудно объяснить, что повлияет на достоверность и приемлемость результатов; время обучения слишком велико и может даже не достичь цели обучения.

### 1.3. Описание используемых библиотек

#### 1.3.1. Подготовка среды для проекта ENV.

При начале работы над проектом необходимо было установить несколько пакетов библиотек Python таких как tensorflow, seaborn и др. Выяснилось, что невозможно установить пакеты через PIP install, так как win 10 не воспринимает путь к файлу длиннее 260 символов. Пришлось написать программу по изменению Политики доступа для win 10 с расширением .bat. Однако это тоже не дало результатов. Пришлось с помощью изменения regedit вносить изменения в ключ

Computer\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem, что привело к возможности записывать блинные пути к файлу.

### 1.3.2. Используемые библиотеки Python.

1. NumPy. Общедоступный модуль для Python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций, обеспечивает функционал, который можно сравнить с функционалом MatLab. NumPy (Numeric Python) предоставляет базовые методы для манипуляции с большими массивами и матрицами.

2. Matplotlib. Библиотека на языке программирования Python для визуализации данных двумерной (2D) графикой (3D графика также поддерживается). Получаемые изображения могут быть использованы в качестве иллюстраций в публикациях.

3. Pandas. Данная библиотека делает Python мощным инструментом для анализа данных. Пакет дает возможность строить сводные таблицы, выполнять группировки, предоставляет удобный доступ к табличным данным.

4. Scikit-learn. Один из наиболее широко используемых пакетов Python для Data Science и Machine Learning. Он содержит функции и алгоритмы для машинного обучения: классификации, прогнозирования или разбивки данных на группы.

5. Seaborn. Библиотека для создания статистических графиков на Python. Она основывается на matplotlib и тесно взаимодействует со структурами данных pandas.

6. TensorFlow. Открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия.

7. Math является самым базовым математическим модулем Python. Охватывает основные математические операции, такие как сумма, экспонента, модуль и так далее. Эта библиотека не используется при работе со сложными математическими операциями, такими как умножение матриц.

8. Pickle реализует мощный алгоритм сериализации и десериализации объектов Python. Широко применяется для сохранения и загрузки сложных объектов в Python как поток байтов.

## 1.4. Методы разведочного анализа данных

### 1.4.1. Разведочный анализ данных

Цель разведочного анализа - получение первоначальных представлений о характеристах распределений переменных исходного набора данных, формирование оценки качества исходных данных (наличие пропусков, выбросов), выявление характера взаимосвязи между переменными с целью последующего выдвижения гипотез о наиболее подходящих для решения задачи моделях машинного обучения.

В качестве инструментов разведочного анализа используется: оценка статистических характеристик датасета; гистограммы распределения каждой из переменной (несколько различных вариантов); диаграммы ящика с усами (несколько интерактивных вариантов); попарные графики рассеяния точек (несколько вариантов); тепловая карта (несколько вариантов); описательная статистика для каждой переменной; анализ и полное исключение выбросов (5 повторных итераций); проверка наличия пропусков и дубликатов; корреляционный анализ для установления статистических зависимостей между переменными.

## Корреляционный анализ

Корреляционный анализ - статистический метод изучения взаимосвязи между двумя и более случайными величинами. В качестве случайных величин в эмпирических исследованиях выступают значения переменных, измеряемые свойства исследуемых объектов наблюдения. Суть корреляционного анализа заключается в расчете коэффициентов корреляции. Коэффициенты корреляции могут принимать, как правило, положительные и отрицательные значения. Знак коэффициента корреляции позволяет интерпретировать направление связи, а абсолютное значение – силу связи.

Способ расчета коэффициентов корреляции зависит от шкал измерения переменных, между которыми исследуется взаимосвязь. Для переменных, измеряемых в количественной шкале (интервальной шкале или шкале отношений), рассчитывают ковариацию или корреляционный момент, а на его основе линейный коэффициент корреляции (коэффициент корреляции Пирсона).

Для оценки силы и направления связи между переменными, измеренными в порядковой шкале, используются непараметрические ранговые коэффициенты корреляции.

Прежде чем передать данные в работу моделей машинного обучения, необходимо обработать и очистить их. Очевидно, что «грязные» и необработанные данные могут содержать искажения и пропущенные значения – это ненадёжно, поскольку способно привести к крайне неверным результатам по итогам моделирования. Но безосновательно удалять что-либо тоже неправильно. Именно поэтому сначала набор данных надо изучить.

После обнаружения выбросов данные, значительно отличающиеся от выборки, полностью удаляются.

#### 1.4.2. Использованные в работе регрессионные методы

Для решения задач были использованы следующие регрессионные методы машинного обучения:

1. LinearRegression - линейная регрессия
2. KNeighborsRegressor - метод ближайших соседей
3. RandomForestRegressor - случайный лес
4. SVR - Support Vector Regressor метод опорных векторов
5. MLPRegressor - многослойный перцептрон
6. sm.OLS - Ordinary Least Squares статистическая линейная регрессия  
- регрессия методом наименьших квадратов

#### 1.4.3. Метрики оценки.

Для оценки работы моделей используется разнообразные метрики оценки.

- MAPE - Mean Absolute Percentage Error - Средняя абсолютная процентная ошибка средняя разница между прогнозируемым значением и фактическим значением.

- MAE - mean\_absolute\_error - Средняя абсолютная ошибка. Это среднее абсолютных разностей между фактическими целевыми значениями и прогнозами. Чем ниже MAE для данной модели, тем точнее модель способна предсказать фактические значения.

- MSE - mean\_squared\_error - Среднеквадратическая ошибка. Эта метрика сообщает среднеквадратичную разницу между прогнозируемыми значениями и фактическими значениями в наборе данных. Чем ниже MSE, тем лучше модель соответствует набору данных

- MaxER - max\_error - Максимальная ошибка  
- RMSE - Root Mean Squared Error- Корень из среднеквадратичной ошибки. Эта метрика сообщает нам квадратный корень из средней квадратич-

ной разницы между прогнозируемыми значениями и фактическими значениями в наборе данных. Эта метрика измеряется в тех же единицах, что и переменная ответа. Чем ниже RMSE, тем лучше модель соответствует набору данных.

- R2 - Коэффициент детерминации - функция оценки регрессии. Он показывает долю дисперсии зависимой переменной, объяснённой с помощью регрессионной модели. Если значение R2 равно 1, это означает, что модель идеальна, а если ее значение равно 0, это означает, что модель будет плохо работать с неизвестным набором данных.

- SCORES - Оценка кросс-валидации
- MCVS - Среднее значение оценки кросс-валидации
- StdDS - Статистическая ошибка оценки кросс-валидации

## 2. Практическая часть

В процессе написания ВКР:

1. Для сокращения длины рабочего кода, часто повторяющиеся части кода собраны в вызываемые модули-функции, которые сохранены в директории `modules_def`.
2. В процессе работы все получаемые графики сохранялись в директории `save_fig/fig_blockN`, где N -номер Блока части исследования от 1 до 5.
3. Создаваемые в процессе работы по Блокам датасеты и модели результатов работы регрессий и нейронной сети, сохранялись в конце каждого блока в директорию `data_storage/data_blockN`, где N -номер Блока части исследования от 1 до 5.
4. Получены полные отчеты по двум датасетам- по исходному объединенному датасету и по очищенному от выбросов, причем это сделано двумя раз-

ными методами: ProfileReport и SweetViz. Отчеты сформированы и представлены в директории profile\_report в исходных html и PDF форматах

5. Представлен 3-й вариант Рабочего Проекта, который разделен на 5 частей в соответствии с решаемыми в каждом Блоке задачами, изложенными в Задании на ВКР.

## 2.1. Блок № 1 Разведочный анализ данных

### 2.1.1. Подготовка к исследовательской части

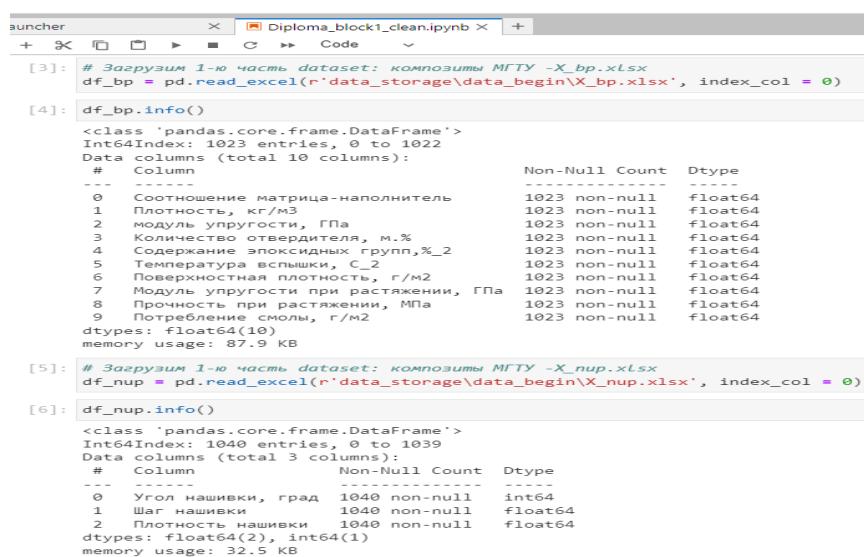
Датасет для исследования расположен по адресу в Интернете:

[https://drive.google.com/file/d/1B1s5gBlvgU81H9GGolLQVw\\_SOi-vyNf2/view?usp=sharing](https://drive.google.com/file/d/1B1s5gBlvgU81H9GGolLQVw_SOi-vyNf2/view?usp=sharing)

и затем был сохранен в папку data\_storage/data\_begin/.

Для исследовательской работы были даны 2 файла: X\_bp.xlsx (с данными о параметрах базальтопластика, состоящий из 1023 строки и 11 столбцов) и X\_nup.xlsx (с данными нашивок углепластика, состоящий из 1040 строк и 4 столбцов). Причем и в первом и во втором наборе имеется дополнительный столбец индексов строк без названия, которые удалялись при загрузки в Jupyter-Lab.

Параметры этих датасетов приведены на Рис.1.



```
[3]: # Загрузим 1-ю часть dataset: композиты МГТУ -X_bp.xlsx
df_bp = pd.read_excel(r'data_storage\data_begin\X_bp.xlsx', index_col = 0)

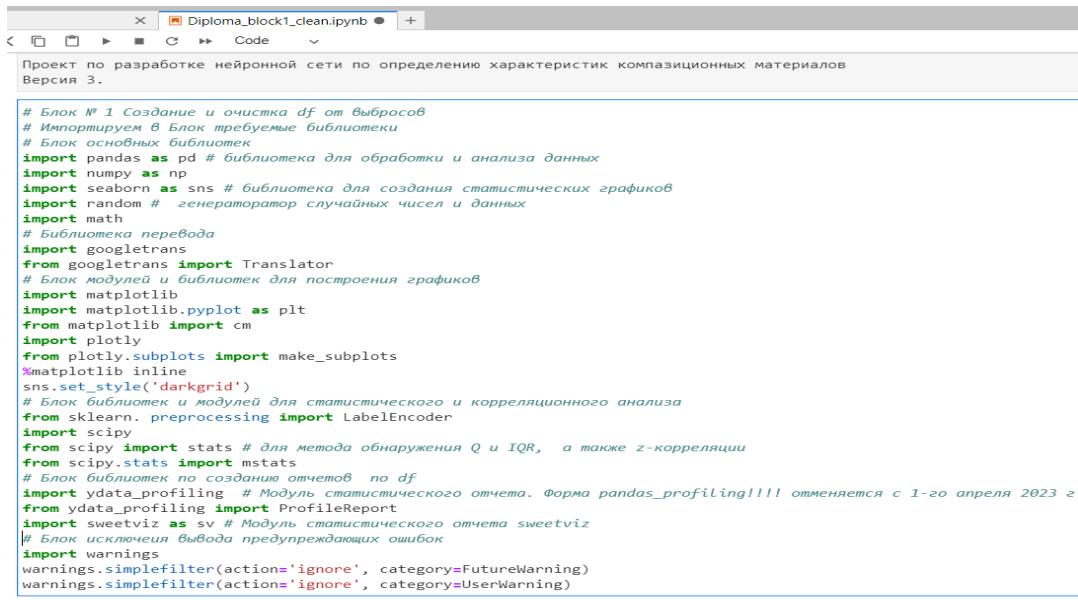
[4]: df_bp.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1023 entries, 0 to 1022
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Соотношение матрица-наполнитель    1023 non-null   float64 
 1   Плотность, кг/м3                  1023 non-null   float64 
 2   Модуль упругости, ГПа            1023 non-null   float64 
 3   Количество отверстий, 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 
dtypes: float64(10)
memory usage: 87.9 KB

[5]: # Загрузим 1-ю часть dataset: композиты МГТУ -X_nup.xlsx
df_nup = pd.read_excel(r'data_storage\data_begin\X_nup.xlsx', index_col = 0)

[6]: df_nup.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1040 entries, 0 to 1039
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Угол нашивки, град      1040 non-null   int64  
 1   Шаг нашивки             1040 non-null   float64 
 2   Плотность нашивки       1040 non-null   float64 
dtypes: float64(2), int64(1)
memory usage: 32.5 KB
```

Рис.1. Параметры заданных датасетов

Библиотеки и модули Python, которые использовались в данном Блоке.



```
# Блок № 1 Создание и очистка df от выбросов
# Импортируем в Блок требуемые библиотеки
# Блок основных библиотек
import pandas as pd # библиотека для обработки и анализа данных
import numpy as np
import seaborn as sns # библиотека для создания статистических графиков
import random # генератором случайных чисел и данных
import math
# Библиотека перевода
import googletrans
from googletrans import Translator
# Блок модулей и библиотек для построения графиков
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
import plotly
from plotly.subplots import make_subplots
%matplotlib inline
sns.set_style('darkgrid')
# Блок библиотек и модулей для статистического и корреляционного анализа
from sklearn.preprocessing import LabelEncoder
import scipy
from scipy import stats # для метода обнаружения Q и IQR, а также z-корреляции
from scipy.stats import mstats
# Блок библиотек по созданию отчетов по df
import ydata_profiling # Модуль статистического отчета. Форма pandas_profiling!!!! отменяется с 1-го апреля 2023 г
from ydata_profiling import ProfileReport
import sweetviz as sv # Модуль статистического отчета sweetviz
# Блок исключения вывода предупреждающих ошибок
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=UserWarning)
```

Рис. 2. Библиотеки и модули Python для Блока №1.

Подключаемые в данном Блоке №1 функции, которые разработаны для всего проекта и которые находятся в папке `modules_def`, приведены на Рис. 3.

```
56]: # Блок импорта созданных функций для проекта из папки modules_def
%run ./modules_def/optimize_memory_usage.ipynb # функция оптимизации размера df
%run ./modules_def/drawing_graphs.ipynb # функция рисования разных графиков
%run ./modules_def/sns_plt_PairGrid.ipynb # функция рисования матрицы графиков
%run ./modules_def/translate_list_data.ipynb # функция перевода списка с русского на англ
%run ./modules_def/mean_round_point.ipynb # функция расчета среднего значения вокруг подозрительного выброса
%run ./modules_def/outliers_IQR_delit.ipynb # функция очистки выбросов по методу IQR
%run ./modules_def/Z_score.ipynb 3 функция очистки выбросов методом Z_score
```

Рис.3 Подключаемые функции

Всего осуществлена проработка 3-х вариантов исследования с разным подходом к значениям параметров, предоставленных для ВКР.

**В первом варианте** в процессе работы над ВКР для датасета объединенного по методу INNER был первоначально проведен полный анализ, вплоть до построения моделей, причем предварительно в `X_nip.xlsx` удалялись лишние 17 строк. Однако ничего не удалось продемонстрировать и никакой линейной зависимости выявлено не было.

**Во втором варианте** исследования над объединенным датасетом были проделаны манипуляции по приданию значениям параметров нелинейности – возведение в квадрат, в куб и логарифмирование всех параметров, что также не привело к улучшению показателей работы регрессионных моделей.

Все файлы по первым двум вариантам исследования приведены в папке `Diploma_naive_old/`.

**В третьем варианте**, который далее рассматривается, задача решалась на абсолютно другом подходе к формированию объединенного датасета.

Так как в обоих датасетах `X_bp.xlsx` (состоящий из 1023 строки и 10 столбцов) и `X_nip.xlsx` (1040 строк и 3 столбца) наблюдалась значительная детерминированность данных, то было принято решение, после анализа первых 40 строк данных обоих файлов (Рис. 4 и 5) об удалении детерминированных данных.

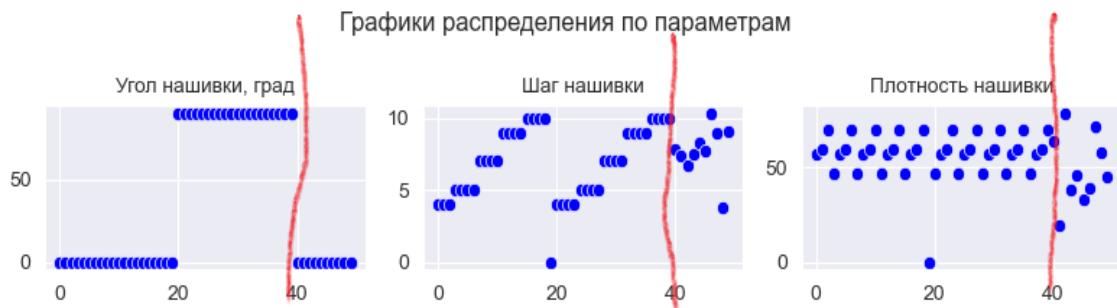


Рис. 4 Детерминированные данные `df_nip`

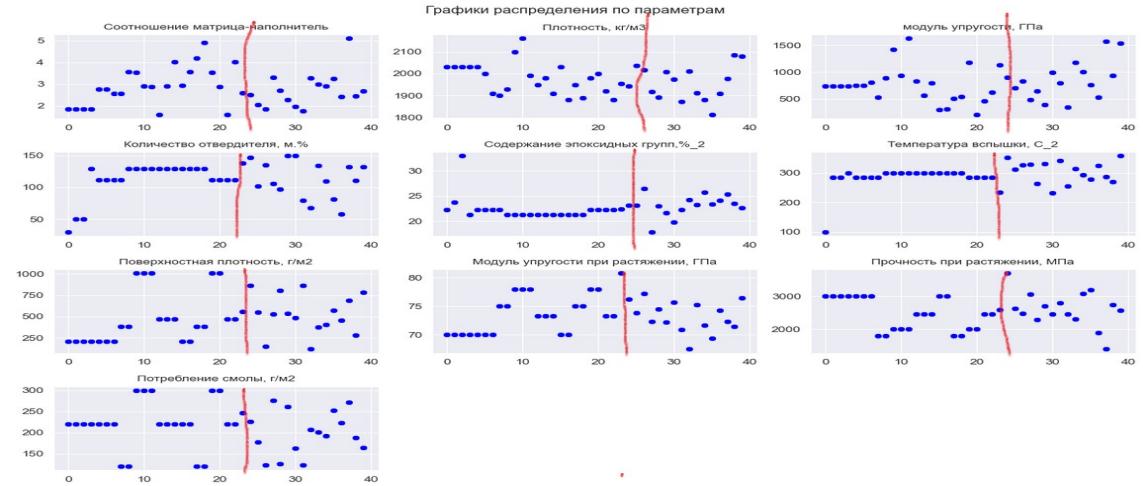


Рис. 5 Детерминированные данные df\_bp

В df\_nup в диапазоне от 0 до 40 строки наблюдается явная детерминированность показателей, что свидетельствует об искусственном искажении датасета. Удалим первые 40 строк из df\_nup как явно искаженные, отбросив первые 40 строк:  $1040 - 40 = 1000$  строк. В df\_bp в диапазоне от 0 до 23 строк также наблюдается явная детерминированность показателей, что также свидетельствует об искусственном искажении df\_bp. В df\_bp отбросим начальные 23 строки:  $1023 - 23 = 1000$ . Размерность наших массивов будет соблюдена и составит 1000 строк. Объединим df\_nup.drop и df\_bp-drop методом INNER в датасет data\_main, предварительно обнуляем индексы соединяемых файлов.

Набор данных `data_main` в итоге содержит 13 столбцов по 1000 строк.

```
[21]: data_main.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Угол нашивки, град      1000 non-null   int64  
 1   Шаг нашивки            1000 non-null   float64 
 2   Плотность нашивки     1000 non-null   float64 
 3   Соотношение матрица-наполнитель 1000 non-null   float64 
 4   Плотность, кг/м3        1000 non-null   float64 
 5   модуль упругости, ГПа    1000 non-null   float64 
 6   Количество отвердителя, м.% 1000 non-null   float64 
 7   Содержание эпоксидных групп,%_2 1000 non-null   float64 
 8   Температура вспышки, С_2     1000 non-null   float64 
 9   Поверхностная плотность, г/м2 1000 non-null   float64 
 10  Модуль упругости при растяжении, ГПа 1000 non-null   float64 
 11  Прочность при растяжении, МПа    1000 non-null   float64 
 12  Потребление смолы, г/м2       1000 non-null   float64 
dtypes: float64(12), int64(1)
memory usage: 101.7 KB
```

Рис. 6. Информация по объединённому датасету `data_main`

Набор данных `data_main` содержит только числовые значения типа `float 64` и `int64` и все данные не нулевые, нет пропущенных данных.

В документации по Python существует рекомендация использования всех наименований параметров – убрать многословные названия столбцов с символами, словами в верхнем и нижнем регистре и пробелами и опечатками. Чтобы упростить выборку данных по имени столбца, осуществим перевод названий столбцов с русского языка на английский.

Рекомендуется использовать: английский язык, нижний регистр, удалить специальные символы и заменить пробелы символами подчёркивания.

Перевод осуществлен созданной функцией перевода `translate_list_data`, которая использует модуль перевода от Google translator.translate (Рис. 6)

```
: # Заменим названия столбцов с русского языка на английский язык
# Вызываем функцию перевода -translate_list_data из директории modules_def

list_new_name = translate_list_data(list_column, 'ru', 'en', prt = True) # Функция перевода названий столбцов

['Угол нашивки, град', 'Шаг нашивки', 'Плотность нашивки', 'Соотношение матрица-наполнитель', 'Плотность, кг/м³', 'модуль упругости, ГPa', 'Количество о твердителя, м.%', 'Содержание эпоксидных групп,%_2', 'Температура вспышки, С_2', 'Поверхностная плотность, г/м²', 'Модуль упругости при растяжении, ГП а', 'Прочность при растяжении, МПа', 'Потребление смолы, г/м²'] -> ['Patch angle, hail', 'Step of the strip', 'Density_strip', 'Ratio_filler_matrix', 'Density', 'Elasticity module, GPa', 'The number of hardeners, m.%', 'The content of epoxy groups,%_2', 'Flash temperature, s_2', 'Surface density, g/m²', 'Elasticity module for stretching, GPa', 'Strapery strength, MPa', 'Resin consumption, g/m²']
```

Рис.7 Перевод названий параметров на английский язык

Заменяем русские названия английскими с помощью созданной функции `changelang_column_name`. И с помощью модуля `lower()`, переведем все названия в нижний регистр для дальнейшего удобного использования названий признаков.

Приведем имена столбцов к нормальному виду - уберем пробелы и спецсимволы создав словарь (Рис. 8).

```
[28]: # Приведем имена столбцов к нормальному виду - уберем пробелы и спецсимволы создав словарь
data_main.rename(columns={
    'patch angle, hail': 'pattern_angle',
    'step of the strip': 'step_strip',
    'the density of the strip': 'density_strip',
    'the ratio of the filler matrix': 'ratio_filler_matrix',
    'density, kg/m³': 'density',
    'elasticity module, gpa': 'elasticity_module',
    'the number of hardeners, m.%': 'number_hardeners',
    'the content of epoxy groups,%_2': 'content_epoxy_groups',
    'flash temperature, s_2': 'flash_temperature',
    'surface density, g/m²': 'surface_density',
    'elasticity module for stretching, gpa': 'elasticity_module_stretching',
    'strapery strength, mpa': 'strapery_strength',
    'resin consumption, g/m²': 'resin_consumption'
}, inplace=True)
```

Рис. 8 Новые названия столбцов- параметров

В итоге получаем датасет соответствующий требованиям Python (Рис. 9).

```
[29]: data_main.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
 --- 
 0   pattern_angle   1000 non-null   int64  
 1   step_strip      1000 non-null   float64 
 2   density_strip   1000 non-null   float64 
 3   ratio_filler_matrix 1000 non-null   float64 
 4   density         1000 non-null   float64 
 5   elasticity_module 1000 non-null   float64 
 6   number_hardeners 1000 non-null   float64 
 7   content_epoxy_groups 1000 non-null   float64 
 8   flash_temperature 1000 non-null   float64 
 9   surface_density  1000 non-null   float64 
 10  elasticity_module_stretching 1000 non-null   float64 
 11  strapery_strength 1000 non-null   float64 
 12  resin_consumption 1000 non-null   float64 
dtypes: float64(12), int64(1)
memory usage: 101.7 KB
```

Рис. 9 Датасет с новыми названиями параметров

Вызываем функцию `optimize_memory_usage` из директории `modules_def` для уменьшения размерности переменных в файле `data_main` и облегчения расчетов. Эта функция будет широко использоваться в дальнейшей работе.

```
[30]: # Вызываем функцию optimize_memory_usage из директории modules_def
# для уменьшения размерности переменных в файле data_main и облегчения расчетов

data_main = optimize_memory_usage(data_main, print_size=True, print_info_before=False)

Memory usage size: before 0.0993 Mb - after 0.0497 Mb (49.9%).
-----
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   pattern_angle    1000 non-null   int32  
 1   step_strip       1000 non-null   float32 
 2   density_strip   1000 non-null   float32 
 3   ratio_filler_matrix 1000 non-null   float32 
 4   density          1000 non-null   float32 
 5   elasticity_module 1000 non-null   float32 
 6   number_hardeners 1000 non-null   float32 
 7   content_epoxy_groups 1000 non-null   float32 
 8   flash_temperature 1000 non-null   float32 
 9   surface_density   1000 non-null   float32 
 10  elasticity_module_stretching 1000 non-null   float32 
 11  strapery_strength 1000 non-null   float32 
 12  resin_consumption 1000 non-null   float32 
 dtypes: float32(12), int32(1)
memory usage: 50.9 KB
None
```

Рис. 10 Функция снижения размерности датасета

Мы снизили размерность данных с 64 байт до 32 байт и общий объем памяти файла сократился на 49,9%.

### 2.1.2. Первичный исследовательский (разведочный) анализ данных

Исследовательский анализ данных — это подход к анализу наборов данных для окончательного определения их основных характеристик, часто с использованием визуальных методов для понимания данных.

В среде Python есть модули отчетов по исследуемым датасетам.

Первый модуль, с помощью которого можно получить полный отчет по датасету - `ProfileReport`, а второй - `SweetVIZ`.

Полученные отчеты с помощь этих модулей в форматах html и PDF по первичному, неочищенному датасету, приведены в папке profile\_report и в Приложении Б.

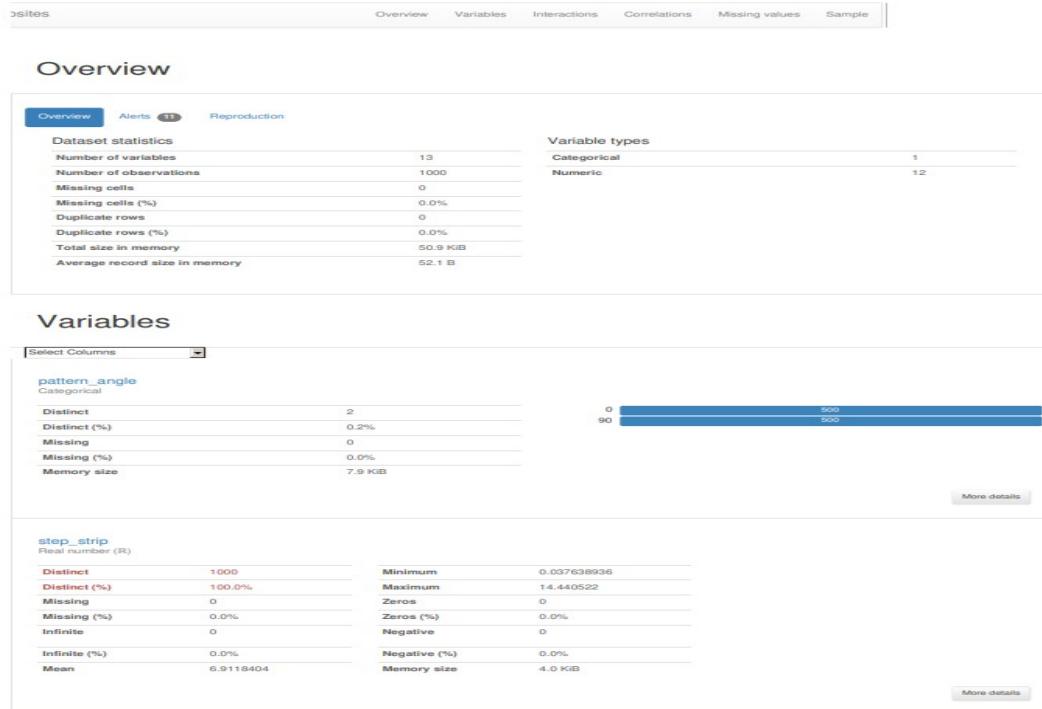


Рис. 11. Отчет по модулю ProfileReport

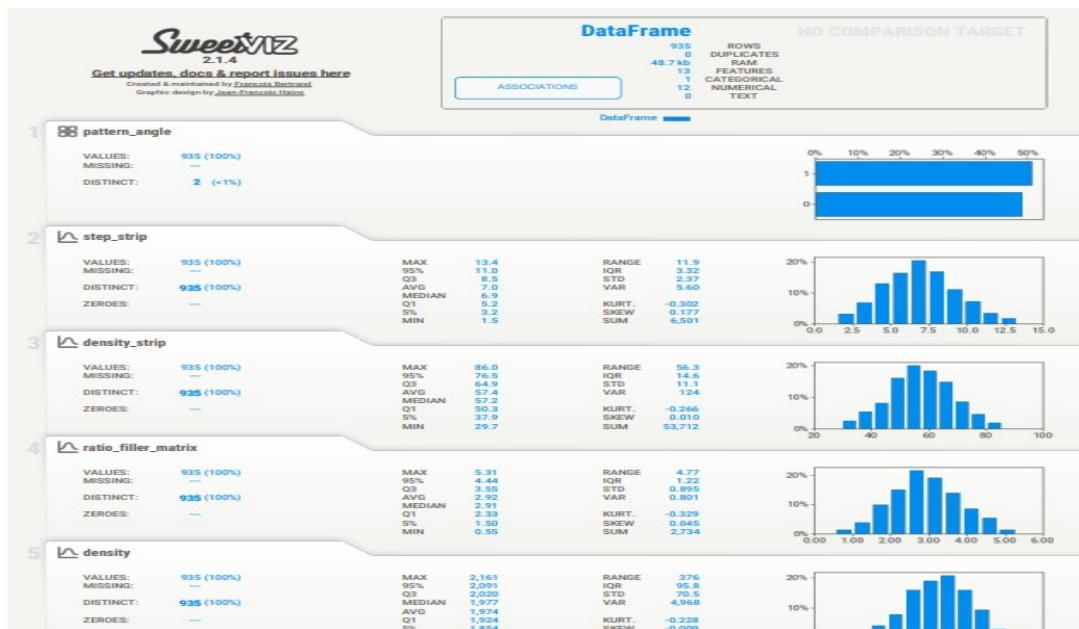


Рис. 12. Отчет по модулю SweetVIZ

Исходя из данных отчетов в датасете нас есть категориальный признак 'pattern\_angle' - Угол нашивки размерностью 0 и 90 градусов. Проведем кодирование категориального признака в новом датасете df\_le в числовой методом LabelEncoder. Благодаря LabelEncoder у нас в параметре 'pattern\_angle' вместо градусов получились значения 0 и 1 класса int.

```
+ X □ C ► Code ▾
[32]: df_le = data_main.copy()
df_le.loc[:, 'pattern_angle'] = labelencoder.fit_transform(df_le.loc[:, 'pattern_angle'])
df_le
labelencoder.classes_
```

[32]: array([ 0, 90])

Благодаря LabelEncoder у нас в параметре 'pattern\_angle' вместо градусов получились значения 0 и 1 класса int

```
[33]: df_le
```

	pattern_angle	step_strip	density_strip	ratio_filler_matrix	density	elasticity_module	number_hardene
0	0	7.856166	64.301964	2.587348	1953.274902	1136.596191	137.6274
1	0	7.401543	19.250534	2.499918	1942.595825	901.519958	146.2522
2	0	6.675780	78.623299	2.046471	2037.631836	707.570862	101.6172
3	0	7.526398	38.176975	1.856476	2018.220337	836.294373	135.4017
4	0	8.325699	46.045429	3.305535	1917.907471	478.286255	105.7869
...	...	...	...	...	...	...	...
995	1	8.088111	47.759178	2.271346	1952.087891	912.855530	86.9921
996	1	7.619138	66.931931	3.444022	2050.089111	444.732635	145.9819
997	1	9.800925	72.858284	3.280604	1972.372925	416.836517	110.5334
998	1	10.079859	65.519478	3.705351	2066.799805	741.475525	141.3979
999	1	9.021043	66.920143	3.808020	1890.413452	417.316223	129.1834

1000 rows × 13 columns

Рис. 13 Замена категориального признака 'pattern\_angle' на числовой

### 2.1.3. Описательная статистика

Описательная статистика — это краткие информационные коэффициенты, которые суммируют данный набор данных, который может быть либо представлением всей совокупности, либо выборкой генеральной совокупности. Описательная статистика подразделяется на показатели центральной тенденции и показатели изменчивости (spread). Показатели центральной тенденции включают среднее значение, медиану и моду, в то время как показатели изменчивости включают стандартное отклонение, дисперсию, минимальные и максимальные переменные.

Выведем описательную таблицу метода `describe()` нашего датасета `df_le` в распорированном виде (.T) и с округлением до 2 знаков после запятой `.round(2)`

	<code>count</code>	<code>mean</code>	<code>std</code>	<code>min</code>	<code>25%</code>	<code>50%</code>	<code>75%</code>	<code>max</code>
<code>pattern_angle</code>	1000.0	0.50	0.50	0.00	0.00	0.50	1.00	1.00
<code>step_strip</code>	1000.0	6.91	2.56	0.04	5.14	6.91	8.57	14.44
<code>density_strip</code>	1000.0	57.25	12.34	11.74	49.97	57.41	65.11	103.99
<code>ratio_filler_matrix</code>	1000.0	2.93	0.91	0.39	2.32	2.91	3.55	5.59
<code>density</code>	1000.0	1975.67	73.80	1731.76	1924.20	1977.57	2021.16	2207.77
<code>elasticity_module</code>	1000.0	739.95	330.33	2.44	498.44	741.15	962.85	1911.54
<code>number_hardeners</code>	1000.0	110.54	28.30	17.74	92.17	110.16	130.31	198.95
<code>content_epoxy_groups</code>	1000.0	22.24	2.41	14.25	20.56	22.23	23.98	28.96
<code>flash_temperature</code>	1000.0	285.91	40.96	160.26	258.54	285.85	313.58	413.27
<code>surface_density</code>	1000.0	483.02	280.81	0.60	268.06	452.97	694.21	1399.54
<code>elasticity_module_stretching</code>	1000.0	73.33	3.12	64.05	71.30	73.25	75.38	82.68
<code>strapery_strength</code>	1000.0	2467.18	485.62	1036.86	2143.83	2461.25	2760.16	3848.44
<code>resin_consumption</code>	1000.0	218.39	59.82	33.80	179.19	217.28	257.50	414.59

Рис. 14 Таблица описательной статистики

Проверим отсутствие: пропущенных и нулевых значений, а также дубликатов строк.

Функция `isnull()` используется для проверки того, есть ли в структуре данных какие-либо пропущенные значения, а метод `.sum()` по умолчанию суммирует эти True или единицы по столбцам (`axis = 0`).

```
df_le.isnull().sum()
[37]: pattern_angle          0
       step_strip            0
       density_strip          0
       ratio_filler_matrix    0
       density                0
       elasticity_module      0
       number_hardeners       0
       content_epoxy_groups   0
       flash_temperature       0
       surface_density         0
       elasticity_module_stretching 0
       strapery_strength       0
       resin_consumption       0
       dtype: int64
```

Рис. 15 Проверка на отсутствие нулевых значений

Пропущенных значений нет!

Проверка того, есть ли в структуре данных какие-либо отсутствующих значений (NaN). Функция `isna()` в Pandas используется для обнаружения отсутствующих значений (NaN), значения NaN в структуре данных сопоставляются с `True`, а значения, отличные от NaN, сопоставляются с `False`.

```
df_le.isna().sum()

[38]: pattern_angle          0
step_strip                  0
density_strip                0
ratio_filler_matrix          0
density                      0
elasticity_module            0
number_hardeners             0
content_epoxy_groups         0
flash_temperature             0
surface_density               0
elasticity_module_stretching 0
strapery_strength             0
resin_consumption             0
dtype: int64
```

Рис. 16 Проверка на отсутствующие значения

Еще один метод определения пропущенных значений - Тепловая карта пропущенных значений `sns.heatmap`. Определяем цвета: желтый - непропущенные данные, синий – пропущенные.

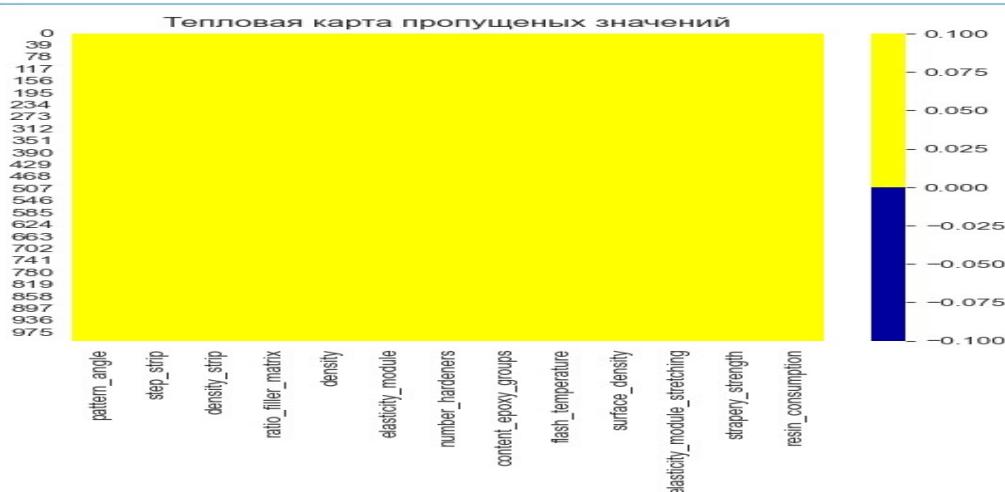


Рис. 17 Тепловая карта пропущенных значений

Проверим датасет на наличие дубликатов строк в данных

```
[40]: # Проверим датасет на наличие дубликатов строк в данных
df_le.duplicated().sum()
```

[40]: 0

Рис. 18 Проверка на отсутствие дубликатов строк

В нашем датасете df\_le НЕТ дубликатов.

Проведем подсчет уникальных значений по каждому признаку.

```
[41]: # подсчет уникальных значений по каждому признаку
df_le.apply(lambda x: x.unique())
```

```
[41]: pattern_angle           2
step_strip                 1000
density_strip              1000
ratio_filler_matrix        1000
density                     1000
elasticity_module          1000
number_hardeners           1000
content_epoxy_groups       1000
flash_temperature           999
surface_density             1000
elasticity_module_stretching 1000
strapery_strength           1000
resin_consumption          999
dtype: int64
```

Рис. 19 Проверка на наличие уникальных значений

Получим среднее и медианное значения данных в столбцах

```
mean_and_50 = df_le.describe()
mean_and_50.loc[['mean', '50%']]
```

	pattern_angle	step_strip	density_strip	ratio_filler_matrix	density	elasticity_module	number_hardeners
<b>mean</b>	0.5	6.911840	57.245335	2.930612	1975.666748	739.950562	110.541115
<b>50%</b>	0.5	6.913444	57.413593	2.907832	1977.574341	741.148102	110.162666

Рис. 20 Средние и медианные значения

Среднее и медианное значения по всем признакам близки друг к другу, что говорит о близости к нормальному распределению значений в параметрах.

#### 2.1.4. Графическая часть статистического анализа

Построим гистограммы наших 13 параметров датасет df\_le до очистки от выбросов с помощью созданной функции по отрисовке графиков разных типов drawing\_graphs из папки modules\_def.

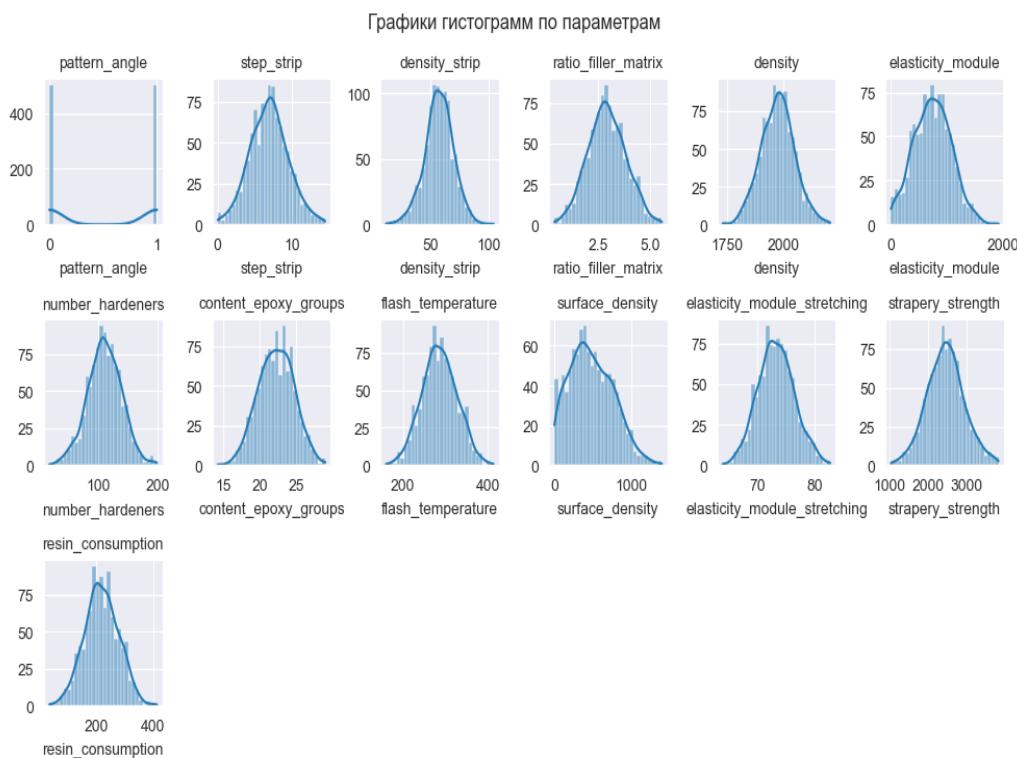


Рис. 21 Гистограммы параметров датасет df\_le до очистки от выбросов

Одним из самых удобных способов определить тип распределения вероятностей является визуализация данных с помощью гистограмм. По оси X отображаются значения в наборе данных, а по оси Y — частота каждого значения. В зависимости от значений в наборе данных гистограмма может принимать различные формы. Распространенной формой является кривая в форме колокола, известная как «нормальное распределение»

Кроме параметра 'pattern\_angle' / "Угол нашивки" и 'surface\_density' / "Поверхностная плотность" остальные переменные в основном соответствуют нормальному распределению.

Данные гистограммы параметра 'surface\_density' - 'Поверхностная плотность' показывают четко выраженную асимметрию. Измерение асимметрии используется для определения того, являются ли данные симметричными или искаженными. Если индекс находится между -1 и 1, то распределение является симметричным. Если индекс не больше -1, то он смещен влево, а если он равен как минимум 1, то он смещен вправо. Большое количество значений параметра имеют показатели близкие к 0, что также указывает на предобработку данных. Хвост гистограммы с правой стороны намного длиннее, чем с левой, и поэтому мы говорим, что асимметрия данного параметра - положительная. Мы можем оценить асимметрию данных количественно при помощи функции библиотеки pandas skew.

Асимметрия в параметре surface\_density может быть эффективным образом смягчена путем взятия логарифма веса при помощи функции библиотеки numpy - np.log. Уберем асимметрию в 'surface\_density' с помощью (np.log) предварительно создав новый df: df\_log\_sd. Выведем гистограммы 'surface\_density' - ДО логарифмирования df\_le и ПОСЛЕ df\_log\_sd.

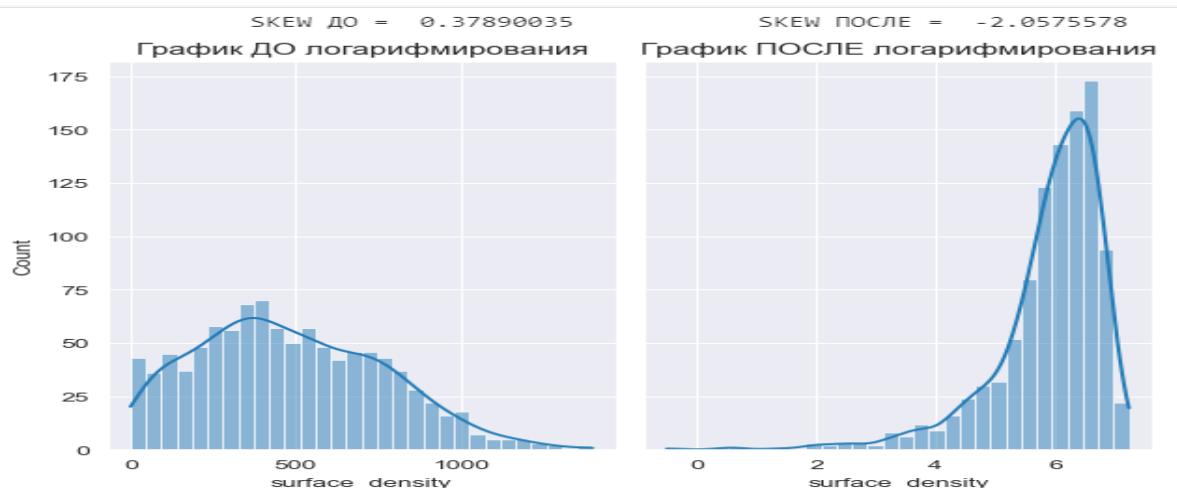


Рис. 22 Гистограммы параметра и изменение SKEW ДО и ПОСЛЕ np.log  
 Гистограмма 'surface\_density' сместилась вправо и стала отрицательной, но мы ушли от значений близких к 0!

Проверим на точечном графике (Рис. 20) как повлияло логарифмирование параметра 'surface\_density' на взаимоотношение с одним из целевых параметров ВКР 'strapery\_strength'. К сожалению, это не привело к выявлению какой-либо регрессионной зависимости.



Рис. 23 Точечные диаграммы параметра ДО и ПОСЛЕ np.log

Рассмотрим графическое распределение значений между параметрами. Построим матричный график зависимости параметров друг от друга.

Создадим новый датасет df\_main для дальнейшего графического анализа копируя и используя df\_le.

Посмотрим, как ведут себя параметры в разных концах массива - вызываем созданную функцию построения матричных графиков sns\_plt\_PairGrid из модуля modules\_def.

Заметим, что в соответствии с Отчетом ProfilesReport, у нас в датасете параметр 'pattern\_angle' имеет 500 первых значений равное 0 и 500 последних

значений равное 1. Построим матричные графики зависимости параметров друг от друга для первых 500 значений, и для последних 500 значений.

Можно в таком типе диаграмм создать сразу 3 типа графиков:

- НАД главной диагональю: scatterplot;
- НА главной диагонали: histplot;
- ПОД главной диагональю: kdeplot

Графики для head (500) и tail (500) очень похожи и представлены одним рисунком Рис. 24

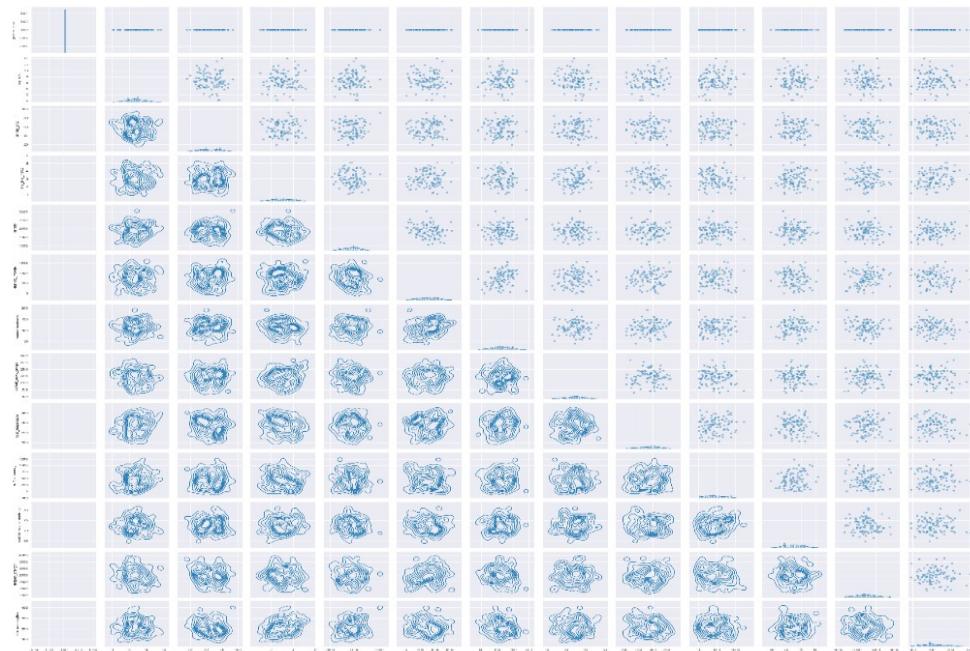


Рис. 24 Матричная диаграмма 50% датасет

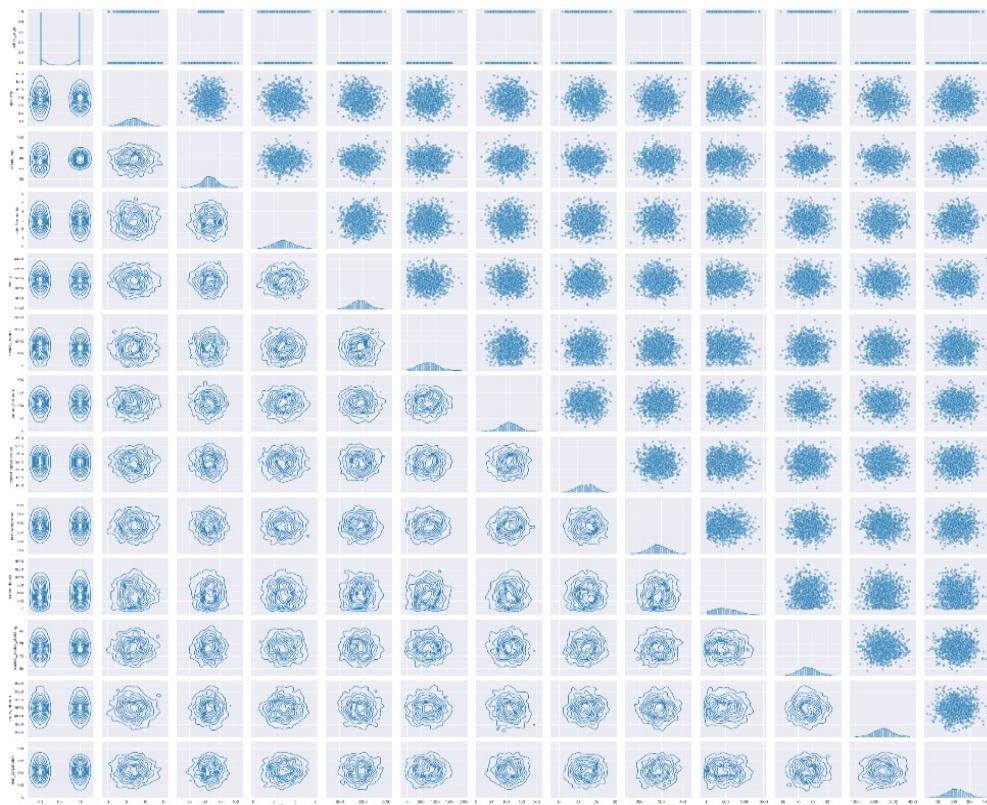


Рис. 25 Матричная диаграмма ПОЛНОГО датасет

Как видно из точечных графиков для всего массива - у нас не наблюдается никакой корреляции между параметрами. Однако для начальных данных, что в голове head, что в конце массива tail (для разных значений Угла нашивки) ВОЗМОЖНО есть слабая корреляция. Проверим это в других Блоках нашего исследования.



### 2.1.5. Глубокие исследования числовых

Создадим датасет df\_min\_max который содержит ЭКСПЕРТНЫЕ мин и макс значения по каждому столбу в качестве критерия отбрасывания строк, где min - индекс строки 0 и max- индекс строки 1 получены оценочным путем при исследовании гистограмм по каждому параметру-столбцу

```
df_min_max = pd.DataFrame({  
    'pattern_angle': [0, 1],  
    'step_strip': [1.5, 13.5],  
    'density_strip': [29.0, 85.6],  
    'ratio_filler_matrix': [0.5, 6.0],  
    'density': [1780.0, 2159.0],  
    'elasticity_module': [50.0, 1560.0],  
    'number_hardeners': [30.0, 190.0],  
    'content_epoxy_groups': [16.0, 28.0],  
    'flash_temperature': [180.0, 394.0],  
    'surface_density': [28.5, 1200.0],  
    'elasticity_module_stretching': [66.0, 81.0],  
    'strapery_strength': [1326.0, 3600.0],  
    'resin_consumption': [80.0, 350.0],  
})
```

Рис. 26 ЭКСПЕРТНЫЕ мин и макс значения по параметрам

Анализ данных описательной статистики и гистограмм параметров показал, что практически у всех параметров есть минимальные значения СЛЕВА, которых достаточно много и простое их удаление приведет к обеднению датасета.

Применим созданную функция очистки выбросов – ‘подозрительных значений СЛЕВА’ clean\_df\_left из папки modules\_def. Функция заменяет подозрительные значения на экспертные минимальные значения. На Рис. 27 приведена часть вывода работы этой функции.

```
[73]: df_main = clean_df_left(df_main, df_min_max)  
  
столбец : pattern_angle  
Количество подозрительных строк в столбце = 0  
-----  
столбец : step_strip  
Количество подозрительных строк в столбце = 17  
-----  
столбец : density_strip  
Количество подозрительных строк в столбце = 20  
-----  
столбец : ratio_filler_matrix  
Количество подозрительных строк в столбце = 2  
-----  
столбец : density  
Количество подозрительных строк в столбце = 2  
-----  
столбец : elasticity_module  
Количество подозрительных строк в столбце = 14
```

Рис. 27 Результат замены Экспертными значениями min СЛЕВА

Всего замена подозрительных значений слева произведена в 111 строках.

Построим опять графики 'гистограммы' и посмотрим, как себя ведут параметры после подстановки минимальных значений на Экспертные мин.

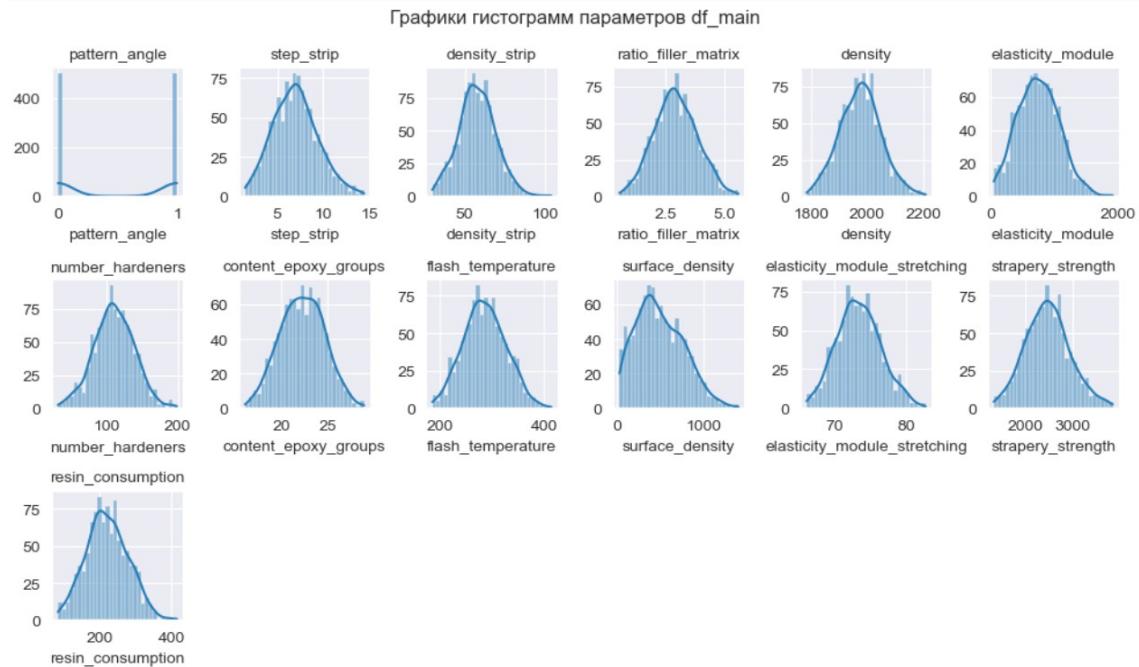


Рис. 28 Гистограммы после подстановки значений мин СЛЕВА

Графики стали более "гаусовскими")

Построим графики 'ящиков с усами' boxplot с 'выбросами' после подстановки и неудаления подозрительных значений.

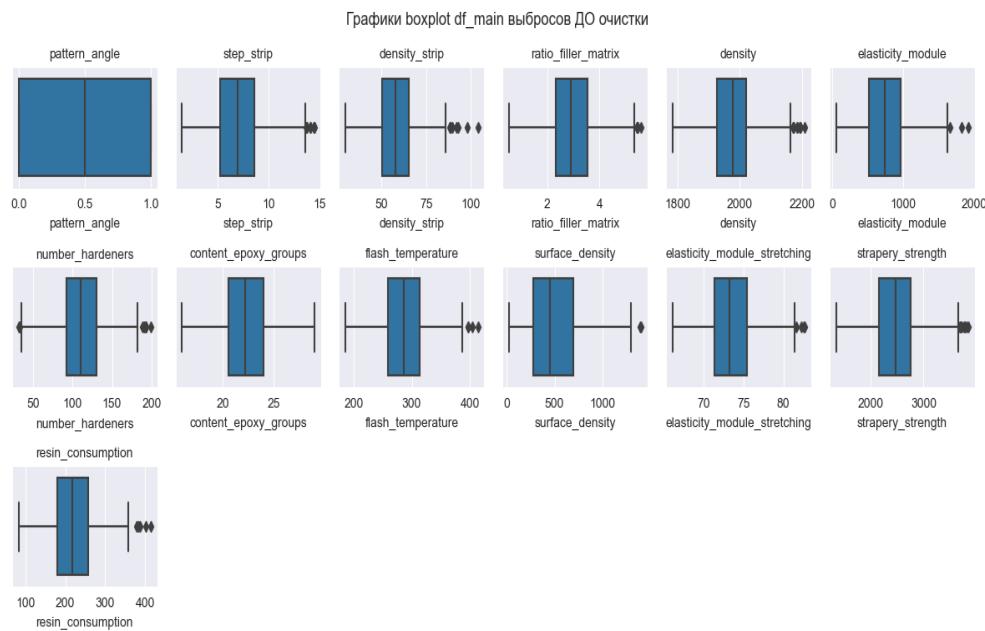


Рис. 29 Графики boxplot ДО очистки датасета

Выбросы есть и их очень много. Проведем очистку нашего df\_main (в котором 1000 строк) от выбросов.

### 1. Способ удаления выбросов IQR

Функцией outliers\_IQR\_delit поиска выбросов в интервале Q1 - Q3 для каждого параметра, для числовых признаков с нормальным распределением используется метод межквартильного диапазона. Формула поиска выбросов по межквартильному диапазону:  $IQR = Q3 - Q1$ ,

где  $Q3$  – третий квартиль (или 75 процентиль);

$Q1$  – первый квартиль (или 25 процентиль);

Уравнения для расчета низких или высоких выбросов с помощью диапазона IQR:

Высокий выброс - значение, превышающее  $Q3 + (1,5 \times IQR)$ ;

Низкий выброс - значение, находящееся ниже границы  $Q1 - (1,5 \times IQR)$ .

```
[103]: # Функция outliers_IQR_delit поиска выбросов в интервале Q1, Q3, и interquartile для каждого параметра
# передаваемые параметры:
# df- исследуемый df
# Возвращаем новый очищенный от выбросов df: data_clean
def outliers_IQR_delit(df):
    Q1 = df.quantile(q=.25)
    Q3 = df.quantile(q=.75)
    IQR = df.apply(stats.iqr) # from scipy import stats
    # сохраняют только строки которые имеют значения в интервале 1.5*IQR от Q1 и Q3
    data_clean = df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR))).any(axis=1)]
    # Выведем на печать как много строк у нас осталось после очистки методом IQR
    data_clean.shape
    print('Количество строк до удаления: ', len(df))
    print('Количество строк после удаления: ', len(data_clean))
    print('Удалено строк :', (len(df) - len(data_clean)))

    return(data_clean)
```

Рис. 30 Код функции outliers\_IQR\_delit

Запишем очищенный массив df\_main в датасет - data\_clean. Результат работы функции outliers\_IQR\_delit:

У нас было 1000 строки до применения метода IQR. Теперь мы получили после очистки 945 строк в data\_clean. Удалили 55 строк.

2. Способ удаления выбросов Z\_score, посмотрим сколько строк уберет этот метод из массива data\_clean. Вызываем функцию очистки выбросов по методу Z\_score.

```
# Функция Z_score для вычисления Z оценки и удаления выбросов методом Z_score
def Z_score(data):
    global outliers,zscore
    outliers = []
    zscore = []
    upper_threshold = 3
    lower_threshold = -3
    mean = np.mean(data)
    std = np.std(data)
    for i in data:
        z_score=(i - mean)/std
        zscore.append(z_score)
        if np.abs(z_score) > upper_threshold or np.abs(z_score) < lower_threshold:
            outliers.append(i)
    return print("Total number of outliers are",len(outliers)) #outliers,, zscore
```

Рис. 31 Код функции Z\_score

Данный метод Z - Score НЕ нашел возможных для удаления строк!

Опять построим графики "ящики с усами" и посмотрим есть ли выбросы.

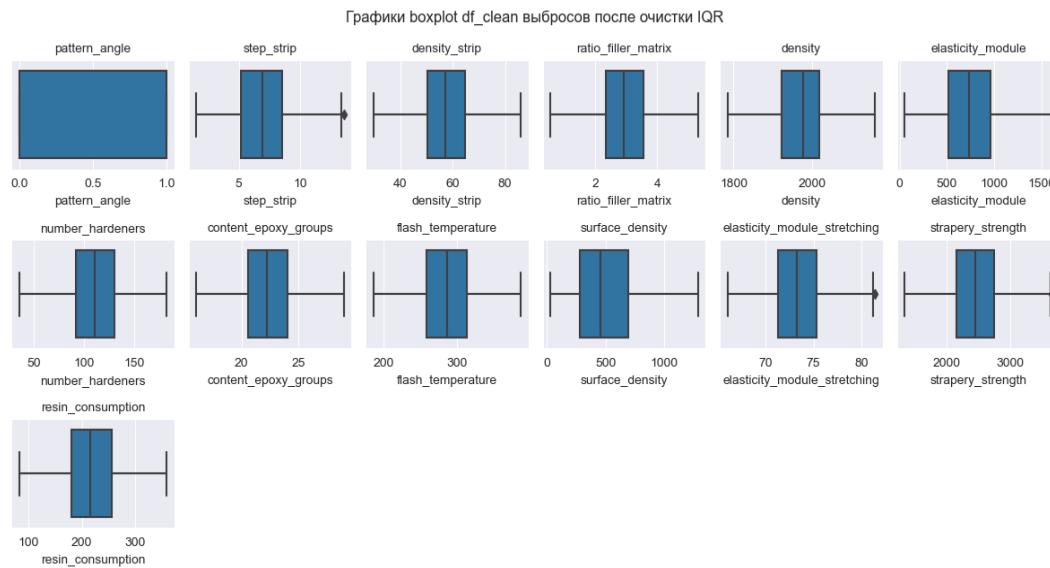


Рис.32 Графики boxplot после очистки методами IQR и Z-score

Функция `mean_round_point` расчета СРЕДНЕГО значения для "подозрительного элемента" в столбце с разбросом как ВЫШЕ, так и НИЖЕ от "подозрительного элемента"

У нас только в 3-х столбцах наблюдаются выбросы после замены 'подозрительных значений' функцией '`mean_round_point`' на средние значения +/- 5 значений в столбце и очистки стандартными методами IQR и Z-C.

Проведем следующую итерацию - дополнительную очистку в столбцах: '`step_strip`', '`elasticity_module_stretching`' и '`strapery_strength`'. Используем функцию `mean_round_point` и метод подбора коэффициентов в '`df_min_max`' и удалением этих выбросов в датасете `data_clean`.

(944, 13)

Количество подозрительных строк в столбце = 1

(941, 13)

Количество подозрительных строк в столбце = 3

(935, 13)

Количество подозрительных строк в столбце = 6

(935, 13)

После 3-й итерации по очистке данных проверим наличие выбросов

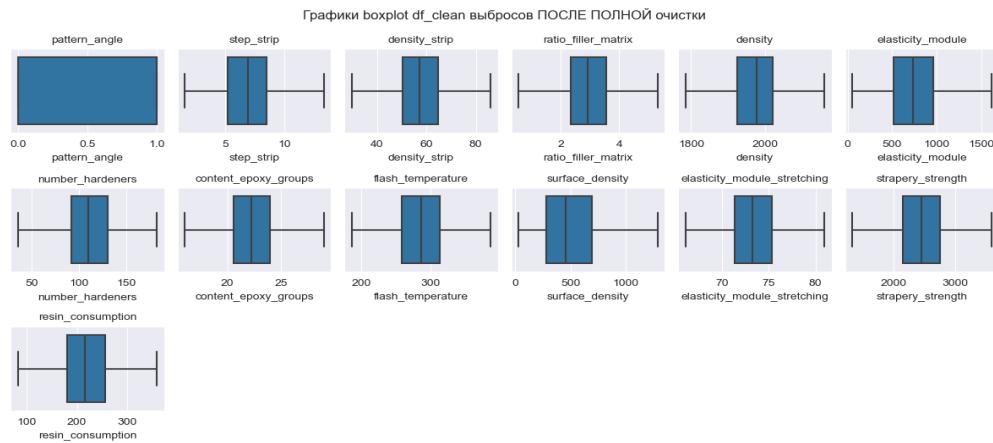


Рис. 33 Графики boxplot после очистки СПРАВА

У нас получились "чистые", без выбросов "ящики с усами"!

Построим графики зависимости переменных друг от друга с выделением категориальной переменной 'pattern\_angle', возможно параметры зависят от нашего категориального параметра.

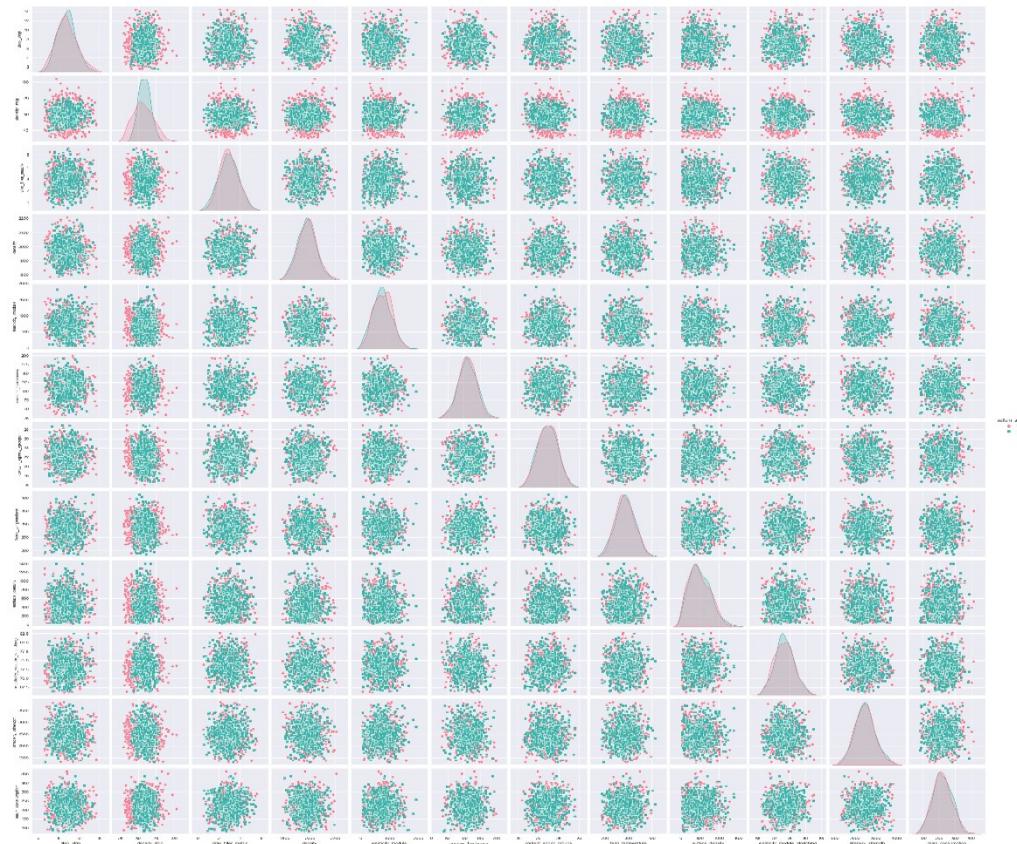


Рис. 34 Графики зависимости переменных друг от друга

На графиках видно, что имеется некое разделение на 2 класса, хотя никакой зависимости впрямую не наблюдается.

Всего общее количество удалённых подозрительных строк из-за выбросов СПРАВА на данном этапе = 10. У нас осталось  $945 - 10 = 935$  строк.

Количество строк до очистки: 1000

Количество строк после очистки: 935

Удалено строк: 65

Удалено: 6.5 %

У нас получился датасет ‘data\_clean’ со следующими параметрами:

```
[91]: data_clean.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 935 entries, 0 to 999
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   pattern_angle    935 non-null    int32  
 1   step_strip       935 non-null    float32 
 2   density_strip    935 non-null    float32 
 3   ratio_filler_matrix 935 non-null    float32 
 4   density          935 non-null    float32 
 5   elasticity_module 935 non-null    float32 
 6   number_hardeners 935 non-null    float32 
 7   content_epoxy_groups 935 non-null    float32 
 8   flash_temperature 935 non-null    float32 
 9   surface_density   935 non-null    float32 
 10  elasticity_module_stretching 935 non-null    float32 
 11  strapery_strength 935 non-null    float32 
 12  resin_consumption 935 non-null    float32 
dtypes: float32(12), int32(1)
memory usage: 54.8 KB
```

Рис. 35 Параметры чистого датасета ‘data\_clean’

Считаем, что датасет ‘data\_clean’ очищен от выбросов и с ним можно проводить исследования на регрессию.

Запишем ‘data\_clean’ в новый файл 'data\_main\_clean.csv' для дальнейшей работы и изучения методами sklearn.

### 2.1.6. Выводы по Блоку № 1

С массивом с конечным названием data\_clean мы провели следующие исследования:¶

В df\_nup в диапазоне от 0 до 40 строки наблюдается явная детерминированность показателей, что свидетельствует об искусственном искажении DF. Удалим первые 40 строк из df\_nup как явно искаженные, оставив  $1040 - 40 = 1000$  строк.

В df\_bp в диапазоне от 0 до 23 строк наблюдается явная детерминированность показателей, что также свидетельствует об искусственном искажении df\_bp. В df\_bp мы отбросили начальные 23 строки, получили  $1023 - 23 = 1000$ . Размерность наших массивов будет одинакова и их можно было объединять.

Объединили df\_nup.drop и df\_bp-drop методом INNER в один датафрейм.

Все признаки являются числовыми.

Исходный dataset не содержит пропущенных данных и дубликатов строк.

Все признаки за исключением "Угол нашивки, град" имеют много уникальных данных.

Проверили возможность логарифмирования параметра, имеющего искаженную гистограмму. Это не дало результатов.

Провели большой графический анализ матричными графиками на первоначальных, конечных, средних и полном объемах данных. Видимых зависимостей на графиках не обнаружено.

Все признаки за исключением "Угол нашивки, град" имеют выбросы.

Провели замену подозрительных значений слева в 111 строках.

Провели очистку выбросов методом IQR и удалили 55 строк.

Провели очистку выбросов методом Z\_Score. Метод не нашел дополнительные выбросы.

Провели графический анализ по графикам 'boxplot', мы убедились, что в ряде столбцов остались выбросы. Чтобы убрать оставшиеся выбросы, мы в ручном режиме на основе экспертных оценок пороговых значений для каждого параметра убрали по шаблону 'df\_min\_max' подозрительных значений СЛЕВА и СПРАВА меньше МИН и больше МАХ, еще 10 строк.

Всего мы удалили  $1000 - 935 = 65$  строк или 6.5 % из исходного файла.

Считаем, что df - data\_clean очищен от выбросов и с ним можно проводить исследования на регрессию.

Запишем 'data\_clean' в новый файл 'data\_main\_clean.csv' для дальнейшей работы и изучения методами sklearn.

Код по данному блоку Diploma\_block1\_clean.ipynb.

## **2.2. Блок № 2 Корреляционный анализ данных**

### **2.2.1. Исходные данные для Блока № 2**

В данном Блоке № 2 мы проведем всесторонний корреляционный анализ нашего датасета data\_main\_clean, полученного после очистки от выбросов в Блоке № 1.

```
# Блок 2 Корреляционные зависимости
```

```
# Импортируем в проект требуемые библиотеки
# Блок основных библиотек
import pandas as pd # библиотека для обработки и анализа данных
import numpy as np
import seaborn as sns # библиотека для создания статистических графиков
import random # генераторатор случайных чисел и данных
import os # библиотека функций для работы с операционной системой.
import math

# Блок модулей и библиотек для построения графиков
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
import plotly
from plotly.subplots import make_subplots
from scipy.interpolate import make_interp_spline
from scipy.interpolate import interp1d
%matplotlib inline
sns.set_style('darkgrid')

# Блок библиотек и модулей для статистического и корреляционного анализа
from sklearn.preprocessing import MinMaxScaler
import scipy
from scipy import stats
from scipy.stats import mstats

import ydata_profiling # Модуль статистического отчета. Форма pandas_prof
from ydata_profiling import ProfileReport
import sweetviz as sv # Модуль статистического отчета sweetviz

# Блок исключения вывода предупреждающих ошибок
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=UserWarning)
```

Рис.36 Библиотеки и модули Python для Блока №2

Подключаемые в данном Блоке №1 функции, которые разработаны для всего проекта и которые находятся в папке modules\_def.

```
# Блок импорта созданных функций для проекта из папки modules_def
%run ./modules_def/optimize_memory_usage.ipynb # функция оптимизации размера df
%run ./modules_def/drawing_graphs.ipynb          # функция рисования разных графиков
%run ./modules_def/sns_plt_PairGrid.ipynb         # функция рисования матрицы графиков
%run ./modules_def/plot_corr_heatmap.ipynb        # функция рисования корреляционной матрицы
```

Рис. 37 Подключаемые функции для Блока №2

### 2.2.2. Общая корреляционная матрица

Загрузили из папки ‘data\_storage/data\_block1\_clean’ датасет ‘data\_main\_clean’ с параметрами.

```
[7]: data_main_clean.apply(lambda x: x.unique())
[7]: pattern_angle           2
      step_strip            935
      density_strip          935
      ratio_filler_matrix    935
      density                935
      elasticity_module      935
      number_hardeners       935
      content_epoxy_groups   935
      flash_temperature       934
      surface_density          935
      elasticity_module_stretching 935
      strapery_strength       935
      resin_consumption       935
      dtype: int64
```

Рис. 39 Количество уникальных значений

В нашем датасете 13 параметров по 935 уникальных строк в каждом.

Получили отчеты с помощью модулей ProfileReport и SweetVIZ по очищеному от выбросов датасету в форматах html и PDF, приведены в папке profile\_report и в Приложении Б.

Построим корреляционную матрицу датасета data\_main\_clean с помощью функции plot\_corr\_heatmap, эта функция построения корреляционно матрицы и записи результатов в директорию.

Корреляционная матрица очищенного DataSet data\_main\_clean



Рис.40 Корреляционная матрица очищенного от выбросов датасета

Рассчитаем максимальные коэффициенты всего датасета и первых 100 строк.

```
[7]: # Максимальные корреляции в нашем df для полного объема строк
print(data_main_clean.corr().abs().apply(lambda x: sorted(x)[-2]))
```

	pattern_angle	step_strip	density_strip	ratio_matrix	density	elasticity_module	number_hardeners	ent_epoxy_groups	flash_temperature	surface_density	elasticity_module_stretching	strapery_strength	resin_consumption
pattern_angle	0.107487												
step_strip	0.060461	0.107487											
density_strip		0.107487	0.078895										
ratio_matrix			0.078895	0.080297									
density				0.080297	0.056466								
elasticity_module					0.056466	0.084787							
number_hardeners						0.084787	0.066516						
ent_epoxy_groups							0.066516	0.065471					
flash_temperature								0.065471	0.060695				
surface_density									0.060695	0.084787			
elasticity_module_stretching										0.084787	0.080297		
strapery_strength											0.080297	0.078895	
resin_consumption													0.078895
dtype:	float64												

Рис. 41 MAX коэффициенты корреляции для ПОЛНОГО датасета

Как видно из графика корреляционной матрицы и приведенного снимка экрана, корреляционные коэффициенты всего массива крайне малы, что

свидетельствует о слабой взаимосвязи между параметрами датасета. Однако при выведении максимальных значений корреляции для первых 100 значений, максимальные коэффициенты значительно выросли - от более чем в два раза до 8 раз, как для параметров elasticity\_module и number\_hardeners.

```
[8]: # Максимальные корреляции в нашем df для первых 100 строк
print(data_main_clean.head(100).corr().abs().apply(lambda x: sorted(x)[-2]))

pattern_angle           NaN
step_strip              0.181442
density_strip           0.171892
ratio_filler_matrix     0.293409
density                 0.170690
elasticity_module       0.365598
number_hardeners        0.365598
content_epoxy_groups    0.157392
flash_temperature        0.158004
surface_density          0.276080
elasticity_module_stretching 0.161475
strapery_strength        0.276080
resin_consumption        0.293409
dtype: float64
```

Рис. 41 MAX коэффициенты корреляции датасета из 100 значений

Изучим этот феномен более внимательно. Исследуем влияние количества строк в выборке на максимальные коэффициенты корреляции.

Создадим пустой датасет df\_corr\_value для максимумов коэффициентов корреляции для разных значений количества строк выборки значений от 100 до len(data\_main\_clean).

Функция ‘value\_corr\_show’ заполняет массив ‘df\_corr\_value’ максимальными значениями для коэффициентов корреляции по каждому параметру при разных значениях N - числа выборок из нашего датасета от 100 до 935.

```
def value_corr_show(df, df_corr_value):
    for column in df_corr_value.columns:
        df_corr_value[column] = df.head(column).corr().abs().apply(lambda x: sorted(x)[-2]).round(3)
    return(df_corr_value)
```

```
: value_corr_show(data_main_clean, df_corr_value)
```

	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800	850	900	935
<b>pattern_angle</b>	NaN	0.071	0.079	0.085	0.090	0.089	0.087	0.091	0.099	0.101	0.107							
<b>step_strip</b>	0.181	0.170	0.124	0.124	0.142	0.148	0.118	0.100	0.089	0.094	0.078	0.058	0.066	0.048	0.044	0.050	0.054	0.060
<b>density_strip</b>	0.172	0.170	0.115	0.100	0.103	0.093	0.087	0.085	0.072	0.079	0.085	0.090	0.089	0.087	0.091	0.099	0.101	0.107
<b>ratio_filler_matrix</b>	0.293	0.278	0.184	0.152	0.137	0.116	0.121	0.121	0.116	0.109	0.091	0.100	0.107	0.087	0.083	0.092	0.087	0.079
<b>density</b>	0.171	0.118	0.152	0.175	0.175	0.171	0.146	0.108	0.111	0.105	0.084	0.087	0.080	0.076	0.073	0.081	0.081	0.080
<b>elasticity_module</b>	0.366	0.215	0.171	0.129	0.092	0.077	0.093	0.109	0.096	0.099	0.077	0.066	0.063	0.070	0.057	0.051	0.055	0.056
<b>number_hardeners</b>	0.366	0.215	0.171	0.173	0.151	0.129	0.107	0.096	0.084	0.087	0.089	0.082	0.078	0.079	0.075	0.086	0.091	0.085
<b>content_epoxy_groups</b>	0.157	0.088	0.078	0.116	0.103	0.104	0.082	0.085	0.063	0.058	0.064	0.067	0.062	0.055	0.058	0.061	0.069	0.067
<b>flash_temperature</b>	0.158	0.142	0.124	0.081	0.092	0.080	0.107	0.109	0.096	0.099	0.077	0.066	0.062	0.058	0.066	0.074	0.073	0.065
<b>surface_density</b>	0.276	0.243	0.207	0.175	0.175	0.171	0.146	0.108	0.111	0.105	0.090	0.087	0.068	0.070	0.056	0.062	0.061	0.061
<b>elasticity_module_stretching</b>	0.161	0.135	0.141	0.105	0.072	0.098	0.090	0.098	0.091	0.092	0.089	0.080	0.074	0.073	0.075	0.086	0.091	0.085
<b>strapery_strength</b>	0.276	0.243	0.207	0.173	0.151	0.148	0.119	0.104	0.107	0.094	0.090	0.082	0.080	0.079	0.073	0.081	0.081	0.080
<b>resin_consumption</b>	0.293	0.278	0.184	0.152	0.125	0.129	0.132	0.121	0.116	0.109	0.091	0.100	0.107	0.087	0.083	0.092	0.087	0.079

Рис. 42 Функция и датасет максимальных коэффициентов корреляции

Построим сравнительные графики изменения MAX размера корреляции от количества N строк выборки.

В датасете `df_corr_value'` есть пропущенные значения в параметре `pattern_angle`, так как в исходном датасете в этом параметре до 500 все значение 0! Заменили значения Nan на 0.082 - ближайшее известное значение.



Рис. 43 Графики изменения MAX размера корреляции от количества строк  
Рассмотрим более внимательно каждый график отдельно.



Рис. 44 Графики параметров изменения max коэффициентов корреляции  
MAX коэффициенты корреляции стремительно падают с ростом N!!!



### 2.2.3. Гипотеза по сокращению числа выборки значений

В соответствии с теорией хрупкого разрушения (теория Гриффита) [29] прочность ( $\sigma_p$ ) определяется удельной энергией ( $\alpha$ ) вновь образованной поверхности разрушения:  $(\sigma_p) = f(\alpha * E)$ , где ( $\sigma_p$ ) это наш параметр 'Прочность при растяжении' - 'strapery\_strength' (размерность в Мпа), а  $E$  это 'Модуль упругости при растяжении' - 'elasticity\_module\_stretching' (размерность в ГПа, т.е в 1000 раз больше чем  $\sigma_p$ ). Учитывая такую физическую связь между параметрами, введем один новый признак/столбец ALFA- 'удельную энергию' ( $\alpha$ ) =  $\sigma_p / E$  в наш новый датасет 'df\_add\_col'.

```
[25]: # Введем новый признак ALFA
df_Sig = data_main_clean.strapery_strength
df_E = data_main_clean.elasticity_module_stretching
df_add_col = (data_main_clean.assign(alfa = df_Sig / df_E))

[26]: df_add_col.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 935 entries, 0 to 934
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   pattern_angle    935 non-null    int32  
 1   step_strip       935 non-null    float32 
 2   density_strip    935 non-null    float32 
 3   ratio_filler_matrix 935 non-null    float32 
 4   density          935 non-null    float32 
 5   elasticity_module 935 non-null    float32 
 6   number_hardeners 935 non-null    float32 
 7   content_epoxy_groups 935 non-null    float32 
 8   flash_temperature 935 non-null    float32 
 9   surface_density   935 non-null    float32 
 10  elasticity_module_stretching 935 non-null    float32 
 11  strapery_strength 935 non-null    float32 
 12  resin_consumption 935 non-null    float32 
 13  alfa             935 non-null    float32 
dtypes: float32(13), int32(1)
memory usage: 51.3 KB
```

Рис. 45 Новый признак ALFA

Ввиду того, что максимальные коэффициенты корреляции резко падают с увеличением количества значений в выборке практически до нуля и ниже нуля выдвигаем гипотезу.

**Гипотеза:** Проверка работоспособности регрессионной модели при сокращении числа значений в выборке, как фактора наличия статистически-значимой корреляции между параметрами датасета.

Возьмем для исследования на различные виды регрессии массив из датасета df\_add\_col с новым параметром ALFA 200 СЛУЧАЙНЫХ строк, здесь еще сохраняются небольшие коэффициенты корреляции.

```
[30]: # Максимальные корреляции в нашем df для СЛУЧАЙНЫХ 200 строк
print(df_add_col.sample(n=200, random_state=42).corr().abs().apply(lambda x: sorted(x)[-2]))
```

pattern_angle	0.148352
step_strip	0.098215
density_strip	0.148352
ratio_filler_matrix	0.223873
density	0.092884
elasticity_module	0.153614
number_hardeners	0.080614
content_epoxy_groups	0.132030
flash_temperature	0.097019
surface_density	0.153614
elasticity_module_stretching	0.190468
strapery_strength	0.971485
resin_consumption	0.223873
alfa	0.971485
dtype:	float64

Рис.46 Максимальные коэффициенты корреляции из N=200 значений

Создадим в качестве массива TRAIN датасет ‘df\_Train\_lineReg’ из 200 случайных строк (random\_state=42):

```
df_Train_lineReg = df_add_col.sample(n=200, random_state=42)
```

И возьмем для массива TEST датасет ‘df\_Test\_lineReg’ из 100 СЛУЧАЙНЫХ строк (random\_state=50):

```
df_Test_lineReg = df_add_col.sample(n=100, random_state=50).
```

При этом у нас сохранится классическое разделение датасета на тестовую и тренировочную выборку с коэффициентом 0.3.

#### 2.2.4. Выбор параметров (фитч) для целевых переменных при анализе регрессионных моделей

В задании к ВКР целевыми переменными для исследования регрессионного анализа заданы 'elasticity\_module\_stretching' (префикс EMS) и 'strapery\_strength' (префикс SS). Проверим для каждой целевой переменной зави-

симости от параметров с выводением регрессионной прямой на графиках зависимости.

Построим "точечные графики" с линией регрессии типа 'regplot': для 'elasticity\_module\_stretching'- EMS и для второй целевой переменной 'strapery\_strength' – SS.

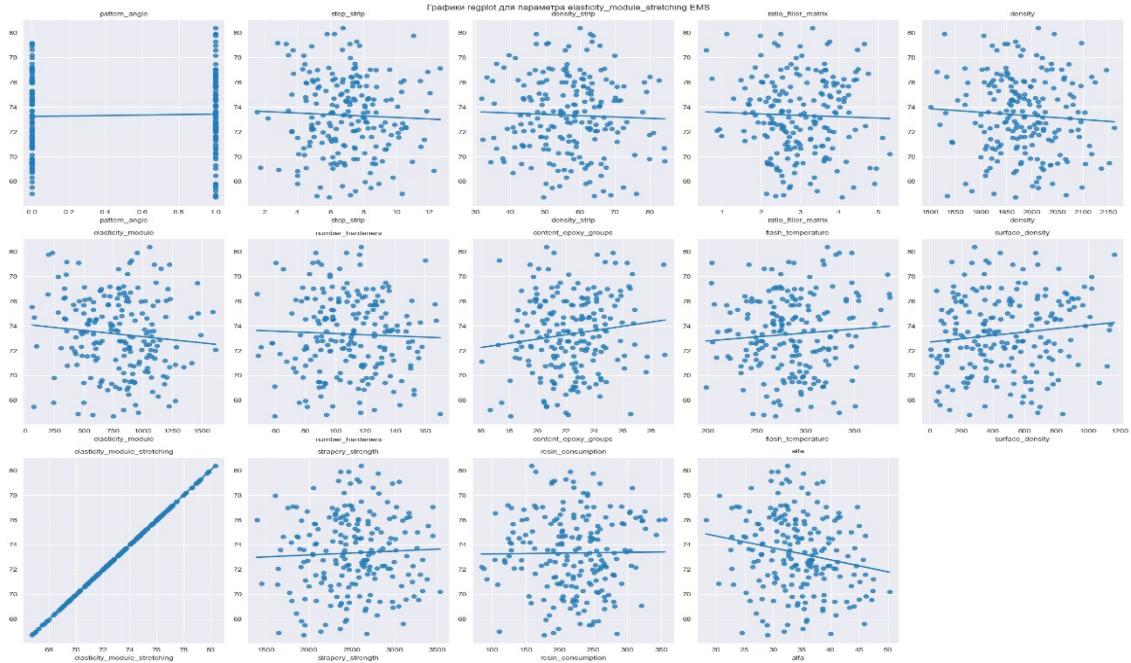


Рис.47 Графики regplot для параметра elasticity\_module\_stretching EMS'

Как видно из графиков между целевой переменной 'elasticity\_module\_stretching' (EMS):

- практически не наблюдается корреляция с параметрами: 'pattern\_angle', 'step\_strip', 'density\_strip', 'ratio\_filler\_matrix', 'density', 'flash temperature', 'surface\_density', 'resin\_consumption', 'number\_hardeners', 'strapery\_strength'
- корреляция наблюдается с параметрами: 'elasticity\_module', 'content\_epoxy\_groups', 'alfa'.

Вероятно, можно проводить регрессивный анализ по целевой переменной 'elasticity\_module\_stretching' с параметрами: 'elasticity\_module', 'content\_epoxy\_groups' и 'alfa'.

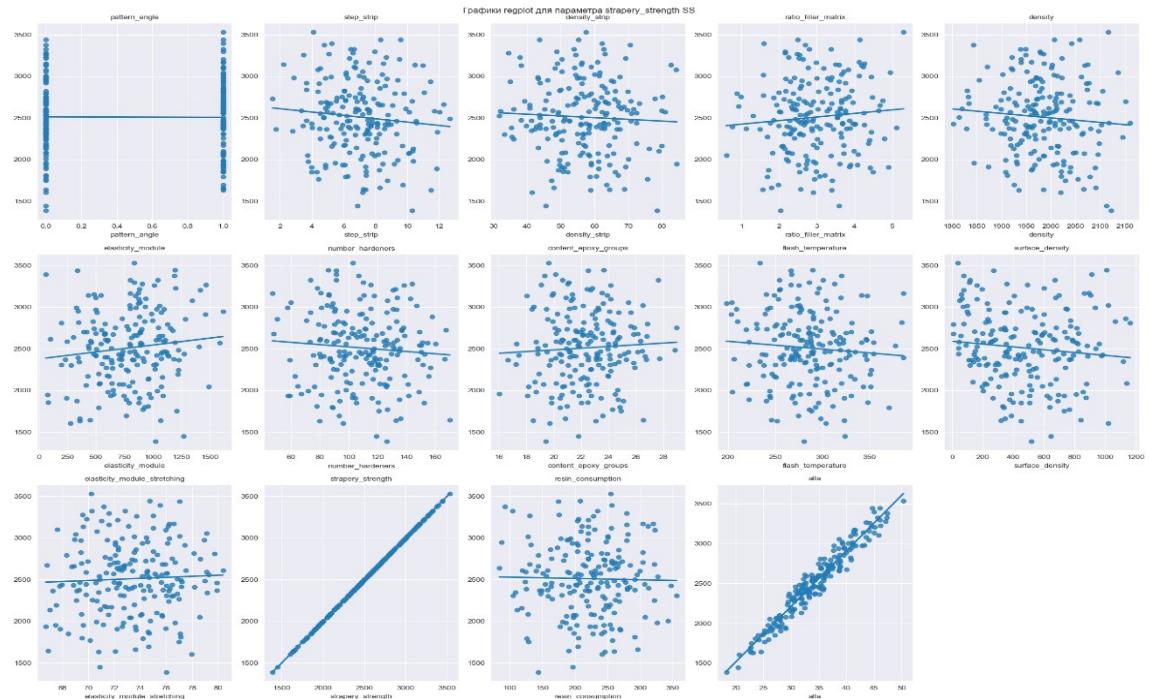


Рис. 48 Графики regplot для параметра 'strapery\_strength'-SS

Как видно из графиков для второй целевой переменной 'strapery\_strength'

а) практически не наблюдается корреляция с параметрами: 'pattern\_angle', 'step\_strip', 'density\_strip', 'density', 'content\_epoxy\_groups', 'number\_hardeners' , 'flash\_temperature', 'surface\_density', 'resin\_consumption', 'elasticity\_module\_stretching'.

б) корреляция наблюдается с параметрами: 'ratio\_filler\_matrix', 'elasticity\_module', 'alfa'.

Вероятно, можно проводить регрессивный анализ по целевой переменной 'strapery\_strength' с параметрами: 'ratio\_filler\_matrix', 'elasticity\_module' и 'alfa'.

Сокращение количества параметров для исследования ускорит вычисления в несколько раз.

Построим корреляционную матрицу для ‘df\_Train\_lineReg’ с помощью разработанной функции ‘plot\_corr\_heatmap’ из внешней директории ‘modules\_def’.

```
# Построим корреляционную матрицу data_main_clean с помощью функции plot_corr_heatmap
plot_corr_heatmap(df_Train_lineReg, 10, 6,
                   'Корреляционная матрица очищенного для df_Train_lineReg',
                   r'save_fig\fig_block2_corr\corr_heatmap_df_Train_lineReg.png'
)
```



Рис. 49 Корреляционная матрица для df\_Train\_lineReg

Запишем два файла- массив df\_Train\_lineReg (из 200 СЛУЧАЙНЫХ строк массива df\_add\_col) в ‘data\_Train\_ML’ и ‘df\_Test\_lineReg (из 100 СЛУЧАЙНЫХ строк массива df\_add\_col) в data\_Test\_ML для изучения в Блоке № 3 регрессий.

В следующей части нашей работы мы попробуем провести анализ регрессии применительно к нашему набору данных.

### 2.2.5. Выводы по Блоку № 2

Проведено исследование корреляционных матриц очищенного от выбросов и подозрительных значений датасета в зависимости от количества значений в выборке.

Сделан вывод, что при рассмотрении полного датасета корреляция отсутствует практически полностью, однако при значениях  $N = 100$  и  $200$  еще наблюдается некоторая корреляционная зависимость между параметрами, с увеличением числа выборки зависимость падает до около нулевых значений.

Проведен графический анализ с использованием регрессионной ‘regplot’ прямой по каждому параметру с целевыми переменными “Модуля упругости при растяжении” и “Прочности при растяжении”.

На основании теоретических исследований, приведенных в научной литературе по данной теме [29] введен новый параметр ALFA как функция Модуля упругости при растяжении и Прочности при растяжении.

Произведен выбор фитчей для наших целевых параметров, что позволит сократить вычислительные ресурсы и увеличить точность расчетов.

Подготовлен датасет `data_Train_ML.csv` из 200 случайных значений (`random_state=42`) в качестве обучающего массива для регрессионного анализа в оке 3 и датасет `data_Test_ML.csv` для использования в качестве тестового набора данных (`random_state=50`). Полученные датасеты записаны в соответствующую директорию `data_storage/data_block2_corr`.

Код Блока № 2 - `Diploma_block2_corr.ipynb`.

## 2.3. Блок № 3 Создание регрессионных моделей для прогноза ‘модуля упругости при растяжении’ и ‘прочности при растяжении’.

### 2.3.1. Исходные данные для работы Блока № 3

Функции, разработанные и используемые в данном Блоке.

```
[2]: # Блок импорта созданных функций для проекта из папки modules_def
%run ./modules_def/optimize_memory_usage.ipynb # функция оптимизации размера df
%run ./modules_def/drawing_graphs.ipynb      # функция рисования разных графиков
%run ./modules_def/sns_plt_PairGrid.ipynb     # функция рисования матрицы графиков
%run ./modules_def/plot_corr_heatmap.ipynb    # функция рисования корреляционной матрицы df
%run ./modules_def/analysis_core_density.ipynb # функция графического анализа плотности ядра df
%run ./modules_def/model_creation.ipynb       # функция расчета основные оценочные показатели типа регрессии
```

Рис.50 Используемые функции из другой директории

Библиотеки и модули Python, которые использовались в данном Блоке.

```
*[1]: # Блок основных библиотек
import pandas as pd
import numpy as np
import math

# Блок графических пакетов
import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
import seaborn as sns
import plotly
from plotly.subplots import make_subplots
from scipy.interpolate import make_interp_spline
from scipy.interpolate import interp1d
import pylab
%matplotlib inline
sns.set_style('darkgrid')

# Блок библиотеки sklearn
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_percentage_error # метрика для проверки
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score, cross_val_predict

from sklearn import linear_model
from sklearn.svm import SVR # метод опорных векторов
from sklearn.linear_model import LinearRegression # линейная регрессия
from sklearn.neighbors import KNeighborsRegressor # метод ближайших соседей
from sklearn.ensemble import RandomForestRegressor # случайный лес
from sklearn.neural_network import MLPRegressor # многослойный персептрон, нейросеть с учителем
import statsmodels.api as sm

# Блок исключения вывода предупреждающих ошибок
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=UserWarning)
warnings.filterwarnings("ignore")
```

Рис. 51 Библиотеки Python для работы Блока № 3

Далее проводим 2 одинаковые процедуры для двух файлов, полученных из Блока № 3: data\_Train\_ML и data\_Test\_ML. (на примере data\_Train\_ML).

1. Загружаем датасет data\_Train\_ML и смотрим его основные параметры

```
[3]: data_Train_ML = pd.read_csv(r'data_storage\data_block2_corr\data_Train_ML.csv')
```

```
[4]: data_Test_ML = pd.read_csv(r'data_storage\data_block2_corr\data_Test_ML.csv')
```

```
[5]: data_Train_ML.describe().T
```

	count	mean	std	min	25%	50%	75%	max
pattern_angle	200.0	0.505000	0.501230	0.000000	0.000000	1.000000	1.000000	1.000000
step_strip	200.0	7.026589	2.169412	1.525230	5.519591	7.031957	8.288326	12.678927
density_strip	200.0	56.812242	11.102082	31.804142	49.244229	56.861518	63.676563	84.310770
ratio_filler_matrix	200.0	2.970017	0.906174	0.596783	2.344010	2.894289	3.647081	5.295842
density	200.0	1981.942511	70.605937	1801.940700	1934.969850	1978.069700	2022.651750	2160.751500
elasticity_module	200.0	790.986054	311.236154	60.474888	570.088223	814.960845	1021.213350	1615.096900
number_hardeners	200.0	109.180384	25.754899	47.233414	90.363547	109.323345	126.774040	170.332570
content_epoxy_groups	200.0	22.307112	2.403920	16.048979	20.620342	22.363540	24.008937	28.955095
flash_temperature	200.0	285.273407	38.330166	198.172640	259.224575	285.853960	309.306460	385.894780
surface_density	200.0	475.146115	280.837422	6.779255	245.157882	446.525425	680.963520	1165.544400
elasticity_module_stretching	200.0	73.338284	3.125765	66.733850	71.098230	73.353063	75.799815	80.384926
strapery_strength	200.0	2509.748484	442.259163	1388.849400	2243.262300	2497.807600	2794.452575	3533.008500
resin_consumption	200.0	220.699298	55.954084	84.490130	188.632575	222.843100	256.854910	355.758700
alfa	200.0	34.271397	6.173258	18.272623	30.198047	34.092260	38.610857	50.312656

Рис.52 Основные параметры датасет data\_Train\_ML

### 2.3.2. Создание регрессионных моделей

Датасет ‘data\_Train\_ML’ состоит из 200 строк, параметры имеют разные размеры числовых значений. Для применения метода нормализации данных, рассмотрим графические распределения параметров.

Распределение плотности параметров data\_Test\_ML

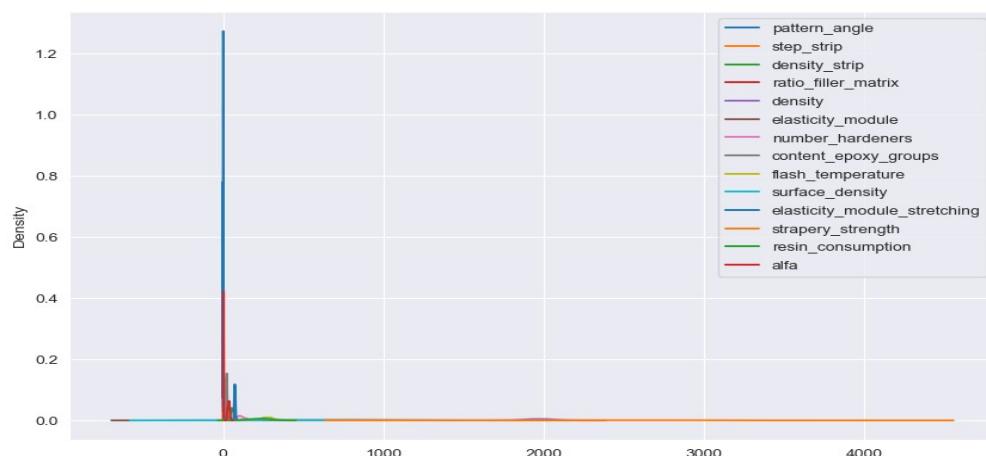


Рис.53 Распределение плотности параметров data\_Train\_ML

Графики плотности ядра показывают, что наши данные находятся в разных диапазонах, что требует нормализации данных. Нормализуем наш dataset - data\_Train\_ML и data\_Test\_ML по методу MinMaxskaler.

	count	mean	std	min	25%	50%	75%	max
<b>pattern_angle</b>	200.0	0.505000	0.501230	0.0	0.000000	1.000000	1.000000	1.0
<b>step_strip</b>	200.0	0.493232	0.194502	0.0	0.358120	0.493713	0.606355	1.0
<b>density_strip</b>	200.0	0.476285	0.211442	0.0	0.332150	0.477223	0.607017	1.0
<b>ratio_filler_matrix</b>	200.0	0.505045	0.192842	0.0	0.371825	0.488929	0.649130	1.0
<b>density</b>	200.0	0.501662	0.196778	0.0	0.370750	0.490869	0.615118	1.0
<b>elasticity_module</b>	200.0	0.469896	0.200201	0.0	0.327805	0.485318	0.617988	1.0
<b>number_hardeners</b>	200.0	0.503228	0.209221	0.0	0.350369	0.504390	0.646151	1.0
<b>content_epoxy_groups</b>	200.0	0.484897	0.186262	0.0	0.354201	0.489269	0.616759	1.0
<b>flash_temperature</b>	200.0	0.463988	0.204186	0.0	0.325225	0.467080	0.592012	1.0
<b>surface_density</b>	200.0	0.404195	0.242359	0.0	0.205718	0.379496	0.581813	1.0
<b>elasticity_module_stretching</b>	200.0	0.483803	0.228976	0.0	0.319710	0.484886	0.664121	1.0
<b>strapery_strength</b>	200.0	0.522769	0.206262	0.0	0.398484	0.517200	0.655550	1.0
<b>resin_consumption</b>	200.0	0.502119	0.206268	0.0	0.383909	0.510022	0.635403	1.0
<b>alfa</b>	200.0	0.499337	0.192673	0.0	0.372204	0.493746	0.634776	1.0

Рис. 54 Нормализованные данные data\_Train\_ML

Оценка плотности ядра data\_Train\_norm после нормализации.

Распределение плотности параметров data\_Train\_norm после нормализации

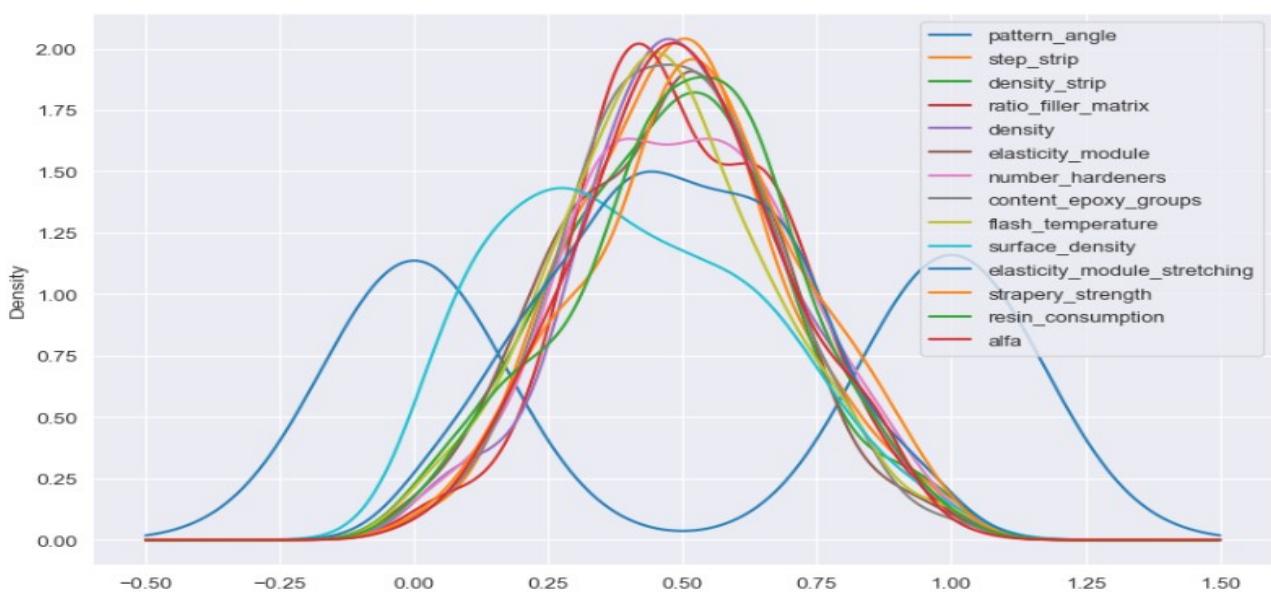


Рис. 55 Оценка плотности параметров после нормализации data\_Train\_norm.

Учитывая наши исследования в Блоке № 2 по корреляции, возьмем для исследования в Анализ регрессии только те параметры, которые показали хотя бы слабую корреляцию для наших целевых показателей: 'elasticity\_module\_stretching' и 'strapery\_strength'.

1. Собираем датасет для целевой переменной 'elasticity\_module\_stretching'.

Как мы выяснили в Блоке 2, можно проводить регрессивный анализ по целевой переменной 'elasticity\_module\_stretching' с параметрами 3 параметра: 'elasticity\_module', 'content\_epoxy\_groups' и 'alfa'.

```
# Train для EMS
data_Train_EMS = data_Train_norm.copy()
y_Train_EMS = data_Train_EMS[["elasticity_module_stretching"]]
X_Train_EMS = data_Train_EMS.drop(columns=['pattern_angle', 'step_strip', 'density_strip',
                                             'ratio_filler_matrix', 'density', 'number_hardeners',
                                             'flash_temperature', 'surface_density',
                                             'elasticity_module_stretching', 'strapery_strength', 'resin_consumption'])
X_Train_EMS.shape
(200, 3)
```

Рис. 56 Разбиение на y/X\_Train\_EMS

```
# Test - для EMS
data_Test_EMS = data_Test_norm.copy()
y_Test_EMS = data_Test_EMS[["elasticity_module_stretching"]]
X_Test_EMS = data_Test_EMS.drop(columns=['pattern_angle', 'step_strip', 'density_strip',
                                            'ratio_filler_matrix', 'density', 'number_hardeners',
                                            'flash_temperature', 'surface_density',
                                            'elasticity_module_stretching', 'strapery_strength', 'resin_consumpti
X_Test_EMS.shape
(100, 3)
```

Рис.57 Разбиение на y/X\_Test\_EMS

Проведем исследование и расчеты для наших целевых переменных следующими 6 регрессионными инструментами:

lr = LinearRegression() - линейная регрессия

knr = KNeighborsRegressor() - метод ближайших соседей

rfr = RandomForestRegressor() - случайный лес

svr = SVR() - метод опорных векторов

mlpr = MLPRegressor() - многослойный перцептрон - обучение с 'учителем'

smlr = sm.OLS() - statsmodels.regression.linear\_model.OLS() статистическая линейная регрессия - регрессия методом наименьших квадратов

В итоге сравним полученные результаты по расчетным параметрам:

R2 - коэффициент детерминации - функция оценки регрессии. Чем ближе R2 к 1 тем лучше модель описывает процесс.

MAE - mean\_absolute\_error() Средняя абсолютная ошибка. Чем ниже MAE для данной модели, тем точнее модель способна предсказать значение.

MSE - mean\_squared\_error() Среднеквадратическая ошибка

MaxER - max\_error() Максимальная ошибка

RMSE - среднеквадратическая ошибка

SCORES - оценка кросс-валидации

MCVS - Среднее значение оценки кросс-валидации

StdDS - Статистическая ошибка оценки кросс-валидации

Ввиду особенностей расчета по модели sm.OLS(), она будет рассмотрена отдельно.

Создаем датасеты для записи результатов параметров работы моделей регрессии для целевых параметров df\_error\_calc\_EMS и df\_error\_calc\_SS.

Далее инициализируем модели регрессии и переменные для расчётов.

```
# Создаем df для записи результатов параметров работы моделей регрессии для целевых параметров
df_error_calc_EMS = pd.DataFrame(columns=['Model', 'R2', 'MAE', 'MSE', 'MaxER', 'RMSE', 'MCVS', 'StdDS'])
df_error_calc_SS = pd.DataFrame(columns=['Model', 'R2', 'MAE', 'MSE', 'MaxER', 'RMSE', 'MCVS', 'StdDS'])

# Инициализируем модели регрессии.
# Модель sm.OLS() - statsmodels.regression.linear_model.OLS() будет рассмотрена ПОСЛЕ этих типов

model_lr = LinearRegression()
model_knr = KNeighborsRegressor()
model_rfr = RandomForestRegressor()
model_svr = SVR()
model_mlpr = MLPRegressor()

# Переменные для расчётов
index_err_EMS = 0 # счетчик для 'elasticity_module_stretching'
index_err_SS = 0 # счетчик для 'strapery_strength'
cv_err = 5 # количество интервалов при расчёте скросс-валидации
param_EMS = 'elasticity_module_stretching'
param_SS = 'strapery_strength'
```

Рис. 57 Инициализация регрессий

### 2.3.3 Тестирование регрессионных моделей

Проводим расчеты по моделям, записываем результатов параметров работы моделей.

```
:] : # Функция model_creation() рассчитывает основные оценочные показатели регрессии
# Передаваемые в функцию параметры:
# df - df для записи результатов параметров работы моделей для целевой переменной
# model_name - имя рассматриваемой модели
# column - название столбца по которому идет расчет в цикле
# 'target_parameter' - целевая переменная
# X_train, y_train, X_test, y_test- выборки для целевой переменной
# cv_err # целое число - количество выборок кросс-валидации (cv > 3, лучше от 5 до 10)
# index_err # счетчик для целевой переменной

def model_creation(df, model_name, column, target_param,
                    X_train, y_train, X_test, y_test,
                    cv_err, index_err, color_actual, color_pred):
    # активирует модель
    model_name.fit(X_train, y_train)
    # Оценка модели
    y_pred = model_name.predict(X_test)
    pred = np.array(y_pred)
    actual = y_test
    y_pred = model_name.predict(np.array((X_test)))
    actual = y_test.values
    predicted = y_pred
    scores = cross_val_score(estimator = model_name, X = X_test,y = y_test, cv=cv_err)
    # Стандартные параметры оценки регрессии
    # df.at [index_err, 'Target'] = target_param
    df.at [index_err, 'Model'] = column
    df.at [index_err, 'R2'] = model_name.score(X_test, y_test) # r2
    df.at [index_err, 'MAE'] = metrics.mean_absolute_error(y_test, y_pred) # MAE
    df.at [index_err, 'MSE'] = metrics.mean_squared_error(y_test, y_pred) # MSE
    df.at [index_err, 'MaxER'] = metrics.max_error(y_test, y_pred) # максимальная ошибка
    # Нестандартные параметры оценки регрессии
    df.at [index_err, 'RMSE'] = np.sqrt(np.square(np.subtract(actual, pred)).mean()) # RMSE
    # Кросс-валидационные параметры оценки регрессии
    df.at [index_err, 'MCVS'] = np.mean(scores) # MCVS
    df.at [index_err, 'StdDS'] = np.std(scores) # StdD
    # Вывод на печать Test и Predicted по каждой модели
    plt.figure(figsize=(17,5))
    plt.title(f'Тестовые и прогнозные значения: {model_name}')
    plt.plot(actual, color = color_actual, label='Test')
    plt.plot(pred, color = color_pred, label='Прогноз')
    plt.legend(loc='best')
    plt.ylabel(target_param)
    plt.xlabel('Количество наблюдений')
    return(df, index_err)
```

Рис. 58 Функция model\_creation() рассчитывает основные показатели

Рис. 59 Блок кода расчетов по моделям

Выводим на печать графики: для df\_error\_calc\_EMS цвета - зеленый и желтый, для df\_error\_calc\_SS цвета – синий и красный. Выведем только 2 пары графиков для 2-х моделей для экономии места.



Рис. 60 Линейная модель для df\_error\_calc\_EMS



Рис.61 Линейная модель для df\_error\_calc\_SS

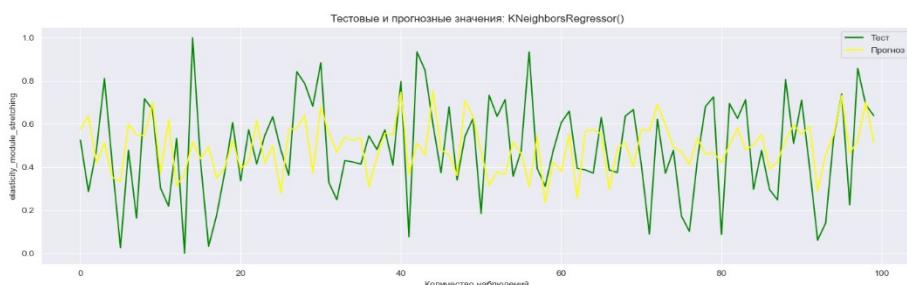


Рис.62 KNeighborsRegressor для df\_error\_calc\_EMS

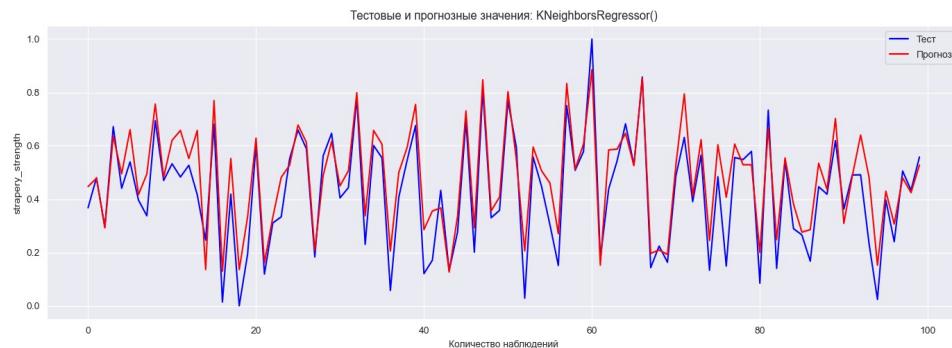


Рис.63 KNeighborsRegressor для df\_error\_calc\_SS

Записываем полученные датасеты в папку по Блоку № 3 и выводим полученные значения в таблицы.

```

: df_error_calc_EMS.to_csv(r'data_storage\data_block3_regress\df_error_calc_EMS.csv', index=False)

: df_error_calc_SS.to_csv(r'data_storage\data_block3_regress\df_error_calc_SS.csv', index=False)

: df_error_calc_EMS

```

	Model	R2	MAE	MSE	MaxER	RMSE	MCVS	StdDS
0	model_lr	0.037741	0.176637	0.049992	0.633008	0.223588	-0.096812	0.168265
1	model_knr	0.026493	0.185602	0.050576	0.622712	0.224891	-0.317342	0.275991
2	model_rfr	0.114498	0.178494	0.046004	0.507401	0.247454	-0.385	0.213267
3	model_svr	-0.075523	0.192128	0.055876	0.612608	0.269565	-0.188618	0.274704
4	model_mlpr	-0.383212	0.217781	0.071861	0.626519	0.259056	-0.486948	0.443502

```

: df_error_calc_SS

```

	Model	R2	MAE	MSE	MaxER	RMSE	MCVS	StdDS
0	model_lr	0.887107	0.055714	0.005047	0.183987	0.071044	0.932461	0.038684
1	model_knr	0.808478	0.072267	0.008563	0.258085	0.092534	0.868781	0.051964
2	model_rfr	0.887779	0.056953	0.005017	0.177833	0.317756	0.907745	0.050048
3	model_svr	0.843971	0.066948	0.006976	0.230442	0.292071	0.832279	0.059178
4	model_mlpr	0.841984	0.068706	0.007065	0.2047	0.281166	0.865347	0.095879

Рис. 64 Таблицы с расчетными значениями показателей регрессии

Для целевой переменной "elasticity\_module\_stretching" - EMS лучший показатель продемонстрировала модель RandomForestRegressor() 0.1145.

Хуже всего результаты у MLPRegressor() (-0.3832).

Для целевой переменной 'strapery\_strength' - SS лучший показатель продемонстрировала модель RandomForestRegressor() 0.8878, следом идет LinearRegression() 0.8871.

Хуже всего результаты у KNeighborsRegressor() 0.8084.

Посмотрим, что покажет Линейная регрессия на основе статистических модулей sm.OLS. Это наиболее полный статистический анализ.

Для X\_Train\_EMS

OLS Regression Results									
Dep. Variable:	elasticity_module_stretching	R-squared:	0.058						
Model:	OLS	Adj. R-squared:	0.044						
Method:	Least Squares	F-statistic:	4.046						
Date:	Thu, 20 Apr 2023	Prob (F-statistic):	0.00808						
Time:	19:07:51	Log-Likelihood:	17.550						
No. Observations:	200	AIC:	-27.10						
Df Residuals:	196	BIC:	-13.91						
Df Model:	3								
Covariance Type:	nonrobust								
	coef	std err	t	P> t	[0.025	0.975]			
const	0.5482	0.070	7.850	0.000	0.411	0.686			
elasticity_module	-0.0689	0.081	-0.856	0.393	-0.228	0.090			
content_epoxy_groups	0.1598	0.086	1.866	0.064	-0.009	0.329			
alfa	-0.2194	0.083	-2.635	0.009	-0.384	-0.055			
Omnibus:	1.904	Durbin-Watson:	1.884						
Prob(Omnibus):	0.386	Jarque-Bera (JB):	1.608						
Skew:	-0.072	Prob(JB):	0.448						
Kurtosis:	2.585	Cond. No.	8.91						

Рис.65 Показатели регрессии по модели sm.OLS Для EMS

Sm.OLS метод наименьших квадратов - для EMS показала очень совсем плохой результат - R-squared: 0.058!!!!

Для X\_Train\_SS

[95]:	model_sm.summary()						
[95]:							
	OLS Regression Results						
	<b>Dep. Variable:</b>	strapery_strength	<b>R-squared:</b>	0.944			
	<b>Model:</b>	OLS	<b>Adj. R-squared:</b>	0.943			
	<b>Method:</b>	Least Squares	<b>F-statistic:</b>	1106.			
	<b>Date:</b>	Thu, 20 Apr 2023	<b>Prob (F-statistic):</b>	1.54e-122			
	<b>Time:</b>	19:08:55	<b>Log-Likelihood:</b>	321.08			
	<b>No. Observations:</b>	200	<b>AIC:</b>	-634.2			
	<b>Df Residuals:</b>	196	<b>BIC:</b>	-621.0			
	<b>Df Model:</b>	3					
	<b>Covariance Type:</b> nonrobust						
		<b>coef</b>	<b>std err</b>	<b>t</b>	<b>P&gt; t </b>	<b>[0.025</b>	<b>0.975]</b>
	<b>const</b>	0.0115	0.014	0.804	0.422	-0.017	0.040
	<b>ratio_filler_matrix</b>	0.0014	0.018	0.080	0.936	-0.034	0.037
	<b>elasticity_module</b>	-0.0220	0.018	-1.249	0.213	-0.057	0.013
	<b>alfa</b>	1.0431	0.018	56.989	0.000	1.007	1.079
	<b>Omnibus:</b>	2.866	<b>Durbin-Watson:</b>	1.841			
	<b>Prob(Omnibus):</b>	0.239	<b>Jarque-Bera (JB):</b>	2.034			
	<b>Skew:</b>	0.025	<b>Prob(JB):</b>	0.362			
	<b>Kurtosis:</b>	2.508	<b>Cond. No.</b>	8.22			

Рис. 66 Показатели регрессии по модели sm.OLS Для SS

Sm.OLS метод наименьших квадратов - для SS показала очень хороший результат - R-squared: 0.944!

Построим Графики изменения размера оценочного параметра от типа модели регрессии для целевого параметра EMS.

Графики изменения размера оценочного параметра EMS от типа модели регрессии

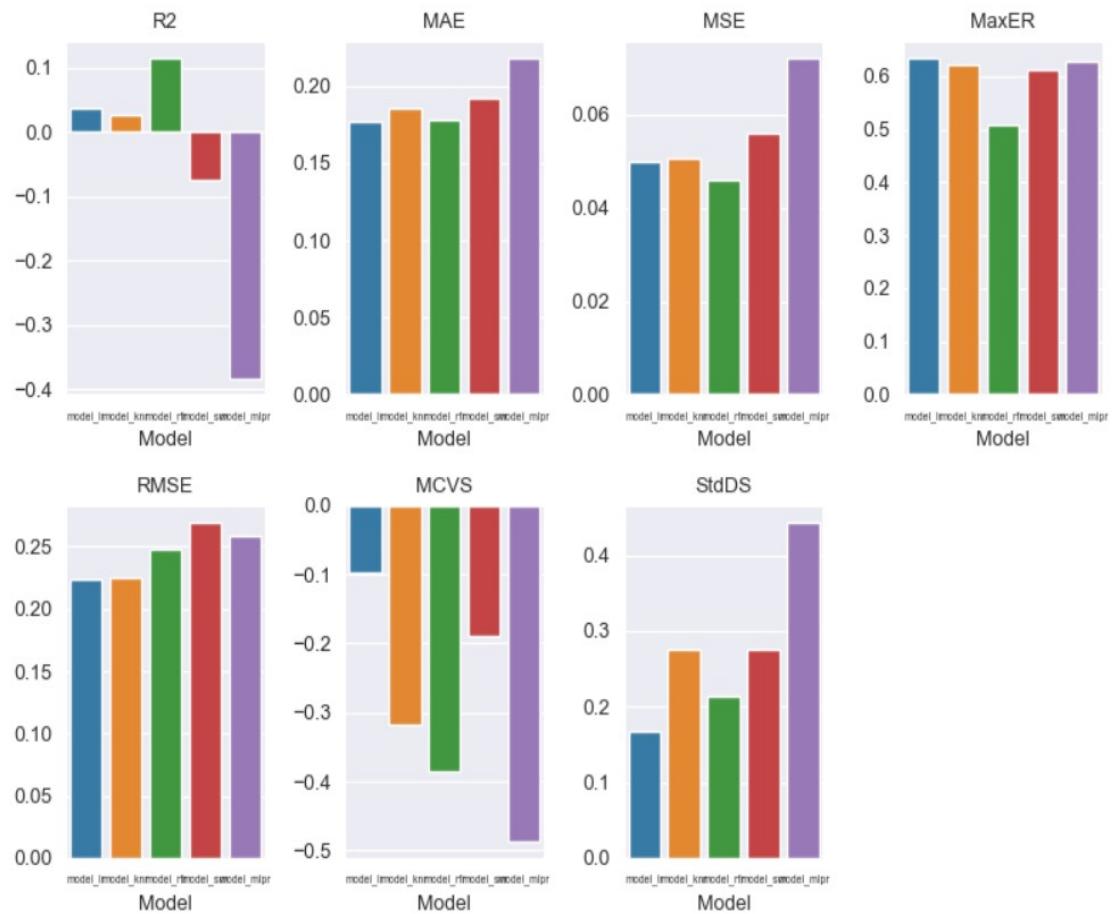


Рис.67 Графики изменения оценочных параметров для EMS

Как видно из графиков основных параметров R2, MAE и MSE, где R2 очень низкий, лучший показатель – чуть выше 0.1 для модели Случайного леса,

и даже отрицательный для Многослойного персептрона, а Средняя абсолютная ошибка MAE достаточно велика.

Можно сделать вывод, что по целевому параметру ‘Модуль упругости при растяжении’ - EMS невозможно осуществлять прогноз.

Sm.OLS метод наименьших квадратов - для EMS показала очень плохой результат - R-squared: 0.058!!!!

Для решения этой задачи необходимы дополнительные данные и другие инструменты для анализа.

Построим Графики изменения размера оценочного параметра от типа модели регрессии для целевого параметра SS.

Графики изменения размера оценочного параметра SS от типа модели регрессии

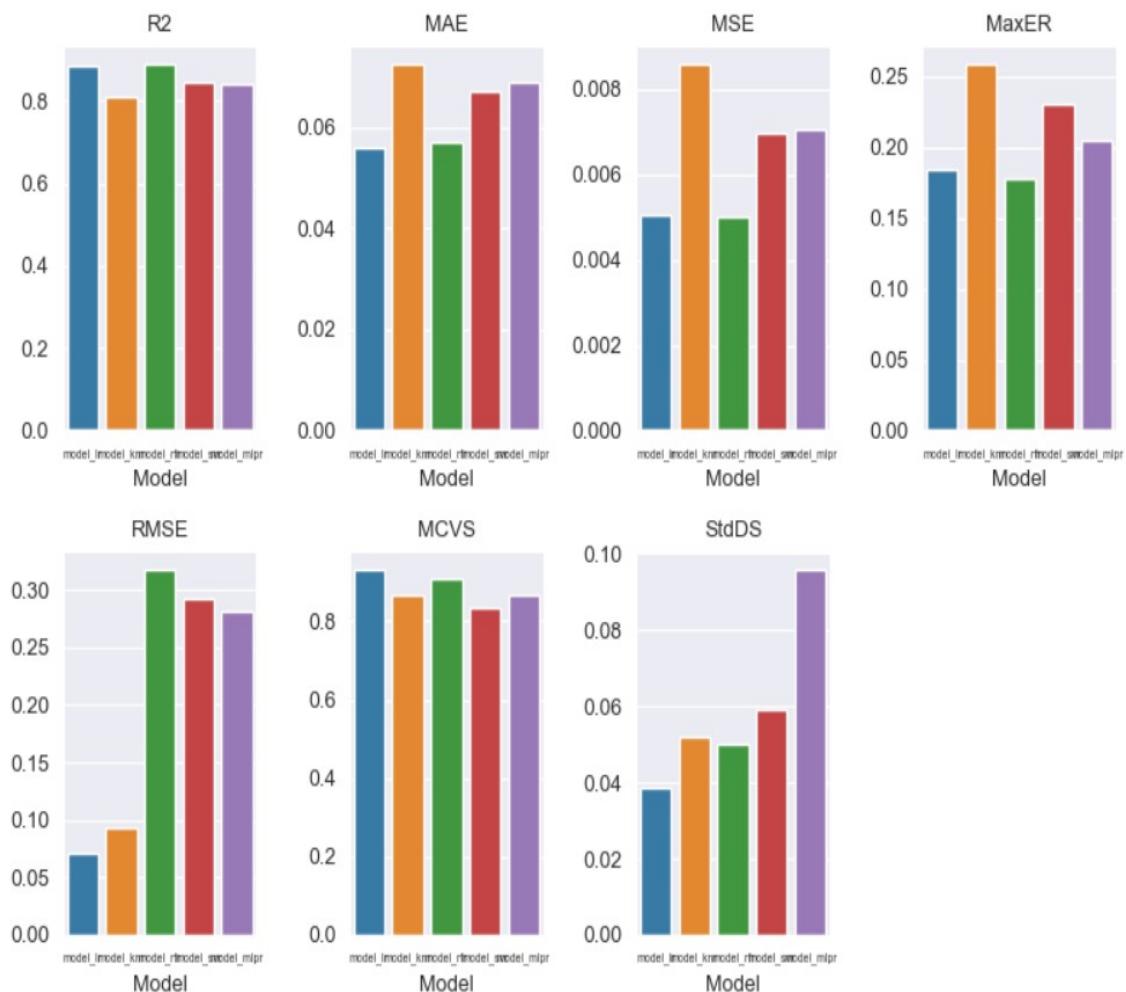


Рис. 68 Графики изменения оценочных параметров для EMS

Как видно из таблиц результатов расчетов и графиков основных параметров R2, MAE и MSE, где R2 показывает достаточно хорошие результаты – более 0.8, а Средняя абсолютная ошибка MAE достаточно невелика, то можно сделать вывод, то по целевому параметру ‘Прочность при растяжении’-SS можно с высокой степенью прогнозировать целевой параметр!!

Sm.OLS метод наименьших квадратов - для SS показала очень хороший результат - R-squared: 0.944! Вполне можно рекомендовать модель sm.OLS для прогнозирования параметра 'strapery\_strength', так как и в целом ряде других моделей на целевой показатель показаны неплохие результаты. И более того при прогнозировании этой целевой переменной и расчетный показатель MAE и MSE имеет очень маленькие значения, что также говорит об устойчивости модели!!

#### 2.3.4. Выводы по Блоку № 3

Проведена предобработка данных, нормализация и стандартизация.

Обучены 6 типов моделей для прогноза модуля упругости при растяжении и прочности при растяжении.

Получены и сведены в 2 df: df\_error\_calc\_EMS.csv и df\_error\_calc\_SS.csv значения по 7 оценочным параметрам работы разных моделей регрессии и записаны в директорию data\_storage/data\_block3\_regress

Сделан вывод о практическом отсутствии регрессии между параметрами и целевой переменной “Модуль упругости при растяжении” и невозможно прогнозировать этот целевой параметр по имеющемуся набору данных.

В тоже время по целевому показателю ‘Прочность при растяжении’ считаю вполне возможно проводить прогнозирование результата с довольно большой точностью!

Полученные в процессе работы Блока 3 графики записаны в соответствующую директорию save\_fig/fig\_block3\_regress.

Созданы полученные в результате расчетов два датасета с 7-ю показателями оценочных параметров по 2-м целевым переменным df\_error\_calc\_EMS и df\_error\_calc\_SS сохранены в папке data\_storage/data\_block3\_regres/.

Код Блока № 3 Diploma\_block3\_regress.ipynb.

#### **2.4. Блок № 4 Создание нейронной сети, которая будет рекомендовать «соотношение матрица-наполнитель».**

##### **2.4.1. Исходные данные для работы Блока № 4**

Библиотеки и модули Python и разработанные функции, используемые в данном Блоке.

Рис.69 Используемые модули Python и функции из другой директории

Для работы при создании модели нейронной сети использовался полный очищенный от выбросов датасет ‘data\_main\_clean.csv’, который получен в Блоке №1.

Датасет ‘data\_clean’ имеет следующие параметры:

```
[91]: data_clean.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 935 entries, 0 to 999
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   pattern_angle    935 non-null    int32  
 1   step_strip       935 non-null    float32 
 2   density_strip    935 non-null    float32 
 3   ratio_filler_matrix 935 non-null    float32 
 4   density         935 non-null    float32 
 5   elasticity_module 935 non-null    float32 
 6   number_hardeners 935 non-null    float32 
 7   content_epoxy_groups 935 non-null    float32 
 8   flash_temperature 935 non-null    float32 
 9   surface_density   935 non-null    float32 
 10  elasticity_module_stretching 935 non-null    float32 
 11  strapery_strength 935 non-null    float32 
 12  resin_consumption 935 non-null    float32 
dtypes: float32(12), int32(1)
memory usage: 54.8 KB
```

Рис. 70 Данные датасета ‘data\_clean’

Мы имеем 13 параметров по 935 строк каждый. Все значения числовые, размерностью int32 и float32, отсутствуют нулевые значения, пропущенных значений нет, датасет очищен от выбросов.

#### 2.4.2. Создание нейронной сети

Теоретические основы (расчет количества слоев и количество нейронов, выбор активационных функций) создания полносвязной нейронной сети изложены в п. 1.2.7 настоящего исследования.

У нас имеется одна целевая переменная - 'ratio\_filler\_matrix' RFM - Соотношение матрица-наполнитель, для которой мы разработаем модель нейронной

сети, которая будет предсказывать на основании остальных 12 параметров значение целевого параметра.

Следовательно, у нас во входном слое будет 12 нейронов, а в выходном слое будет 1 нейрон.

У нас будет 2 скрытых слоя, так как наши данные определены на конечном множестве точек, непрерывны и определены на компактной области.

В качестве активационной функции выбираем функцию активации SELU - масштабируемые экспоненциальные линейные единицы. Она обладает свойствами самонормализации, гарантирует, что все выходные данные будут нормализованы без явного добавления слоя нормализации BatchNorm в модель.

При использовании SELU необходимо использовать инициализатор LecunNormal() - kernel\_initializer='lecun\_normal' и использовать Alpha Dropout().

Проведен обоснованный расчет количества слоев нейронной сети, количества нейронов в каждом слое по методу геометрической пирамиды и выбор активационных функций для каждого слоя.

```
# Проведем расчет количества нейронов в 2 - х скрытых слоях по правилу геометрической пирамиды:
# k -          # число нейронов в скрытом слое
n = 12         # число нейронов во входном слое
m = 1          # число нейронов в выходном слое
r = (n/m) **(1.0/3.0)
k1 = m * r**2 # число нейронов в 1-м скрытом слое
k2 = m * r    # число нейронов во 2-м скрытом слое
```

```
[5]: print('число нейронов во входном слое - ', n)
print('число нейронов в выходном слое - ', m)
print('число нейронов в 1-м скрытом слое - ', math.ceil(k1)) # округляем в большую сторону
print('число нейронов во 2-м скрытом слое - ', math.ceil(k2)) # округляем в большую сторону
```

```
число нейронов во входном слое - 12
число нейронов в выходном слое - 1
число нейронов в 1-м скрытом слое - 6
число нейронов во 2-м скрытом слое - 3
```

Рис.71 Расчет количества нейронов по слоям

Изменим порядок столбцов в нашем датасете, для наглядной работы в Блоке № 5 при написании приложения FLASK.

Поставим на ПЕРВОЕ место наш целевой показатель 'ratio\_filler\_matrix' – (сокращенно RFM) ‘Соотношение матрица-наполнитель’ и запишем в новый ‘data\_main\_clean\_flask’.

```
data_main_clean_flask = data_main_clean.reindex(columns=['ratio_filler_matrix',
                                                       'pattern_angle',
                                                       'step_strip',
                                                       'density_strip',
                                                       'density',
                                                       'elasticity_module',
                                                       'number_hardeners',
                                                       'content_epoxy_groups',
                                                       'flash_temperature',
                                                       'surface_density',
                                                       'elasticity_module_stretching',
                                                       'strapery_strength',
                                                       'resin_consumption'
])
```

Рис. 72 Новая структура датасета

Мы провели разделение дата сета на целевую переменную y\_RFМ и массив X\_RFМ.

Раздельно проводим нормализацию методом MinMaxScaler() для ‘y\_RFМ’ - data\_scaler\_y\_norm и массива ‘X\_RFМ’- data\_scaler\_X\_norm для использования в приложении Flask при обработке введенных физических значений переменных и обратного инвертирования полученных данных в модели в физические данные. Запишем нормализаторы scaler\_y и scaler\_X в папку data\_storage/model\_RFМ\_flask.

Проводим разделение наших нормализованных объектов data\_scaler\_y\_norm и data\_scaler\_y\_norm методом train\_test\_split, выделив 30% для тестовой выборки.

Создаем последовательную модель нейронной сети и создадим объект класса model\_RFМ = Sequential().

Создаем нейросеть ‘model\_RFМ’ из 4 слоев с помощью библиотеки tensorflow.keras.

```
# Создаем нейросеть из 4 слоев:  
# 1-й - Входной слой размерностью n = 12 - по количеству параметров в нашем df  
# 2-й - Скрытый слой размерностью k1 = 6 - получен из расчета по правилу геометрии  
# 3-й - Скрытый слой размерностью k2 = 3 - получен из расчета по правилу геометрии  
# 4-й - Выходной слой размерностью m = 1 - по количеству целевых параметров  
  
model_RFМ.add(Dense(12)) # входной полносвязный слой  
model_RFМ.add(Dense(6, kernel_initializer='lecun_normal', activation='selu'))  
model_RFМ.add(AlphaDropout(0.25))  
model_RFМ.add(Dense(3, kernel_initializer='lecun_normal', activation='selu'))  
model_RFМ.add(AlphaDropout(0.25))  
model_RFМ.add(Dense(1, activation= 'linear')) # выходной слой
```

Рис.73 Код модели нейронной сети

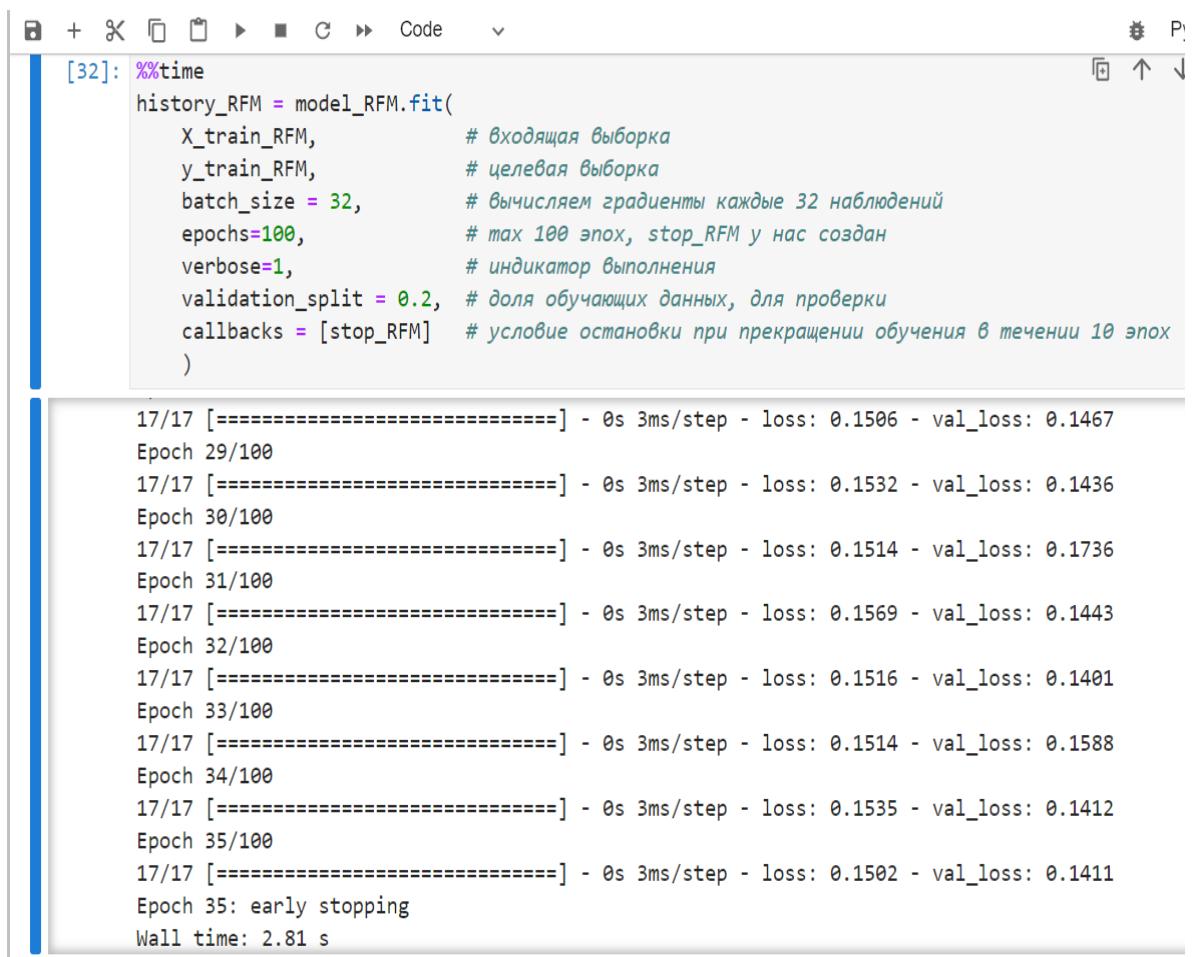
Создали ‘stop\_RFМ’ - индикатор для остановки обучения нейросети методом ‘EarlyStopping()’, когда сеть перестает улучшаться в течении 15 эпох – зададим аргумент patience=15. Затем проводим компиляцию модели. Вызывать будем оптимизатор по имени Adam, learning\_rate=0.02 - Шаг обучения 0.02 и loss - функция ошибки - "средняя абсолютная ошибка".

```
[30]: # stop_RFM - индикатор для остановки обучения нейросети, когда она перестает улучшаться в течении 15
stop_RFM = EarlyStopping(monitor='val_loss', min_delta=0, patience=15, verbose=1, mode='auto')
```

```
[31]: # Компиляция модели. Вызывать будем оптимизатор по имени Adam.
# Learning_rate=0.02 - Шаг обучения 0.02
# Lossu -функция ошибки - "средняя абсолютная ошибка"
model_RFM.compile(optimizer = tf.optimizers.Adam(learning_rate=0.02), loss = 'mean_absolute_error')
```

Рис. 74 Индикатор остановки и компилятор модели

Запускаем модель на обучение.



[32]: %time

```
history_RFM = model_RFM.fit(
    X_train_RFM,           # входящая выборка
    y_train_RFM,           # целевая выборка
    batch_size = 32,        # вычисляем градиенты каждые 32 наблюдений
    epochs=100,             # max 100 эпох, stop_RFM у нас создан
    verbose=1,              # индикатор выполнения
    validation_split = 0.2,  # доля обучающих данных, для проверки
    callbacks = [stop_RFM]  # условие остановки при прекращении обучения в течении 10 эпох
)
```

17/17 [=====] - 0s 3ms/step - loss: 0.1506 - val\_loss: 0.1467  
Epoch 29/100  
17/17 [=====] - 0s 3ms/step - loss: 0.1532 - val\_loss: 0.1436  
Epoch 30/100  
17/17 [=====] - 0s 3ms/step - loss: 0.1514 - val\_loss: 0.1736  
Epoch 31/100  
17/17 [=====] - 0s 3ms/step - loss: 0.1569 - val\_loss: 0.1443  
Epoch 32/100  
17/17 [=====] - 0s 3ms/step - loss: 0.1516 - val\_loss: 0.1401  
Epoch 33/100  
17/17 [=====] - 0s 3ms/step - loss: 0.1514 - val\_loss: 0.1588  
Epoch 34/100  
17/17 [=====] - 0s 3ms/step - loss: 0.1535 - val\_loss: 0.1412  
Epoch 35/100  
17/17 [=====] - 0s 3ms/step - loss: 0.1502 - val\_loss: 0.1411  
Epoch 35: early stopping  
Wall time: 2.81 s

Рис. 75 Пример работы модели нейронной сети

Модель остановила работу на 35 эпохе - сработал предохранитель!

Лучший показатель по MAE- 0.1633 достигнут в 9 эпоху.

```
[33]: # Оценка модели по лучшему расчетному показателю MAE
evaluation_model = model_RFM.evaluate(x=X_test_RFM,
                                         y=y_test_RFM)
print("Оценка модели = ", evaluation_model)
```

```
9/9 [=====] - 0s 1ms/step - loss: 0.1634
Оценка модели = 0.1633787751197815
```

```
[ ]: # Лучший показатель MAE был достигнут во время 9 эпохи и составил MAE(9) = 0.1633
```

Рис. 76 Оценка модели нейронной сети

Модель обучила 259 параметров.

```
[34]: model_RFM.summary()
Model: "sequential"
-----  

Layer (type)           Output Shape        Param #
-----  

dense (Dense)          (None, 12)         156  

dense_1 (Dense)         (None, 6)          78  

alpha_dropout (AlphaDropout) (None, 6)      0  

)  

dense_2 (Dense)         (None, 3)          21  

alpha_dropout_1 (AlphaDropout) (None, 3)      0  

)  

dense_3 (Dense)         (None, 1)          4  

-----  

Total params: 259
Trainable params: 259
Non-trainable params: 0
```

Рис. 77 Архитектура модели и количество обученных параметров

Использую созданные функцию `model_loss_plot` построения графика ошибки модели на тренировочной и тестовой выборках и функцию `actual_predict_plot` для визуализации тестовых и прогнозных значений MSE построили графики.

```
[80]: # Построение графика ошибки MAE - mean_absolute_error функцией model_loss_plot
model_loss_plot(history_RF, graf_name_path = r'save_fig\fig_block4_neuro\model_loss_MAE_flask.png')
```

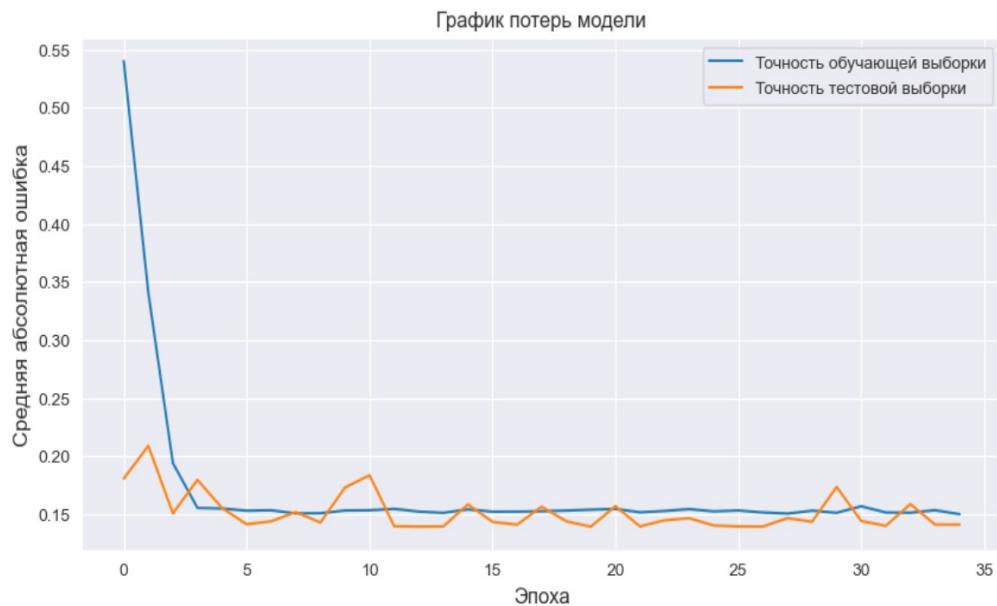


Рис.78 График МАЕ обучающей и тестовой выборки

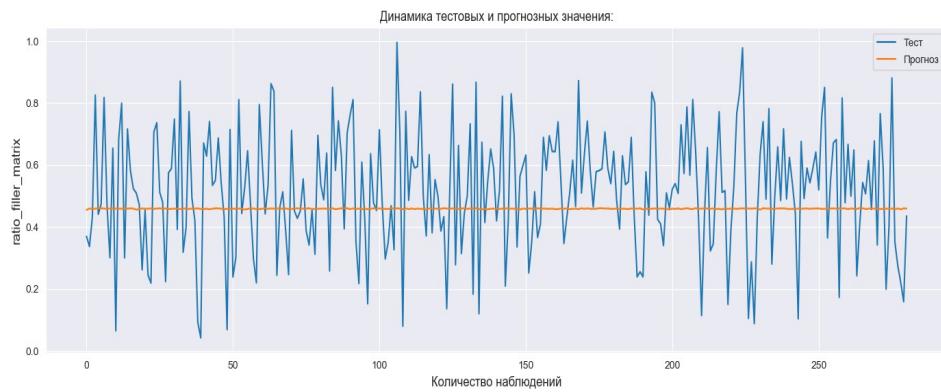


Рис. 79 График динамики тестовых и прогнозных значений

Выводим первое значений нашего предсказанного моделью model\_RFМ значения целевой переменной для последующего сравнения с результатами из загруженной из папки модели - 'y\_pred\_RFМ[0][0]' = 0.4555432

Полученная и обученная модель нейронной сети была сохранена в директории data\_storage/data\_block5\_neuro/model\_RFМ\_flask.

```
[68]: # Сохранение модели в папку
model_RFМ.save(r'data_storage\data_block5_flask\model_RFМ_flask')

INFO:tensorflow:Assets written to: data_storage\data_block5_flask\model_RFМ_flask\assets

[83]: # Сохранение 'MinMaxScaler'
with open(r'data_storage\data_block5_flask\model_RFМ_flask\scaler_y.pkl', 'wb') as f:
    pickle.dump(scaler_y,f)

[84]: with open(r'data_storage\data_block5_flask\model_RFМ_flask\scaler_X.pkl', 'wb') as f:
    pickle.dump(scaler_X,f)
```

Рис. 80 Код записи модели в папку

```
[70]: # Загрузка сохраненной модели из папки
loaded_model_RFМ = keras.models.load_model(r'data_storage\data_block4_neuro\model_RFМ_flask')
loaded_model_RFМ.predict(X_test_RFМ)[0][0] # проверка загруженной модели- совпадает с рабочей!
9/9 [=====] - 0s 2ms/step
[70]: 0.4555432

[ ]: # Результаты совпали с данными полученными перед сохранением модели в папку

[72]: # загрузка 'MinMaxScaler'
with open(r'data_storage\data_block5_flask\model_RFМ_flask\scaler_y.pkl', 'rb') as f:
    scaler_y = pickle.load(f)
with open(r'data_storage\data_block5_flask\model_RFМ_flask\scaler_X.pkl', 'rb') as f:
    scaler_X = pickle.load(f)

[66]: # Переводим полученное значение y_pred_RFМ в естественную шкалу значений по процедуре inverse_transform
y_pred_origin_data = scaler_y.inverse_transform(np.array(y_pred_RFМ))
y_pred_origin_data[0][0]
[66]: 2.7188525
```

Рис. 81 Код загрузки модели и расчета физического значения y\_pred\_RFМ

Мы получили предсказанное значение y\_pred\_RFМ[0][0] = 0.4555432 из загруженной сохраненной модели, которое равное значению до сохранения модели в папку.

Инвертировали нормализованное значение в физически значимое и получили значение y\_pred\_origin\_data = 2.7188. Диапазон нашей целевой переменной 'ratio\_filler\_matrix' действительно лежит в пределах 0.6 - 5.3!

Модель работает!

#### 2.4.2. Выводы по Блоку № 4

Для работы с нейросетью был применен полный очищенный от выбросов df df data\_main\_clean.csv, который получен в Блоке №1.

Проведен обоснованный расчет количества слоев нейронной сети, количества нейронов в каждом слое и выбор активационных функции для каждого слоя.

Для последующей работы при создании приложения на flask целевой параметр 'ratio\_filler\_matrix' был перемещен на первый столбец в data\_main\_clean\_flask. Также создан df\_min\_max\_flask для приложения flask с границами допустимых значений для ввода значений пользователя при расчете целевого параметра.

Написана нейронная сеть, которая будет рекомендовать соотношение целевого параметра 'ratio\_filler\_matrix' "Матрица-наполнитель".

Оценены точности моделей на тренировочном и тестовом датасете.

Методом обратной инверсии проверено, что предсказанные моделью данные y\_pred, полученные в нормализованном виде при инверсии на сохраненной и вновь загруженной модели, показывают отличное корреляцию с естественной размерностью данного параметра.

Предобученная модель нейронной сети была сохранена в директории data\_storage/data\_block4\_neuro/model\_RFМ\_flask.

Полученные в процессе работы Блока № 4 графики записаны в соответствующую директорию save\_fig/fig\_block4\_neuro.

Код по Блоку № 4 - Diploma\_block4\_neuro.ipynb.

### 2.5. Разработка приложения

Разработано веб-приложение для рекомендательной системы «Соотношение матрица-наполнитель». Приложение разработано в среде разработки

VSCode. Для разработки приложения был использован веб-фреймворк Flask. На рисунке \_\_\_\_\_ показан интерфейс приложения. Пользователь вводит 12 параметров в реальных физических значениях в соответствующие окна ввода, с учетом рекомендованных интервалов значений, и при нажатии кнопки «Рассчитать значение» Приложение выводит ниже расчетное значение показателя «Соотношение матрица-наполнитель».

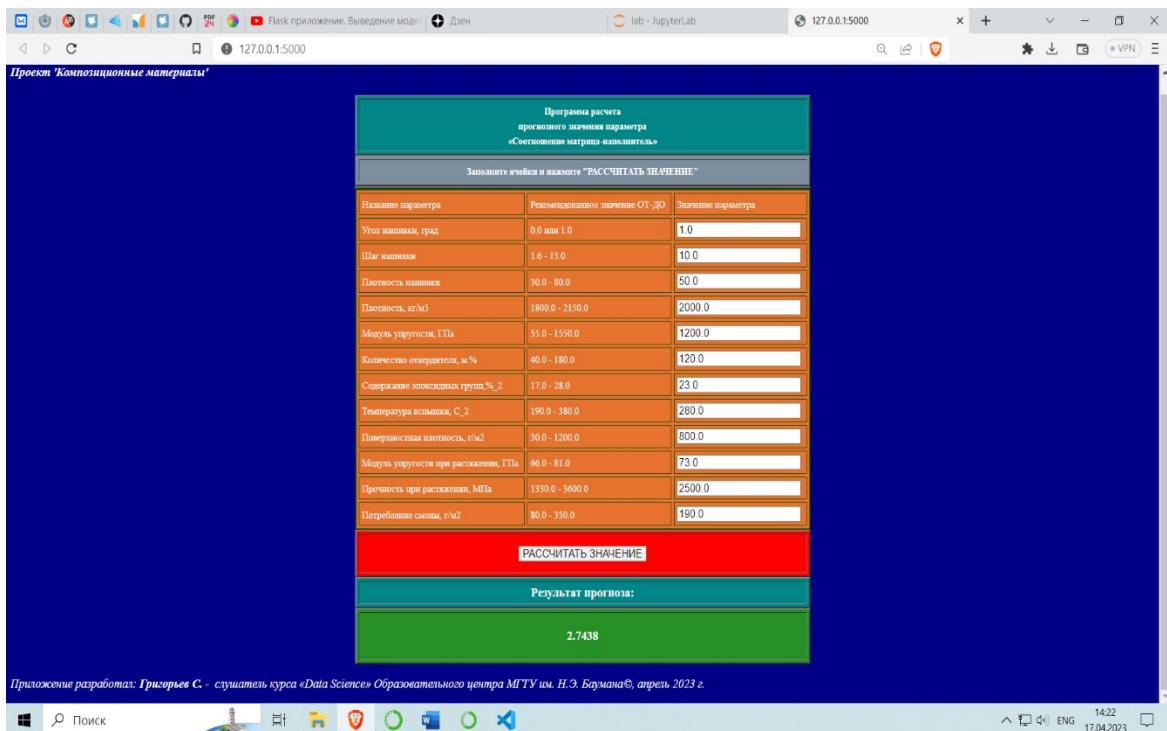


Рисунок 82 - Пример результата работы приложения

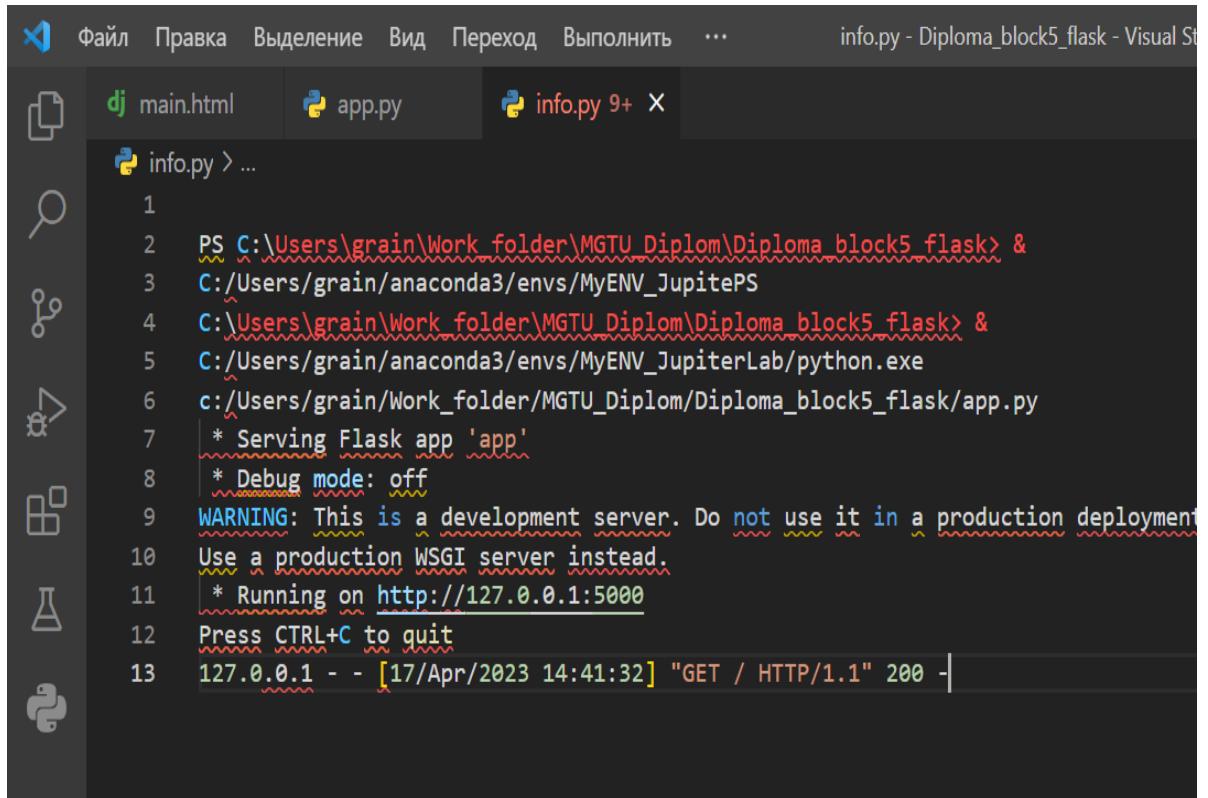
Приложение успешно работает на локальном хосте и показывает результат прогноза для параметра «соотношения матрица – наполнитель».

Данное приложение — это основной файл Flask app.py, расположенный в папке Diploma\_block5\_flask. Код основного файла Приложения - app.py.

Рис. 83 Код основного файла app.py Приложения

Остальные файлы проекта flask расположены в папке Diploma\_block5\_flask, в папке templates находится файл main.html с шаблоном страницы и в папке data\_block5\_flask папка model\_RFМ\_flask с сохранённой моделью для данных и с файлами scaler\_X.pkl и scaler\_y.pkl – нормализаторами для передачи физических данных в сохраненную модель в нормализованном виде и получения предсказанного значения параметра Соотношение матрица-наполнитель обратно из нормализованного значения в физически значимое и ввод его в Результаты расчета.

При запуске приложения пользователь переходит на страницу сайта Приложения, расположенную на локальном хосте: <http://127.0.0.1:5000/>. Код 200 свидетельствует о нормальной работе приложения.



```

Файл Правка Выделение Вид Переход Выполнить ...
info.py - Diploma_block5_flask - Visual Studio Code

dj main.html app.py info.py 9+ X

info.py > ...
1
2 PS C:\Users\grain\Work_folder\MGTU_Diplom\Diploma_block5_flask> &
3 C:/Users/grain/anaconda3/envs/MyENV_JupiterPS
4 C:\Users\grain\Work_folder\MGTU_Diplom\Diploma_block5_flask> &
5 C:/Users/grain/anaconda3/envs/MyENV_JupiterLab/python.exe
6 c:/Users/grain/Work_folder/MGTU_Diplom/Diploma_block5_flask/app.py
7 * Serving Flask app 'app'
8 * Debug mode: off
9 WARNING: This is a development server. Do not use it in a production deployment
10 Use a production WSGI server instead.
11 * Running on http://127.0.0.1:5000
12 Press CTRL+C to quit
13 127.0.0.1 - - [17/Apr/2023 14:41:32] "GET / HTTP/1.1" 200 -

```

Рис.84 Ссылка для открытия страницы на локальном сервере

## 2.6. Создание удалённого репозитория и загрузка результатов работы на него

Репозиторий был создан на [github.com](https://github.com/) по адресу: [https://github.com/Grain1963/MHTS\\_Diploma](https://github.com/Grain1963/MHTS_Diploma)

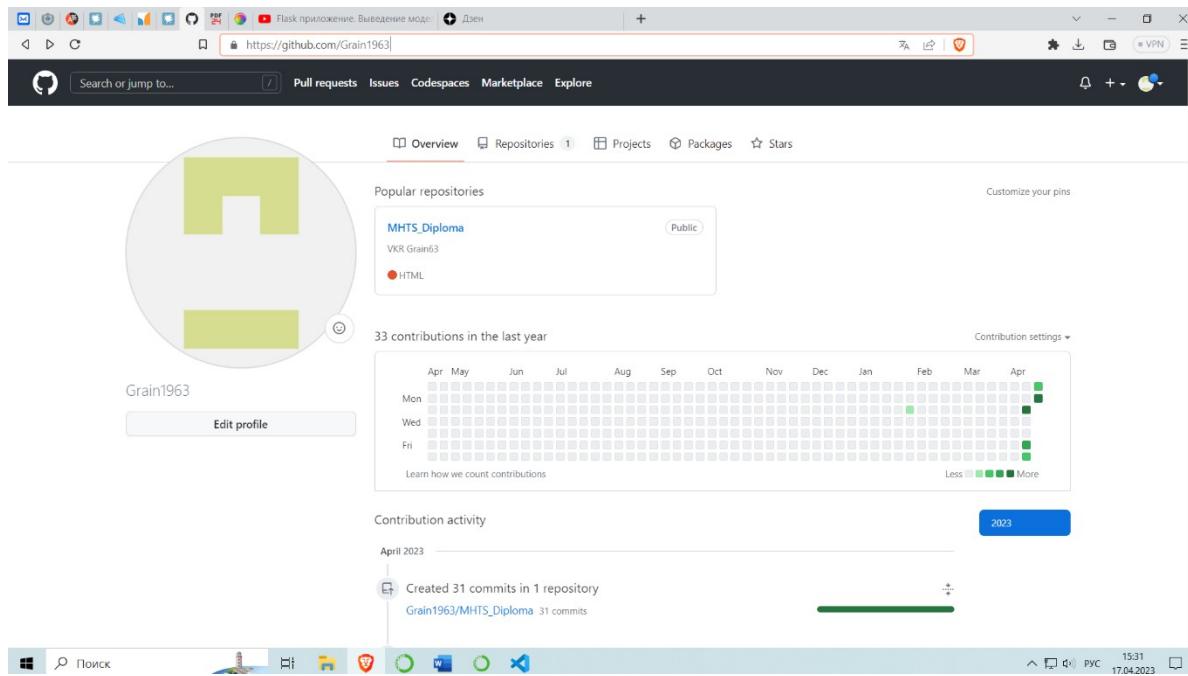


Рис. 85 Часть страницы на [github.com](https://github.com)

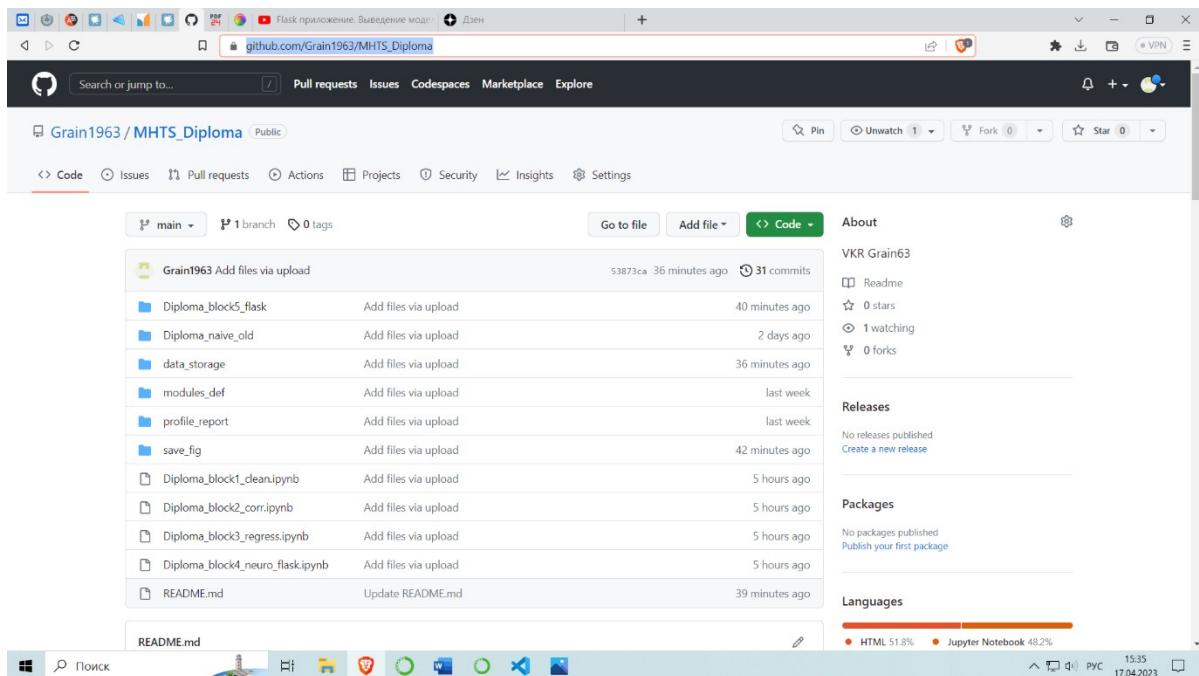


Рис 86 Структура репозитария на github.com

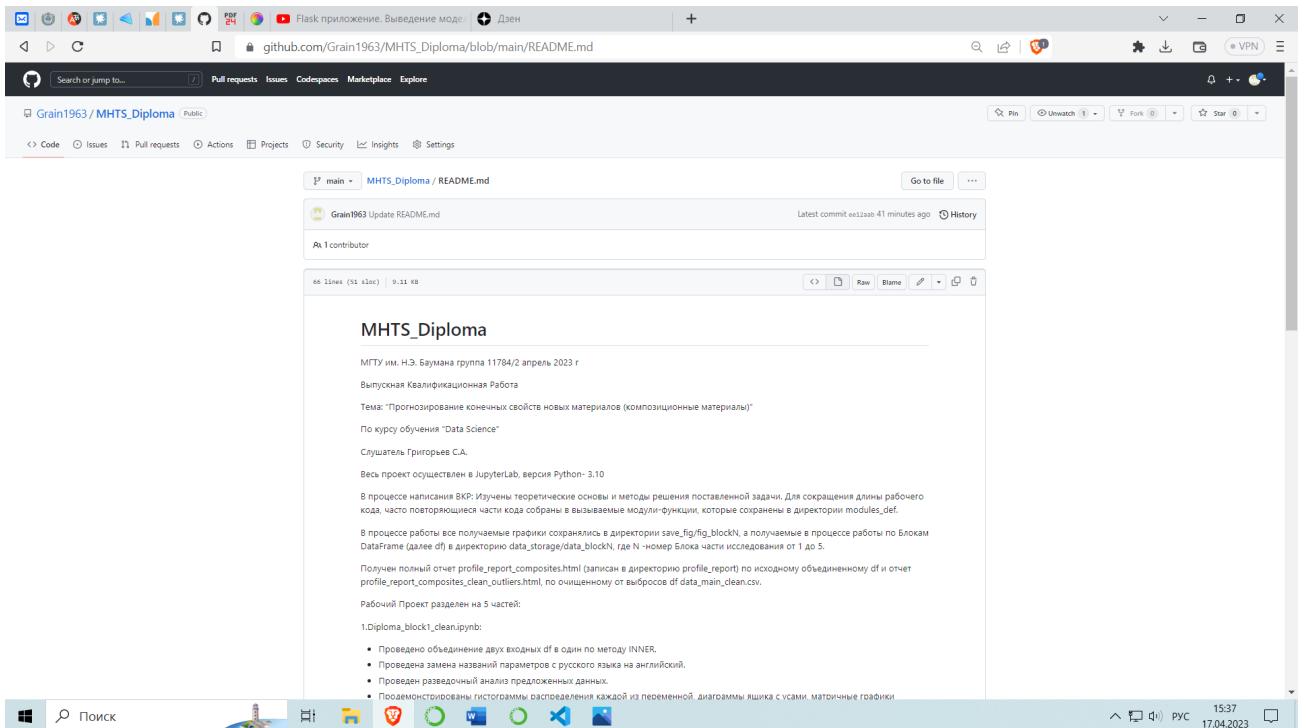


Рис. 87 Часть созданного описательного файла README

## Заключение

В ходе выполнения данной работы были изучены теоретические основы методов машинного обучения, изучены основные библиотеки Python, как одного из основных инструментов для работы аналитика данных.

На основании практической задачи были изучены и реализованы:

- разведочный анализ данных;
- предобработка данных;
- построение регрессионных моделей и модели нейронной сети;
- визуализация модели и оценка качества прогноза;
- разработка и тестирование веб-приложения.

Данная исследовательская работа позволяет сделать некоторые основные выводы по теме. Распределение полученных данных в объединённом датасете близко к нормальному, но коэффициенты корреляции между парами признаков стремятся на полном датасете к нулю.

При работе с небольшим набором данных 200 строк для Train и 100 строк для Test получилось решить задачу по целевому показателю ‘Прочность при растяжении’ считаю вполне возможно проводить прогнозирование результата с довольно большой точностью! Как показали расчеты по 5 моделям основных параметров R<sup>2</sup>, MAE и MSE, где R<sup>2</sup> показывает достаточно хорошие результаты – более 0.8, а Средняя абсолютная ошибка MAE достаточно невелика, то можно сделать вывод, что по целевому параметру ‘Прочность при растяжении’/‘strapery\_strength’-SS можно с высокой степенью прогнозировать целевой параметр!!

Отличный результат показала модель Sm.OLS метод наименьших квадратов - для 'strapery\_strength' SS - R-squared: 0.944! Вполне можно рекомендовать

модель sm.OLS для прогнозирования параметра ‘Прочность при растяжении’/ ‘strapery\_strength’. Более того, при прогнозировании этой целевой переменной и расчетный показатель MAE и MSE имеет очень маленькие значения, что также говорит об устойчивости модели!!

Был сделан вывод, что невозможно определить из свойств материалов соотношение «матрица – наполнитель». Данный факт не указывает на то, что прогнозирование характеристик композитных материалов на основании представленного набора данных невозможно, но может указывать на недостатки базы данных, подходов, использованных при прогнозе, необходимости пересмотра инструментов для прогнозирования.

Необходимы дополнительные вводные данные, получение новых результирующих признаков в результате математических преобразований, релевантных доменной области, консультации экспертов предметной области, новые исследования, работа эффективной команды, состоящей из различных учёных.

Разработанные модели при определенных условиях и дальнейшей доработки могут быть использованы для переобучения и решения аналогичных задач на новых входных данных.

Рабочий Проект разделен на 5 частей:

### **1.Diploma\_block1\_clean.ipynb:**

Проведено объединение двух входных df в один по методу INNER.

Проведена замена названий параметров с русского языка на английский.

Проведен разведочный анализ предложенных данных.

Продемонстрированы гистограммы распределения каждой из переменной, диаграммы ящика с усами, матричные графики рассеяния точек.

Для каждого параметра получены среднее, медианное значение, проверено наличие пропусков, проведен итерационный анализ и исключение выбросов несколькими методами.

Осуществлена замена “подозрительных значений” на средние значения ближайших 5 соседей.

По результатам работы в Блоке № 1 для дальнейшего исследования создан очищенный от выбросов df data\_main\_clean.csv, который записан в папку data\_block1\_clean, которая входит в директорию верхнего уровня data\_storage.

Все полученные графики сохранены в директории save\_fig/fig\_block1\_clean.

#### **Diploma\_block2\_corr.ipynb:**

Проведено исследование корреляционных матриц очищенного от выбросов и подозрительных значений df в зависимости от количества значений в выборке.

Сделан вывод, что при рассмотрении полного df корреляция отсутствует практически полностью, однако при значениях  $N = 100$  и  $200$  еще наблюдается некоторая корреляционная зависимость между параметрами, с увеличением числа выборки зависимость падает до около нулевых значений.

Проведен графический анализ использованием регрессионной прямой по каждому параметру с целевыми переменными “Модуля упругости при растяжении” и “Прочности при растяжении”.

На основании теоретических исследований, приведенных в научной литературе по данной теме введен новый параметр ALFA как функция Модуля упругости при растяжении и Прочности при растяжении.

Произведен выбор фитчей для наших целевых параметров, что позволит сократить вычислительные ресурсы и увеличить точность расчетов.

Подготовлен df data\_Train\_ML.csv из 200 случайных значений (random\_state=42) в качестве обучающего массива для регрессионного анализа в оке 3 и df data\_Test\_ML.csv для использования в качестве тестового набора данных (random\_state=50). Полученные df записаны в соответствующую директорию data\_storage/data\_block2\_corr.

Полученные в процессе работы Блока 2 графики записаны в соответствующую директорию save\_fig/fig\_block2\_corr

#### **Diploma\_block3\_regress.ipynb:**

Проведена предобработка данных, нормализация и стандартизация.

Обучены 6 типов моделей для прогноза модуля упругости при растяжении и прочности при растяжении.

Получены и сведены в 2 df: df\_error\_calc\_EMS.csv и df\_error\_calc\_SS.csv значения по 7 оценочным параметрам работы разных моделей регрессии и записаны в директорию data\_storage/data\_block3\_regress/

Сделан вывод о практическом отсутствии регрессии между параметрами и целевой переменной “Модуль упругости при растяжении” и невозможно прогнозировать этот целевой параметр по имеющемуся набору данных.

В тоже время по целевому показателю ‘Прочность при растяжении’ считаю вполне возможно на небольшом датасете проводить прогнозирование результата с довольно большой точностью!

Полученные в процессе работы Блока 3 графики записаны в соответствующую директорию save\_fig/fig\_block3\_regress.

#### **Diploma\_block4\_neuro.ipynb:**

Для работы с нейросетью был применен полный очищенный от выбросов df df data\_main\_clean.csv, который получен в Блоке №1.

Проведен обоснованный расчет количества слоев нейронной сети, количества нейронов в каждом слое и выбор активационных функции для каждого слоя.

Написана нейронная сеть, которая будет рекомендовать соотношение цевого параметра “Матрица-наполнитель”.

Оценены точности моделей на тренировочном и тестовом датасете.

Полученная и обученная модель нейронной сети была сохранена в директории `data_storage/data_block5_neuro/model_RFМ_flask`.

Полученные в процессе работы Блока 4 графики записаны в соответствующую директорию `save_fig/fig_block4_neuro`.

#### **Diploma\_block5\_flask (папка):**

Блок создания приложения в модуле FLASK по прогнозированию целевого параметра Соотношение матрица-наполнитель.

Создано приложение `app.py` и запущено в работу. Результаты отработанного приложения приведены в конце файла. В процессе работы приложения Пользователь заносит рабочие значения параметров, а Приложение сначала нормализует данные, затем передает их в сохраненную модель, а затем инвертирует расчетные значения в обычную размерность параметра “Соотношение матрица-наполнитель”.

Создан шаблон сайта - файл `main.html` в папке `templates`

В папке `templates` расположен снимок с экрана работающего Приложения на рабочем локальном хосте `http://127.0.0.1:5000`

По итогам работ над Проектом был создан репозиторий на web ресурсе GitHub, в котором выложены все материалы. Ссылка на репозиторий: [https://github.com/Grain1963/MHTS\\_Diploma](https://github.com/Grain1963/MHTS_Diploma).

## Список используемой литературы и веб ресурсы

1. Артеменко, С.Е. Структура и свойства базальто-, стекло- и углепластиков, сформированных по интеркаляционной технологии / С.Е. Артеменко, О.Г. Васильева, Ю.А. Кадыкова, А.Н. Леонтьев // Полимеры-2004: III Всерос. Каргинская конф. М., 2004. Т. 2. С. 162.
2. Джулли, П.: Библиотека Keras - инструмент глубокого обучения / пер. с англ. А.А. Слинкин. – ДМК Пресс, 2017. – 249 с.
3. Грас, Джоэл. Data Science. Наука о данных с нуля: пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.
4. 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.
5. Кадыкова, Ю.А. Полимерные композиционные материалы на основе волокон различной химической природы / Ю.А. Кадыкова, А.Н. Леонтьев, О.Г. Васильева, С.Е. Артеменко // Строительные материалы, оборудование, технологии XXI века. 2002. № 6. С. 10-11.
6. Николенко, С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей. - СПб.: Питер. - 2020. - 480 с. ISBN: 978-5-4461-1537-2.

7. Рассел С., Норвиг П. Искусственный интеллект: современный подход, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2007. - 1408 с.
8. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. - М.: Горячая Линия - Телеком. - 2013. - 384 с. ISBN: 978-5-9912-0320-3.
9. Статистическая обработка данных, планирование эксперимента и случайные процессы : учебное пособие для вузов / Берикашвили В. Ш., Оськин С. П. - 2-е изд., испр. и доп. - М.: Юрайт, 2021. - 163 с.
10. Фостер Д. Генеративное глубокое обучение. Творческий потенциал нейронных сетей. - СПб.: Питер. - 2020. - 336 с. - ISBN: 978-5-4461-1566-2.
11. Документация по библиотеке keras: – Режим доступа: <https://keras.io/api/>.(дата обращения: 18.03.2023).
12. Документация по библиотеке matplotlib: – Режим доступа: <https://matplotlib.org/stable/users/index.html>. (дата обращения: 20.12.2022)
13. Документация по библиотеке numpy: – Режим доступа: <https://numpy.org/doc/1.22/user/index.html#user>. (дата обращения: 23.12.2022).
14. Документация по библиотеке pandas: – Режим доступа: [https://pandas.pydata.org/docs/user\\_guide/index.html#user-guide](https://pandas.pydata.org/docs/user_guide/index.html#user-guide). (дата обращения: 04.12.2022).
15. Документация по библиотеке scikit-learn: – Режим доступа: [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html). (дата обращения: 05.01.2023).
16. Документация по библиотеке seaborn: – Режим доступа: <https://seaborn.pydata.org/tutorial.html>. (дата обращения: 16.01.2023).
17. Документация по библиотеке Tensorflow: – Режим доступа: <https://www.tensorflow.org/overview> (дата обращения: 10.02.2023).
18. Иванов Д.А., Ситников А.И., Шляпин С.Д – Композиционные материалы: учебное пособие для вузов, 2019. 13 с.

19. Краткий обзор алгоритма машинного обучения Метод Опорных Векторов (SVM) – Режим доступа: <https://habr.com/ru/post/428503/> (дата обращения 07.03.2023)

20. Шитиков В.К., Мастицкий С.Э. (2017) Классификация, регрессия и другие алгоритмы Data Mining с использованием R. 351 с. – Электронная книга, адрес доступа: <https://github.com/ranalytics/data-mining>.

21. Кашнитский, Ю. Открытый курс машинного обучения. Тема 3. Классификация, деревья решений и метод ближайших соседей [Электронный ресурс] : – Режим доступа: <https://habr.com/ru/company/ods/blog/322534/>. (дата обращения: 14.03.2023).

22. Константинов, М. Краткий курс машинного обучения или как создать нейронную сеть для решения скоринг задачи [Электронный ресурс]: – Режим доступа: <https://habr.com/ru/post/340792/>. (дата обращения: 12.03.2023).

23. Масзанский, А. Метод k-ближайших соседей (k-nearest neighbour) [Электронный ресурс]: – Режим доступа: <https://proglab.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19/>. (дата обращения: 09.03.2023).

24. Метод обратного распространения ошибки: математика, примеры, код [Электронный ресурс]: – Режим доступа: <https://neurohive.io/ru/osnovy-data-science/obratnoe-rasprostranenie/>. (дата обращения: 10.04.2023).

25. Радченко, В. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес [Электронный ресурс]: – Режим доступа: <https://habr.com/ru/company/ods/blog/324402>. (дата обращения: 05.03.2023).

26. Федоров, А. Решение задачи регрессии полносвязной нейронной сетью [Электронный ресурс]: – Режим доступа: <https://www.bizkit.ru/2019/11/05/14921/>. (дата обращения: 12.03.2023).

27. Функции активации в нейронных сетях [Электронный ресурс]: – Режим доступа: <http://www.aiportal.ru/articles/neural-networks/activation>

function.html. (дата обращения: 12.03.2022).

28. Хлевнюк, А. Основы линейной регрессии [Электронный ресурс]: – Режим доступа: <https://habr.com/ru/post/514818/>. (дата обращения: 03.03.2023).

29. Бондалетова Л.И. Б811 Полимерные композиционные материалы (часть 1): учебное пособие / Л.И. Бондалетова, В.Г. Бондалетов. – Томск: Изд-во Томского политехнического университета, 2013. – 118 с.

## Приложения

### Приложение А

#### Задание по выпускной квалификационной работе

**Тема:** Прогнозирование конечных свойств новых материалов (композиционных материалов).

Датасет со свойствами композитов. Объединение делать по индексу тип объединения INNER. Датасет расположен:

[https://drive.google.com/file/d/1B1s5gBlvgU81H9GGolLQVw\\_SOivyNf2/  
view?usp=sharing](https://drive.google.com/file/d/1B1s5gBlvgU81H9GGolLQVw_SOivyNf2/view?usp=sharing)

Требуется:

1) Изучить теоретические основы и методы решения поставленной задачи.

2) Провести разведочный анализ предложенных данных. Необходимо нарисовать гистограммы распределения каждой из переменной, диаграммы ящика с усами, попарные графики рассеяния точек. Необходимо также для каждой колонке получить среднее, медианное значение, провести анализ и исключение выбросов, проверить наличие пропусков.

3) Провести предобработку данных (удаление шумов, нормализация и т.д.).

4) Обучить нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении. При построении модели необходимо

30% данных оставить на тестирование модели, на остальных происходит обучение моделей.

- 5) Написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель.
- 6) Разработать приложение с графическим интерфейсом или интерфейсом командной строки, которое будет выдавать прогноз, полученный в задании 4 или 5 (один или два прогноза, на выбор учащегося).
- 7) Оценить точность модели на тренировочном и тестовом датасете.
- 8) Создать репозиторий в GitHub / GitLab и разместить там код исследования. Оформить файл README.

## **Приложение Б (в PDF версии)**

1. Отчет по объединенному датасету, полученному с помощью модуля ProfileReport в html и PDF форматах
2. Отчет по очищенному датасету, полученному с помощью модуля ProfileReport в html и PDF форматах
3. Отчет по объединенному датасету, полученному с помощью модуля SweetVIZ в html и PDF форматах
4. Отчет по очищенному датасету, полученному с помощью модуля SweetVIZ в html и PDF форматах