

**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

Богословский Максим Геннадьевич

Москва, 2022

Содержание

Содержание	2
Введение	3
1. Аналитическая часть	4
1.1. Постановка задачи	4
1.2. Описание используемых методов	6
1.3. Разведочный анализ данных	12
2. Практическая часть	18
2.1. Предобработка данных	18
2.2. Разработка и обучение модели	20
2.3. Тестирование модели	21
2.4. Нейронная сеть	24
2.5. Разработка приложения	29
2.6. Создание удаленного репозитория	30
2.7. Заключение	31
2.8. Библиографический список	32

Введение

Композиционные материалы - это искусственно созданные материалы, состоящие из нескольких других с четкой границей между ними. Композиты обладают теми свойствами, которые не наблюдаются у компонентов по отдельности. При этом композиты являются монолитным материалом, т.е. компоненты материала неотделимы друг от друга без разрушения конструкции в целом.

Яркий пример композита - железобетон. Бетон прекрасно сопротивляется сжатию, но плохо растяжению. Стальная арматура внутри бетона компенсирует его неспособность сопротивляться сжатию, формируя тем самым новые, уникальные свойства. Современные композиты изготавливаются из других материалов: полимеры, керамика, стеклянные и углеродные волокна, но данный принцип сохраняется. У такого подхода есть и недостаток: даже если мы знаем характеристики исходных компонентов, определить характеристики композита, состоящего из этих компонентов, достаточно проблематично. Для решения этой проблемы есть два пути: физические испытания образцов материалов, или прогнозирование характеристик. Суть прогнозирования заключается в симуляции представительного элемента объема композита, на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

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

Работа выполнялась в среде Visual Studio Code на базе Python 3.9.12 и Jupyter Notebook 3.3.2

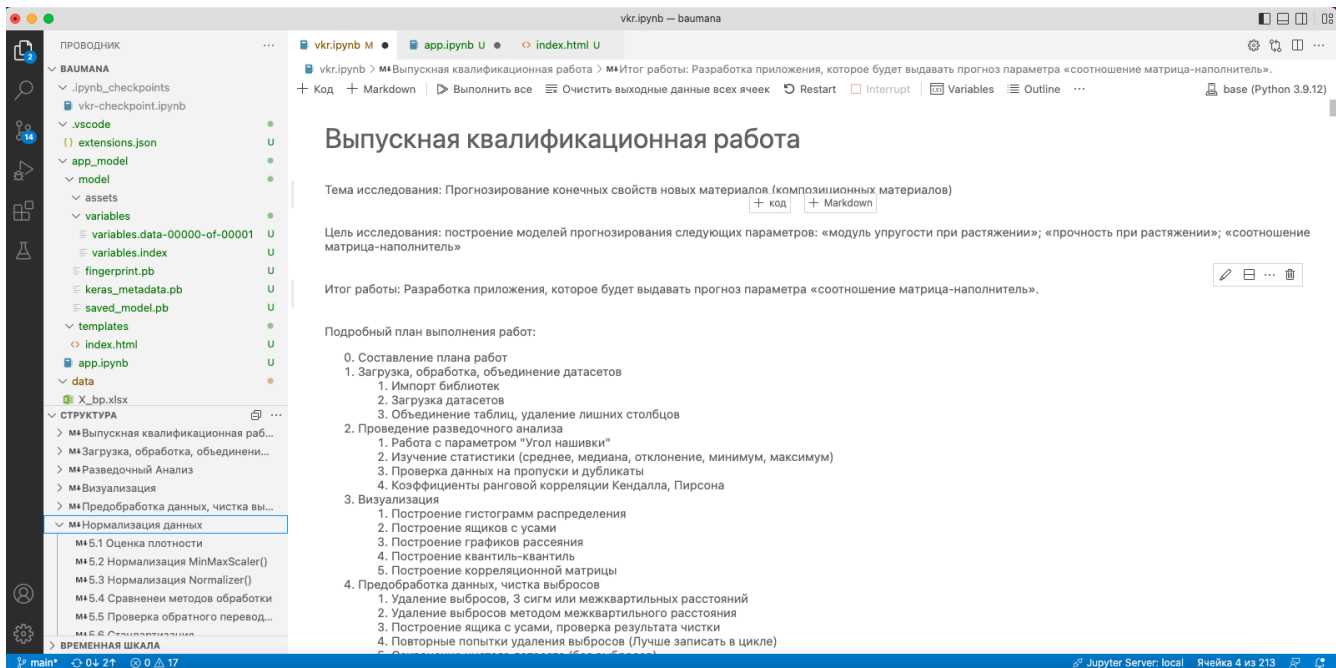


Рисунок 1 – скриншот экрана рабочей области программы VSC

В процессе выполнения работы были разработаны несколько моделей для прогноза “Модуля упругости при растяжении” и “Прочности при растяжении”, а также была создана нейронная сеть, которая предлагает соотношение «Матрицы - наполнитель». На основе нейронной сети было создано приложение на фреймворке Flask.

1. Аналитическая часть

1.1. Постановка задачи

Для достижения поставленной задачи необходимо выполнить ряд последовательных методов, а именно, провести разведочный анализ данных, нарисовать гистограммы распределения, диаграммы ящика с усами, попарные графики рассеяния точек, получить среднее, медианное значение, исключить выбросы, проверить наличие пропусков, удалить шумы и выбросы, сделать нормализацию. Обучить несколько моделей для прогноза модуля упругости при растяжении и прочности при растяжении. Написать нейронную сеть, которая

будет рекомендовать соотношение матрица-наполнитель. Разработать приложение, которое будет выдавать прогноз соотношения «матрица-наполнитель». Оценить точность модели на тренировочном и тестовом датасете.

Для исследовательской работы были представлены 2 файла: X_bp.xlsx и X_nup.xlsx состоящий из 1024 строк, 11 столбцов и 1041 строк, 4 столбцов соответственно. На первом этапе были выполнены работы по загрузке и первоначальной обработке датасетов. Были удалены лишние столбцы и произведено объединение обеих таблиц.

1.2 Загрузка датасетов

```
[297] X_bp = pd.read_excel('./data/X_bp.xlsx')#Загрузка первого датасета "X_bp.xlsx"
      X_bp.shape#Проверка размерности
      Python
      ... (1023, 11)
```

```
[298] X_nup = pd.read_excel('./data/X_nup.xlsx')#Загрузка второго датасета "X_nup.xlsx"
      X_nup.shape#Проверка размерности
      Python
      ... (1040, 4)
```

1.3 Объединение таблиц, удаление лишних столбцов

```
[299] X_bp.drop(['Unnamed: 0'], axis=1, inplace=True)#Удаление столбца
      X_bp.head()#Вывод шапки
      Python
      ...
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп, %_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0

Рисунок 2 – работы по загрузке и обработке датасетов

```
> ✓ 0.9s
#Объединение, тип объединения INNER
data = X_bp.merge(X_nup, left_index = True, right_index = True, how = 'inner')
data.head().T
```

	0	1	2	3	4
Соотношение матрица-наполнитель	1.857143	1.857143	1.857143	1.857143	2.771331
Плотность, кг/м3	2030.000000	2030.000000	2030.000000	2030.000000	2030.000000
модуль упругости, ГПа	738.736842	738.736842	738.736842	738.736842	753.000000
Количество отвердителя, м.%	30.000000	50.000000	49.900000	129.000000	111.860000
Содержание эпоксидных групп, %_2	22.267857	23.750000	33.000000	21.250000	22.267857
Температура вспышки, C_2	100.000000	284.615385	284.615385	300.000000	284.615385
Поверхностная плотность, г/м2	210.000000	210.000000	210.000000	210.000000	210.000000
Модуль упругости при растяжении, ГПа	70.000000	70.000000	70.000000	70.000000	70.000000
Прочность при растяжении, МПа	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
Потребление смолы, г/м2	220.000000	220.000000	220.000000	220.000000	220.000000
Угол нашивки, град	0.000000	0.000000	0.000000	0.000000	0.000000
Шаг нашивки	4.000000	4.000000	4.000000	5.000000	5.000000
Плотность нашивки	57.000000	60.000000	70.000000	47.000000	57.000000

```
> ✓ 0.2s
#Получили итоговой татасет размерностью:
#13 столбцов и 1023 строки, так как левое соединение.
data.shape
```

(1023, 13)

Рисунок 3 - объединение таблиц

2.1 Работа с параметром "Угол нашивки"

```
#Приведение столбца к 0 и 1
data = data.replace({'Угол нашивки, град': {0.0 : 0, 90.0 : 1}})
data.iloc[:,10] = data.iloc[:,10].astype(int)
```

[306] ✓ 0.4s

Рисунок 4 - преобразование параметра "Угол нашивки"

1.2. Описание используемых методов

Для решения поставленной задач были использованы следующие методы:

- К-ближайших соседей
- метод опорных векторов
- линейная регрессия
- градиентный бустинг
- метод случайного леса
- дерево решений

- многослойный перцептрон
- стохастический градиентный спуск
- Лассо
- Метод Grid Search

Метод К-ближайших соседей (kNN - k Nearest Neighbours) этот метод был выбран так-как он довольно универсален, метод ищет ближайшие объекты с известными значения 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 усредняет или голосует за наиболее часто встречающуюся метку.

Достоинства метода: прост в реализации; имеет низкую чувствительность к выбросам; допускает настройку нескольких параметров; позволяет делать дополнительные допущения.

Недостатки метода: замедляется с ростом объема данных; не создает правил; 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; имеет вычислительную трудоемкость.

Метод опорных векторов (Support Vector Regression) – этот бинарный линейный классификатор был выбран, потому что он хорошо работает на небольших датасетах. Данный алгоритм – это алгоритм обучения с учителем, использующихся для задач классификации и регрессионного анализа, это контролируемое обучение моделей с использованием схожих алгоритмов для анализа данных и распознавания шаблонов.

Каждый объект данных представляется как вектор (точка) в p -мерном пространстве. Он создаёт линию или гиперплоскость, которая разделяет данные на классы.

Достоинства метода: для классификации достаточно небольшого набора данных. Эффективен при большом количестве гиперпараметров. Существует возможность гибко настраивать разделяющую функцию.

Недостатки метода: неустойчивость к шуму; для больших наборов данных требуется долгое время обучения; параметры модели сложно интерпретировать.

Линейная регрессия (Linear regression) — это алгоритм машинного обучения, основанный на контролируемом обучении, рассматривающий зависимость между одной входной и выходными переменными. Она определяет зависимость переменных с помощью линии наилучшего соответствия.

Достоинства метода: быстр и прост в реализации; имеет меньшую сложность по сравнению с другими алгоритмами;

Недостатки метода: моделирует только прямые линейные зависимости; требует прямую связь между зависимыми и независимыми переменными; выбросы оказывают большое влияние.

Градиентный бустинг (Gradient Boosting) — основная идея градиентного бустинга: строятся последовательно несколько базовых классификаторов, каждый из которых как можно лучше компенсирует недостатки предыдущих. Финальный классификатор является линейной композицией этих базовых классификаторов.

Достоинства метода: новые алгоритмы учатся на ошибках предыдущих; наблюдения выбираются на основе ошибки; прост в настройке темпа обучения и применения; легко интерпретируем.

Недостатки метода: необходимо тщательно выбирать критерии остановки, иначе это может привести к переобучению; наблюдения с наибольшей ошибкой появляются чаще; менее гибок чем нейронные сети.

Случайный лес (RandomForest) — это множество решающих деревьев. Универсальный алгоритм машинного обучения с учителем, представитель ансамблевых методов. Если точность дерева решений оказалась недостаточной, можно множество моделей собрать в коллектив.

Достоинства метода: не переобучается; не требует предобработки входных данных; имеет высокую точность предсказания и высокую масштабируемость.

Недостатки метода: построение занимает много времени; сложно интерпретируемый; может недо обучаться; иногда работает хуже, чем линейные методы.

Дерево принятия решений (DecisionTreeRegressor) – это инструмент принятия решений, в котором используется древовидная структура, подобная блок-схеме, или модель решений и всех их возможных результатов, включая результаты, затраты и полезность. Алгоритм дерева решений подпадает под категорию контролируемых алгоритмов обучения. Он работает как для непрерывных, так и для категориальных выходных переменных.

Достоинства метода: создаются по понятным правилам; просты в применении и интерпретации; заполняют пропуски в данных наиболее вероятным решением.

Недостатки метода: ошибаются при классификации с большим количеством классов и небольшой обучающей выборкой; имеет затратные вычисления; ограниченное число вариантов решения проблемы.

Многослойный перцептрон (MLPRegressor) — это искусственная нейронная сеть, имеющая 3 или более слоёв перцептронов. Эти слои - один входной слой, 1 или более скрытых слоев и один выходной слой перцептронов.

Достоинства метода: построение сложных разделяющих поверхностей; легко обобщает входные данные; не требует распределения входных векторов; изучает нелинейные модели.

Недостатки метода: имеет не выпуклую функцию потерь; чувствителен к масштабированию функций.

Стохастический градиентный спуск (SGDRegressor) — это простой, но очень эффективный подход к подгонке линейных классификаторов и регрессоров под выпуклые функции потерь. Этот подход подразумевает корректировку весов нейронной сети, используя аппроксимацию градиента функционала, вычисленную только на одном случайном обучающем примере из выборки.

Достоинства метода: эффективен; прост в реализации; имеет множество возможностей для настройки кода.

Недостатки метода: требует ряд гипер-параметров; чувствителен к масштабированию функций; возможно переобучение.

Лассо регрессия (Lasso) — это линейная модель, которая оценивает разреженные коэффициенты. Это простой метод, позволяющий уменьшить сложность модели и предотвратить переопределение, которое может возникнуть в результате простой линейной регрессии. Лассо регрессия хорошо прогнозирует модели временных рядов на основе регрессии, таким как авторегрессии.

Достоинства метода: легко полностью избавляется от шумов в данных; быстро работает; способно полностью убрать признак из датасета.

Недостатки метода: часто страдает качество прогнозирования; выдает ложное срабатывание результата; не оценивает правильность формы взаимосвязи между независимой и зависимой переменными.

Метод GridSearch - этот метод используется для поиска гиперпараметров.

Гиперпараметрами называют параметры алгоритмов, значения которых устанавливаются перед запуском процесса обучения. Они и отличаются от обычных параметров, вычисляемых в процессе обучения. Гиперпараметры используются для управления процессом обучения. Способ настройки гиперпараметров состоит в том, чтобы заставить компьютер попробовать все возможные комбинации значений параметров. Для этого используем модуль GridSearchCV из библиотеки Scikit Learn.

Достоинства метода: возможность получения требуемых оптимальных гиперпараметров.

Недостатки метода: Операция является исчерпывающей. Если диапазон или число гиперпараметров велики, то вероятности могут исчисляться миллионами, и на завершение потребуется довольно много времени.

1.3. Разведочный анализ данных

Перед тем как приступить к разработке моделей необходимо произвести разведочный анализ данных.

data.describe().T

[443] ✓ 0.1s

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп, %_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, C_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.9888901

Рисунок 5 - описательная статистика

data.corr(method = 'kendall')#Коэффициенты ранговой корреляции Кендалла.

[1] ✓ 0.4s Python

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп, %_2	Температура вспышки, C_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
Соотношение матрица-наполнитель	1.000000	-0.003135	0.021247	0.001410	0.010180	-0.009480	-0.002060	-0.004157	0.011614	0.035145	-0.021395	0.022723	0.002788
Плотность, кг/м3	-0.003135	1.000000	-0.008059	-0.021963	-0.007758	-0.019947	0.037302	-0.021151	-0.047426	-0.017079	-0.051525	-0.031220	0.052935
модуль упругости, ГПа	0.021247	-0.008059	1.000000	0.022382	0.002351	0.021028	-0.000442	0.005458	0.022959	0.005169	-0.031695	-0.008305	0.049347
Количество отвердителя, м.%	0.001410	-0.021963	0.022382	1.000000	0.000010	0.059034	0.033110	-0.043140	-0.046507	-0.003677	0.024690	0.006232	0.016607
Содержание эпоксидных групп, %_2	0.010180	-0.007758	0.002351	0.000010	1.000000	-0.002170	-0.006859	0.041994	-0.013441	0.009756	0.004668	-0.004539	-0.021968
Температура вспышки, C_2	-0.009480	-0.019947	0.021028	0.059034	-0.002170	1.000000	0.017196	0.016481	-0.019106	0.035313	0.017880	0.029552	0.005268
Поверхностная плотность, г/м2	-0.002060	0.037302	-0.000442	0.033110	-0.006859	0.017196	1.000000	0.024051	-0.005099	-0.004446	0.045452	0.025514	-0.022320
Модуль упругости при растяжении, ГПа	-0.004157	-0.021151	0.005458	-0.043140	0.041994	0.016481	0.024051	1.000000	-0.006599	0.034814	0.022431	-0.010024	-0.002600

Рисунок 6 - коэффициенты ранговой корреляции Кендалла

```

data.isnull().sum()#Пропуски отсутствуют
[309] ✓ 0.2s

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

data.duplicated().sum()#Поиск дубликатов
[310] ✓ 0.4s

... 0

```

Рисунок 7 - проверка на наличие пропусков и дубликатов

В качестве инструментов разведочного анализа используется: оценка статистических характеристик датасета; гистограммы; диаграммы ящика с усами; попарные графики рассеяния точек; график «квантиль-квантиль»; тепловая карта; описательная статистика; анализ и исключение выбросов; проверка наличия пропусков и дубликатов; ранговая корреляция Кендалла и Пирсона.

Обычный статистический анализ не выявил каких либо закономерностей, далее переходим к использованию методов визуализации.

Попарные графики рассеяния точек не показывают какой-либо зависимости между данными. Зависимость между показателями нелинейная, взаимосвязь отсутствует, на графиках можно наблюдать выбросы, потому что некоторые точки располагаются далеко от общего облака, отсутствие линейной корреляции должно проявиться при построении регрессии.

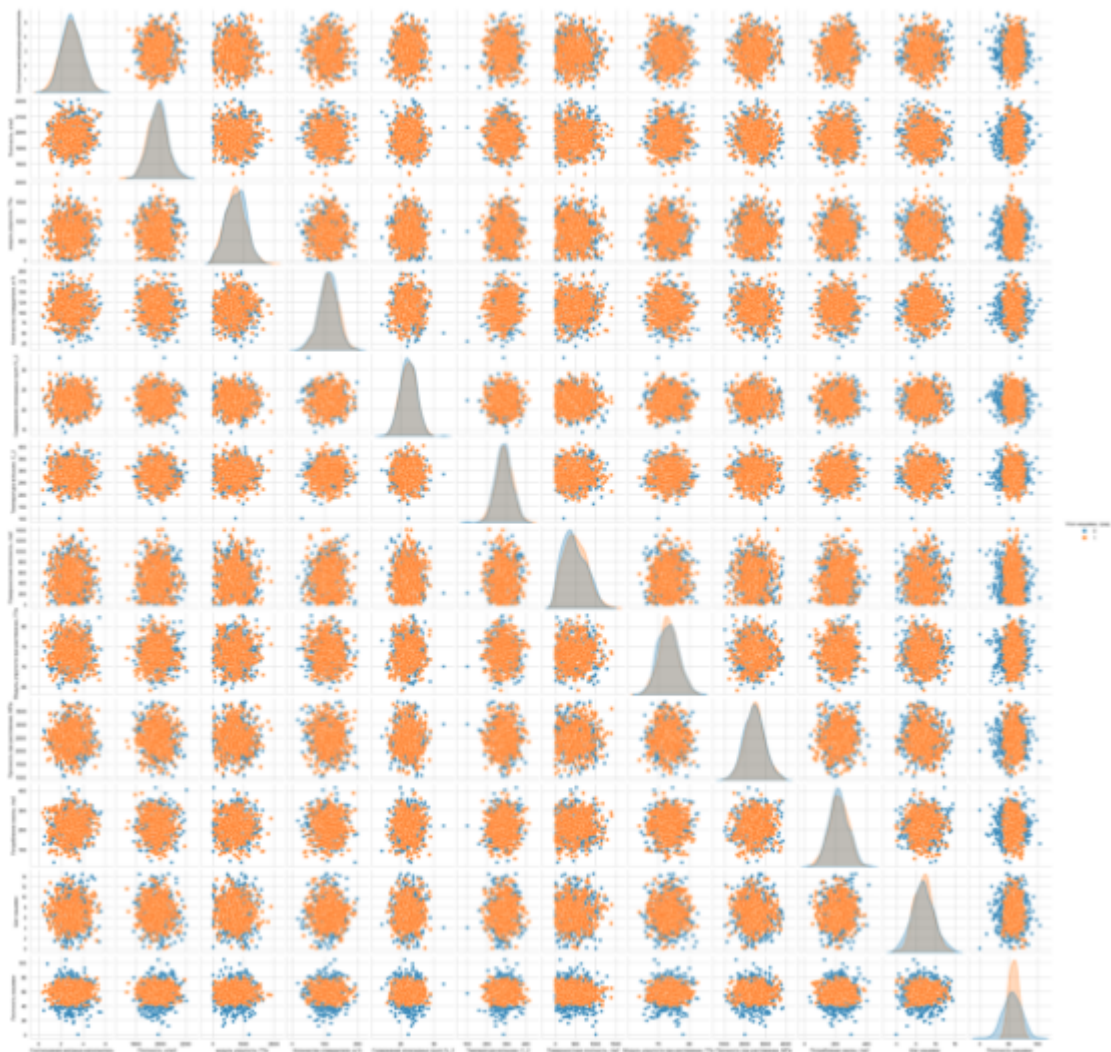


Рисунок 8 - попарные графики рассеяния точек

Методы визуализации позволяют получить наглядное представление о характерах распределений переменных. Какие именно конкретные значения или диапазоны значений исследуемой переменной встречаются наиболее часто, насколько различаются эти значения, расположено ли большинство наблюдений около среднего значения, является распределение симметричным или асимметричным и т.д. По форме распределения можно судить о природе исследуемой переменной.

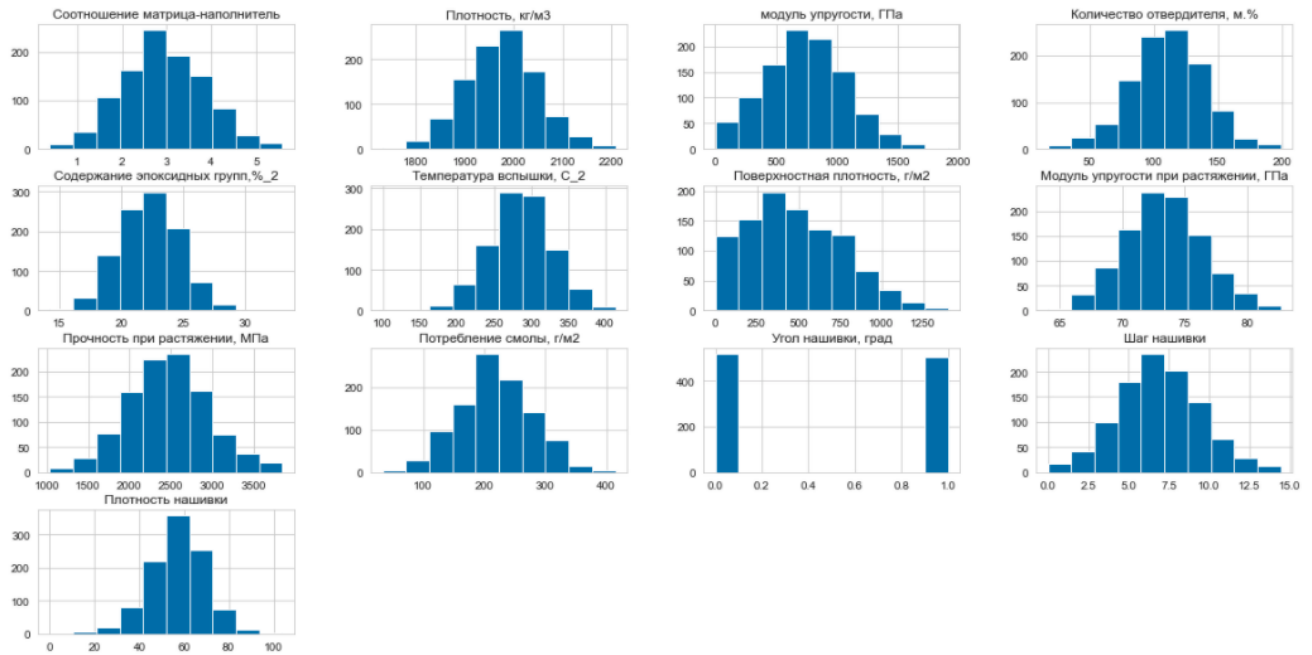


Рисунок 9 - гистограммы распределения

Ящики с усами

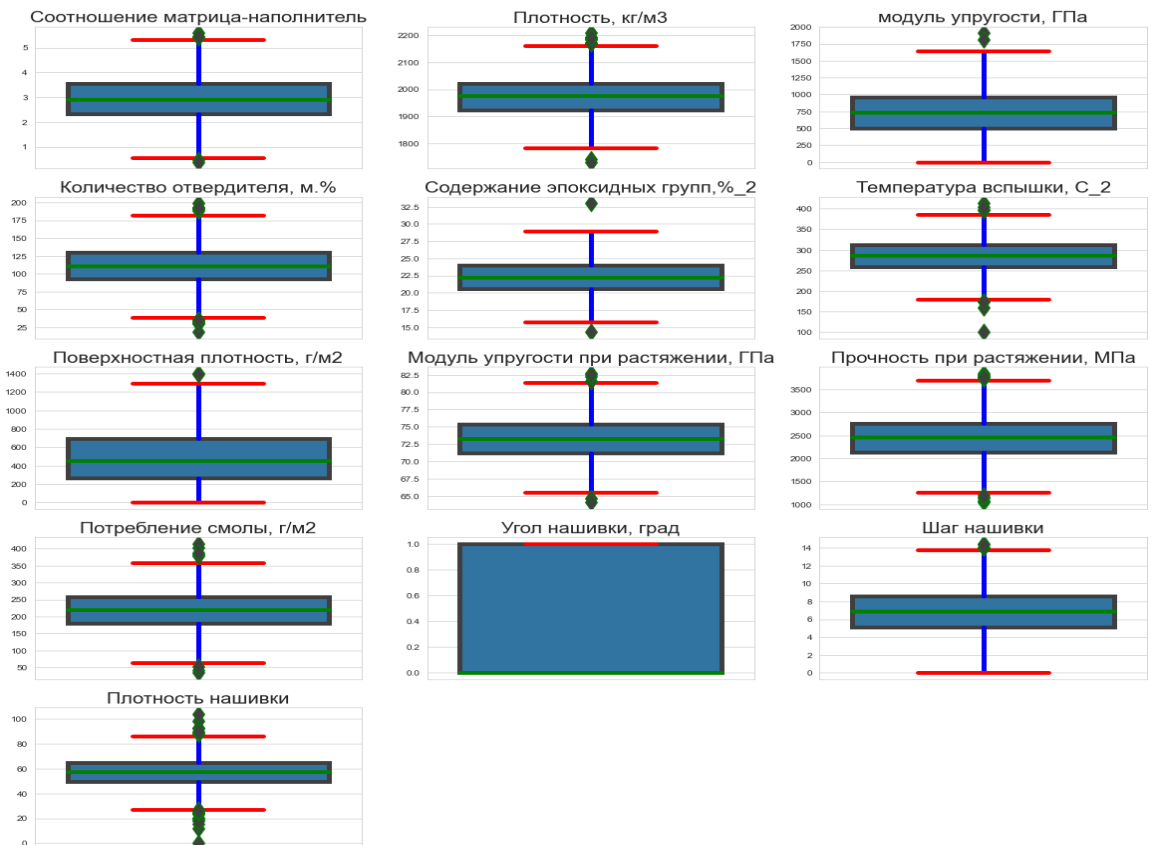


Рисунок 10 - графики "ящиков с усами"

Ящики с усами показывают выбросы: плотность, содержание эпоксидных групп, температура вспышки, плотность нашивки и т.д..

Как видно на Рисунке 9, данные стремятся к нормальному распределению практически везде, кроме угла нашивки, имеющим только 2 значения.

Алгоритмы машинного обучения чувствительны к разбросу. Соответственно, вбросы во входных данных могут исказить и ввести в заблуждение процесс обучения алгоритмов машинного обучения, поэтому в дальнейшем выбросы были удалены методом межквартильных расстояний.

Нормальное распределение, один категориальный параметр - Угол нашивки, состоит всего из двух вариантов.

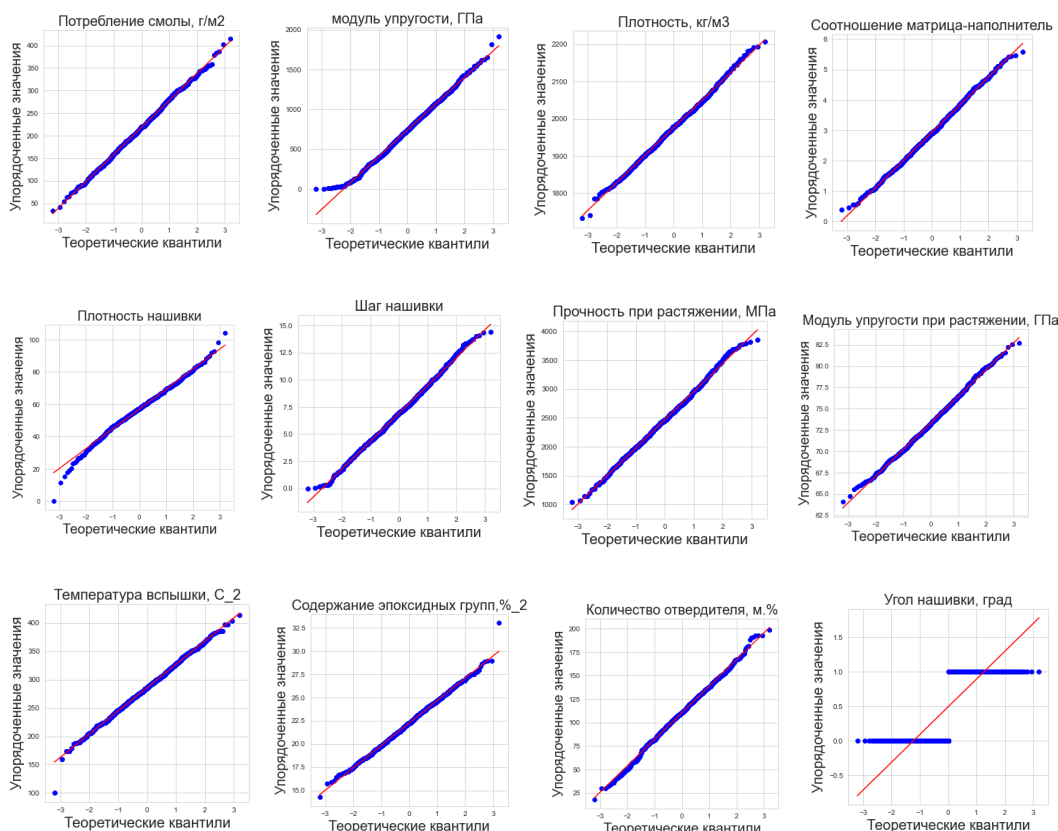


Рисунок 11 - графики «квантиль-квантиль»

Данные объединенного датасета не имеют выраженной зависимости, что подтверждает тепловая карта с матрицей корреляции и матрицы диаграмм рассеяния.

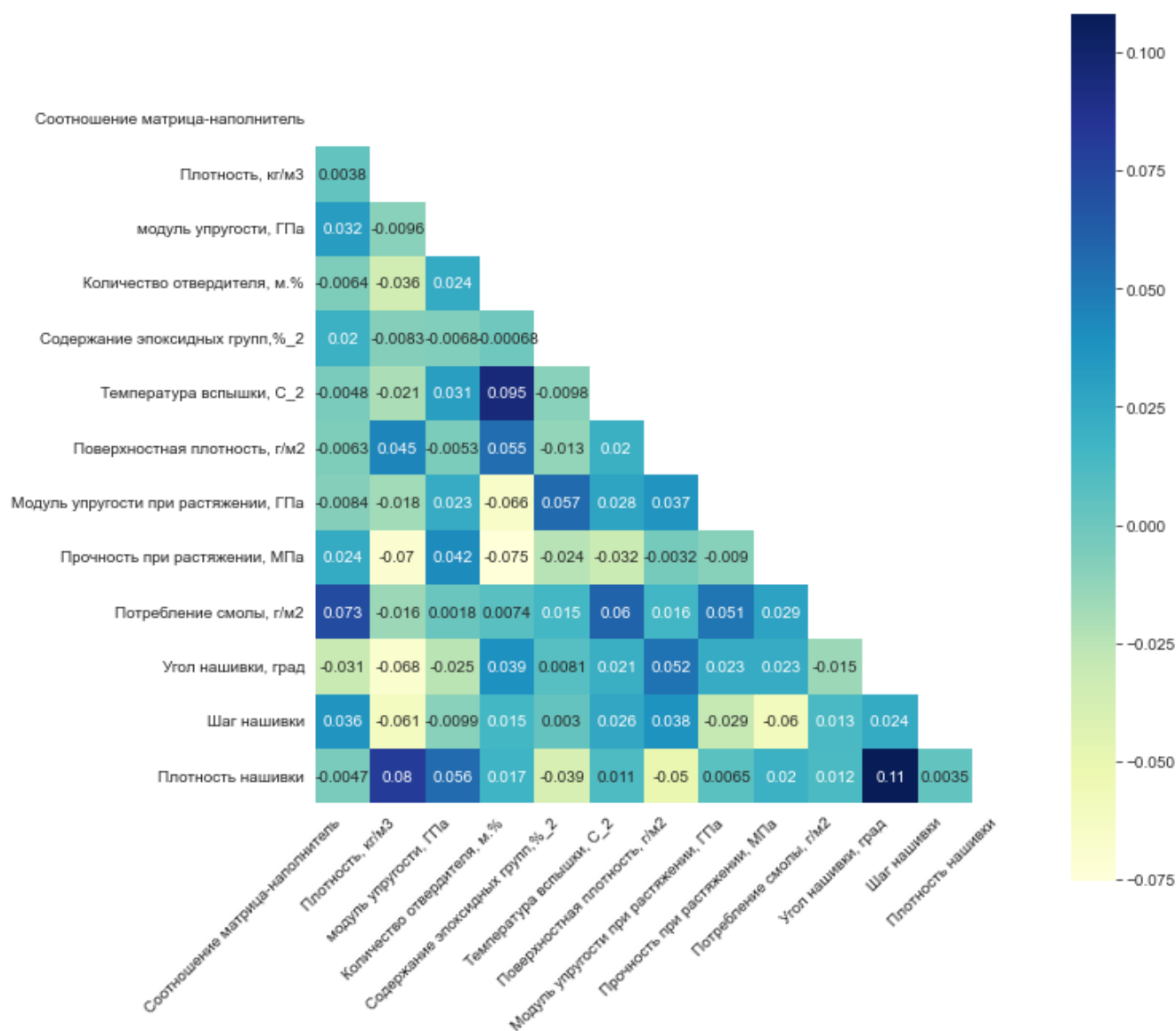


Рисунок 12 - тепловая карта с корреляцией данных

Максимальная корреляция между “Плотностью нашивки” и “Углом нашивки” - 0.11, значит нет зависимости между этими данными. Корреляция между всеми параметрами близка к 0.

2. Практическая часть

2.1. Предобработка данных

Нормализуем значения. Для этого применим `MinMaxScaler()`, затем применим `Normalizer()`. `Normalizer` даёт больше выбросов.

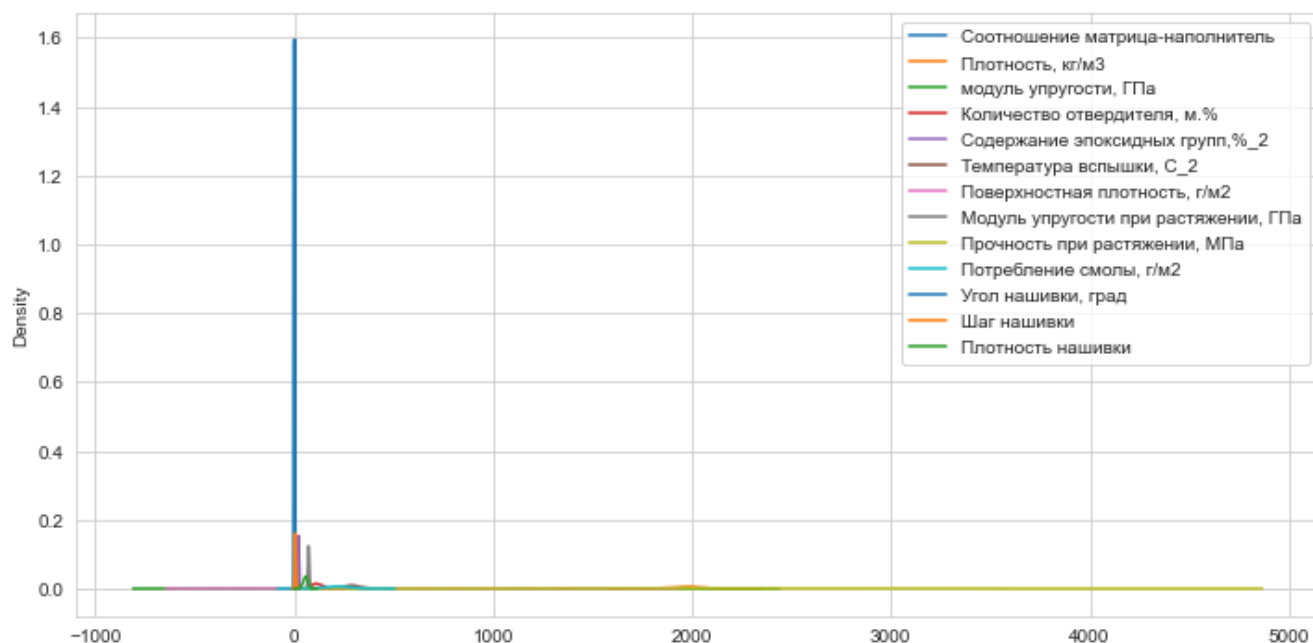


Рисунок 13 - визуализированные данные до нормализации

В процессе выбора способа масштабирования данных была проведена их стандартизация. В дальнейшем использовались данные нормализованные методом `Normalizer()`.

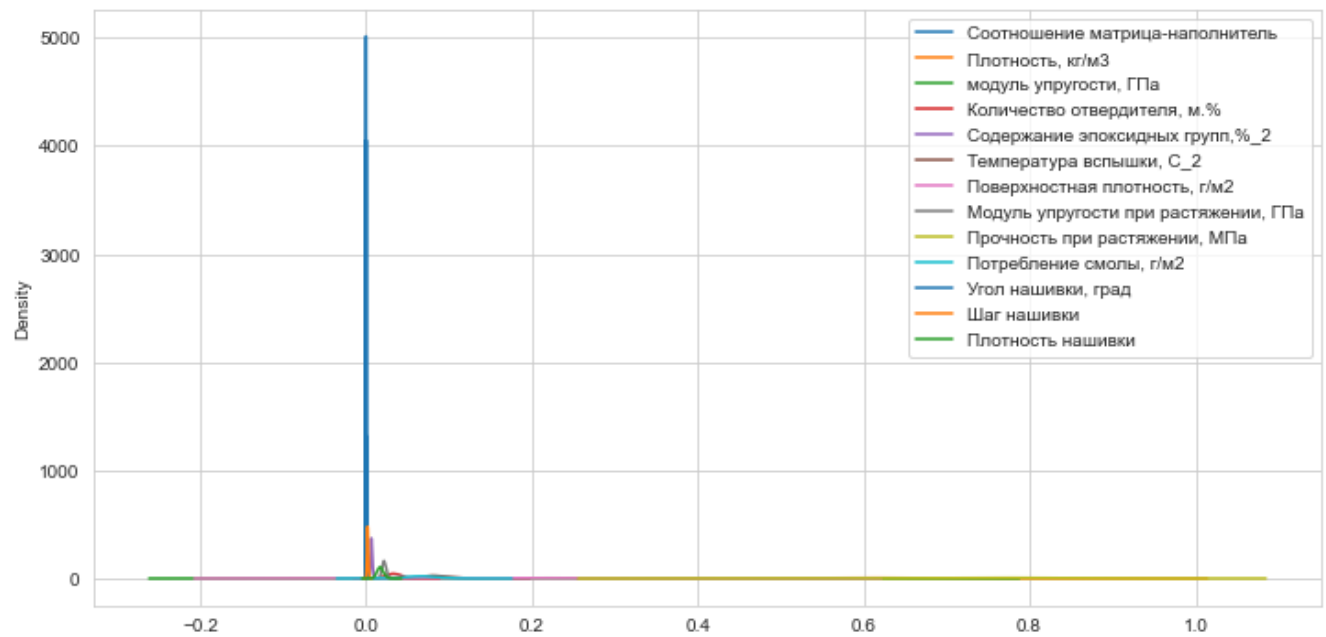


Рисунок 14 - визуализированные данные после нормализации Normalizer

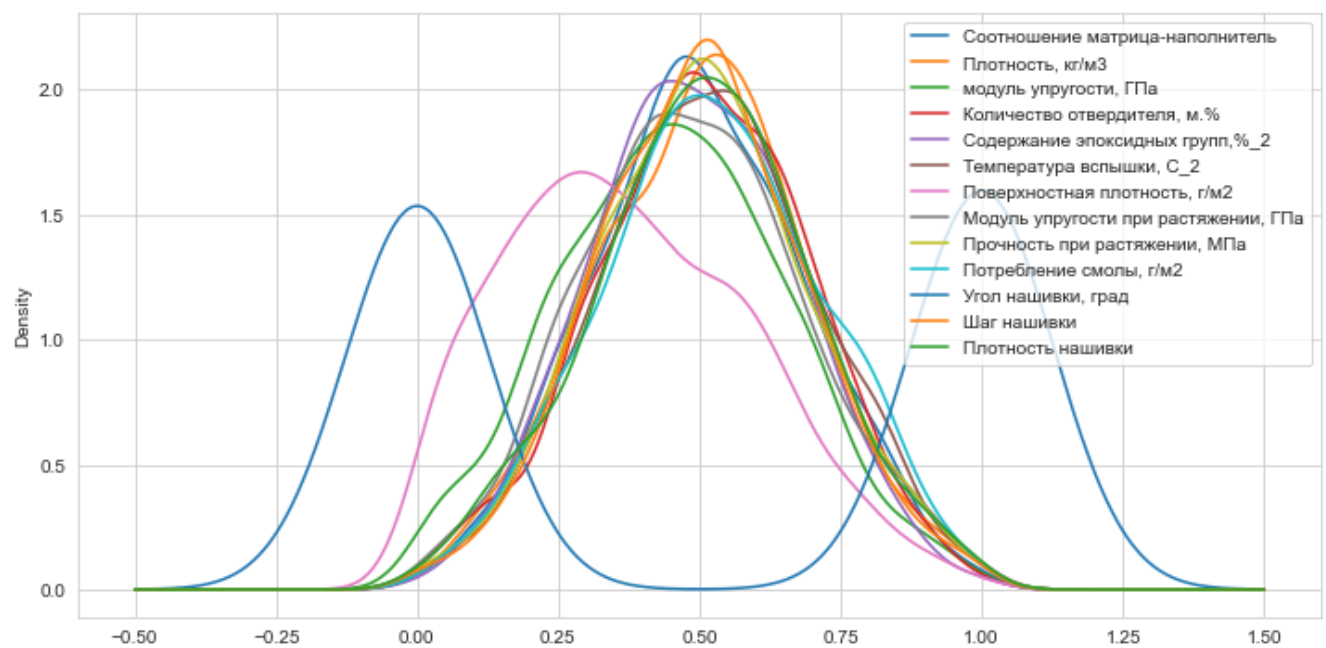


Рисунок 15 - визуализированные данные после нормализации
MinMaxScaler

	min	max		min	max
Соотношение матрица-наполнитель	0.547391	5.314144	Соотношение матрица-наполнитель	0.000163	0.001803
Плотность, кг/м3	1784.482245	2161.565216	Плотность, кг/м3	0.444650	0.824241
модуль упругости, ГПа	2.436909	1628.000000	модуль упругости, ГПа	0.000709	0.525102
Количество отвердителя, м.%	38.668500	181.828448	Количество отвердителя, м.%	0.011339	0.062919
Содержание эпоксидных групп, %_2	15.695894	28.955094	Содержание эпоксидных групп, %_2	0.004113	0.010887
Температура вспышки, C_2	179.374391	386.067992	Температура вспышки, C_2	0.049402	0.147961
Поверхностная плотность, г/м2	0.603740	1291.340115	Поверхностная плотность, г/м2	0.000230	0.414371
Модуль упругости при растяжении, ГПа	65.793845	81.203147	Модуль упругости при растяжении, ГПа	0.016108	0.030620
Прочность при растяжении, МПа	1250.392802	3660.450210	Прочность при растяжении, МПа	0.463136	0.877580
Потребление смолы, г/м2	64.524180	359.052220	Потребление смолы, г/м2	0.017544	0.122973
Угол нашивки, град	0.000000	1.000000	Угол нашивки, град	0.000000	0.000419
Шаг нашивки	0.037639	13.732404	Шаг нашивки	0.000011	0.004519
Плотность нашивки	28.382477	86.012427	Плотность нашивки	0.007195	0.030927

Рисунок 16 - минимальный и максимальные значения до и после нормализации Normalizer()

2.2. Разработка и обучение модели

Разработка и обучение моделей машинного обучения осуществлялась для двух выходных параметров: «Прочность при растяжении» и «Модуль упругости при растяжении» отдельно. Для прогноза были использованы все методы приведенные в аналитической части, а именно - К-ближайших соседей, метод опорных векторов, линейная регрессия, градиентный бустинг, случайный лес, дерево решений, многослойный перцептрон, стохастический градиентный спуск, лассо регрессия.

6.3 Метод случайного леса

```
# Метод случайного леса
rfr = RandomForestRegressor(n_estimators=15,max_depth=7, random_state=33)
rfr.fit(x_train_1, y_train_1.values)
y_pred_forest = rfr.predict(x_test_1)
mae_rfr = mean_absolute_error(y_pred_forest, y_test_1)
print('Результат:')
print("Score: {:.2f}".format(rfr.score(x_train_1, y_train_1))) # Скор для тренировочной выборки
print('RF_MAE: ', round(mean_absolute_error(y_test_1, y_pred_forest)))
print('RF_MAPE: {:.2f}'.format(mean_absolute_percentage_error(y_test_1, y_pred_forest)))
print("Test score: {:.2f}".format(rfr.score(x_test_1, y_test_1))) # Скор для тестовой выборки
```

[620] ✓ 0.2s

Рисунок 17 - пример разработки обучающей модели

Было обучено нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении. При построении модели 30% данных были выделены на тестирование модели, на остальных происходило обучение моделей. При построении моделей был проведен поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10.

```
# Проведем поиск по сетке гиперпараметров с перекрестной проверкой, количество блоков равно 10 (cv = 10), для
# Метода К ближайших соседей – K Neighbors Regressor – 5
knn = KNeighborsRegressor()
knn_params = {'n_neighbors' : range(1, 301, 2),
              'weights' : ['uniform', 'distance'],
              'algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute']}

#Запустим обучение модели. В качестве оценки модели будем использовать коэффициент детерминации (R^2)
# Если R2<0, это значит, что разработанная модель даёт прогноз даже хуже, чем простое усреднение.
gs = GridSearchCV(knn, knn_params, cv = 10, verbose = 1, n_jobs = -1, scoring = 'r2')
gs.fit(x_train_1, y_train_1)
knn_3 = gs.best_estimator_
gs.best_params_
```

[635] ✓ 32.2s

... Fitting 10 folds for each of 1200 candidates, totalling 12000 fits

{'algorithm': 'auto', 'n_neighbors': 5, 'weights': 'distance'}

Рисунок 18 - пример поиска по сетке гиперпараметров

2.3. Тестирование модели

После обучения моделей была проведена оценка точности этих моделей на обучающей и тестовых выборках.



Рисунок 19 - графики тестовых и прогнозных значений

В качестве параметра оценки модели использовалась средняя абсолютная ошибка (MAE)

	Регрессор	MAE_Test	MAE_Train	MAE_Tr%T
0	Опорные вектора	69.613765	22.543784	0.323841
1	Случайный лес	75.724864	43.711638	0.577243
2	Линейная регрессия	64.595424	62.063032	0.960796
3	Градиентный бустинг	62.238541	30.525030	0.490452
4	К ближайшие соседи	106.144166	77.720300	0.732215
5	Деревья решений	112.708075	0.000000	0.000000
6	Стохастический градиентный спуск	184.306969	174.903155	0.948977
7	Многослойный перцептрон	71.700446	70.001791	0.976309
8	Лассо	77.964348	73.821455	0.946862
9	Деревья решений_Grid	170.821054	166.991625	0.977582
10	К ближайшие соседи_Grid	102.433538	166.991625	0.000000
11	Случайный лес_Grid	68.262310	27.349716	0.160107

Рисунок 20 - сравнение параметров прогноза прочности на растяжение

	Регрессор	MAE_Test	MAE_Train	MAE_Tr%T
0	Опорные вектора	0.625399	0.468715	0.749466
1	Случайный лес	2.546952	1.824824	0.716474
2	Линейная регрессия	2.462732	2.422902	0.983827
3	Градиентный бустинг	2.646718	1.631628	0.616472
4	К ближайшие соседи	2.608872	2.154926	0.825999
5	Деревья решений	3.641094	0.000000	0.000000
6	Стохастический градиентный спуск	2.486491	2.460393	0.989504
7	Многослойный перцептрон	2.796519	2.845711	1.017590
8	Лассо	2.461084	2.437003	0.990215
9	Деревья решений_Grid	2.458037	2.375518	0.966429
10	К ближайшие соседи_Grid	2.560575	2.375518	0.000001
11	Случайный лес_Grid	2.531717	1.042790	0.424237

Рисунок 21 - сравнение параметров прогноза модуля упругости при растяжение

При разработке моделей были выполнены следующие этапы:

- разделение нормализованных данных на обучающую и тестовую выборки (в соотношении 70 на 30);
- проверка моделей при стандартных значениях;
- сравнение моделей по метрике MAE;
- поиск сетки гиперпараметров, по которым будет происходить оптимизация модели. В качестве параметра оценки выбран коэффициент детерминации (R2);
- оптимизация подбора гиперпараметров модели с помощью выбора по сетке и перекрестной проверки;
- подстановка оптимальных гиперпараметров в модель и обучение модели на тренировочных данных;
- оценка и сравнение со стандартными значениями.

Модель после настройки гиперпараметров практически не изменилась.

```

pipe = Pipeline([('preprocessing', StandardScaler()), ('regressor', SVR())])
param_grid = [
    {'regressor': [SVR()], 'preprocessing': [StandardScaler(), MinMaxScaler(), None],
     'regressor__gamma': [0.001, 0.01, 0.1, 1, 10, 100],
     'regressor__C': [0.001, 0.01, 0.1, 1, 10, 100]},
    {'regressor': [RandomForestRegressor(n_estimators = 100)], 'preprocessing': [StandardScaler(), MinMaxScaler(), None]},
    {'regressor': [LinearRegression()], 'preprocessing': [StandardScaler(), MinMaxScaler(), None]},
    {'regressor': [GradientBoostingRegressor()], 'preprocessing': [StandardScaler(), MinMaxScaler(), None]},
    {'regressor': [KNeighborsRegressor()], 'preprocessing': [StandardScaler(), MinMaxScaler(), None]},
    {'regressor': [DecisionTreeRegressor()], 'preprocessing': [StandardScaler(), MinMaxScaler(), None]},
    {'regressor': [SGDRegressor()], 'preprocessing': [StandardScaler(), MinMaxScaler(), None]},
    {'regressor': [MLPRegressor(random_state = 1, max_iter = 500)], 'preprocessing': [StandardScaler(), MinMaxScaler(), None]},
    {'regressor': [linear_model.Lasso(alpha = 0.1)], 'preprocessing': [StandardScaler(), MinMaxScaler(), None]},]
grid = GridSearchCV(pipe, param_grid, cv = 10)
grid.fit(x_train_1, np.ravel(y_train_1))
print("Наилучшие параметры:\n{}\n".format(grid.best_params_))
print("Наилучшее значение правильности перекрестной проверки: {:.2f}".format(grid.best_score_))
print("Правильность на тестовом наборе: {:.2f}".format(grid.score(x_test_1, y_test_1)))
print("Наилучшая модель:\n{}\n".format(grid.best_estimator_))

```

[780] ✓ 1m 38.2s

Наилучшие параметры:
{'preprocessing': MinMaxScaler(), 'regressor': Lasso(alpha=0.1)}

Наилучшее значение правильности перекрестной проверки: 0.97

Правильность на тестовом наборе: 0.97

Наилучшая модель:

Pipeline(steps=[('preprocessing', MinMaxScaler()),
('regressor', Lasso(alpha=0.1))])

Рисунок 22 - пример определения наилучших параметров

2.4. Написать нейронную сеть, которая будет рекомендовать соотношение «матрица – наполнитель».

Нейронную сеть была построена с помощью класса Keras.Sequential.

```
def create_model(lyrs=[32], act='softmax', opt='SGD', dr=0.1):  
  
    seed = 7  
    np.random.seed(seed)  
    tf.random.set_seed(seed)  
  
    model = Sequential()  
    model.add(Dense(lyrs[0], input_dim=x_train.shape[1], activation=act))  
    for i in range(1, len(lyrs)):  
        model.add(Dense(lyrs[i], activation=act))  
  
    model.add(Dropout(dr))  
    model.add(Dense(3, activation='tanh')) # выходной слой  
  
    model.compile(loss='binary_crossentropy', optimizer=opt, metrics=['mae', 'accuracy'])  
  
    return model
```

✓ 0.2s

Рисунок 23 - функция создания модели нейронной сети

В процессе создания нейронной сети были определены параметры, найдены оптимальные параметры. С помощью KerasClassifier были определены наилучшие параметры для построения окончательной модели.


```
#построение окончательной модели
model = create_model(layers=[128, 64, 16, 3], dr=0.05)
print(model.summary())
```

✓ 0.3s

Model: "sequential_592"

Layer (type)	Output Shape	Param #
=====		
dense_1519 (Dense)	(None, 128)	1664
dense_1520 (Dense)	(None, 64)	8256
dense_1521 (Dense)	(None, 16)	1040
dense_1522 (Dense)	(None, 3)	51
dropout_587 (Dropout)	(None, 3)	0
dense_1523 (Dense)	(None, 3)	12
=====		
Total params: 11,023		
Trainable params: 11,023		
Non-trainable params: 0		

Рисунок 24 - окончательная модель нейросети

Далее было проведено обучение и оценка модели, построены необходимые графики.

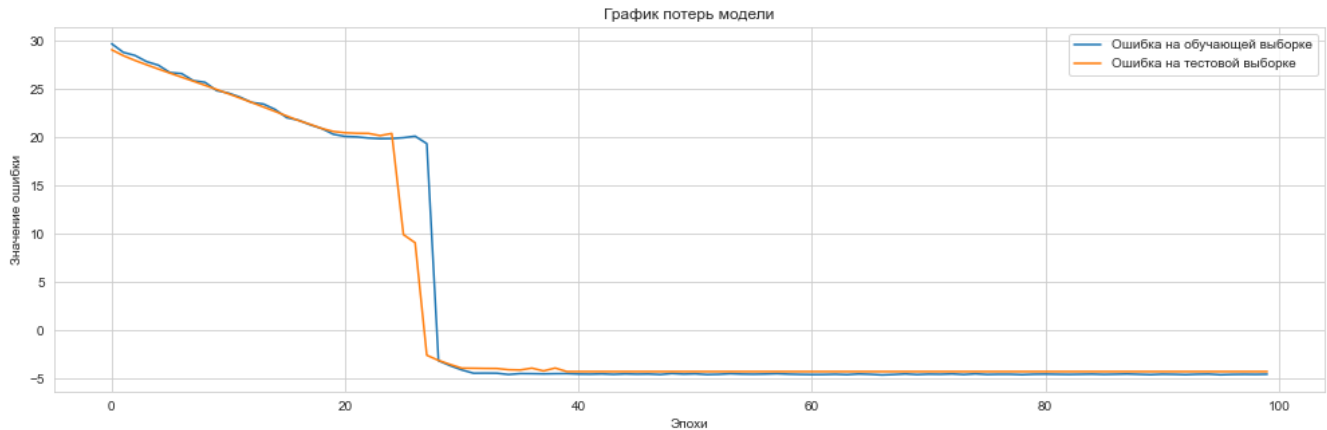


Рисунок 25 - график потерь

Потери на диаграмме в обоих наборах быстро уменьшаются в течение первых тридцати эпох. Для тестового набора потери уменьшаются с той же скоростью, что и для обучающего набора. Это означает, что полученная модель нам не подходит.



Рисунок 26 - тестовые и прогнозные значения модели

Далее в поисках более приемлемого результата была создана следующая модель глубокого обучения с другой архитектурой.

```

# Сконфигурируем другую модель, зададим слои
model1 = Sequential([
    x_train_n,
    Dense(128, activation='relu'),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(16, activation='relu'),
    Dense(1)
])

model1.compile(optimizer = tf.keras.optimizers.Adam(0.001), loss = 'mean_squared_error', metrics = [tf.keras.metrics.RootMeanSquaredError()])
# Посмотрим на архитектуру модели

model1.summary()

```

✓ 0.2s

Model: "sequential_593"

Рисунок 27 - пример создания второй модели нейросети

После обучения модели были построены необходимые графики.

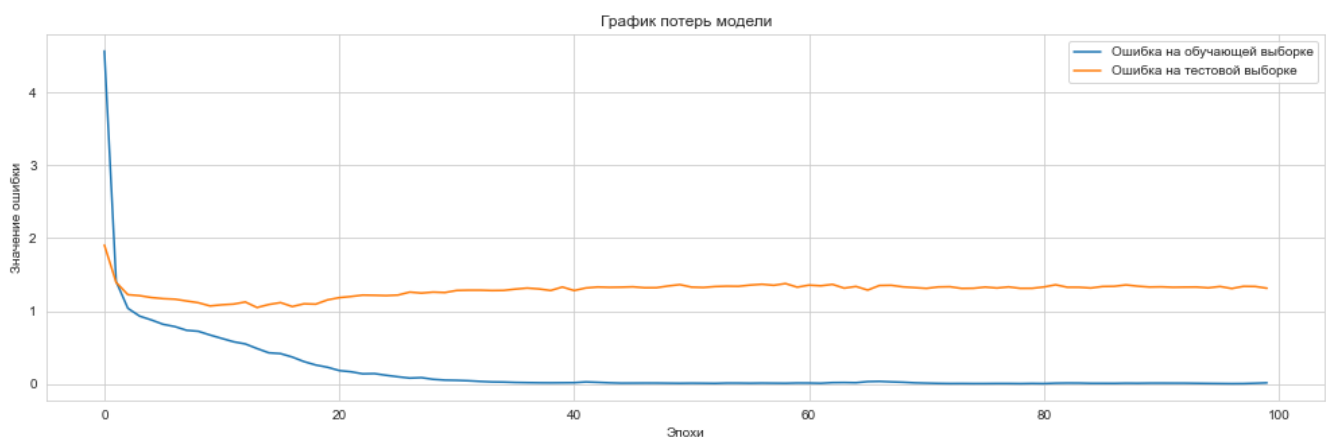


Рисунок 28 - график потерь второй модели

Теперь на диаграмме, потери в обучающем наборе быстро уменьшаются в течение первых эпох. Для тестового набора потери не уменьшаются с той же скоростью, что и для обучающего набора. Это означает, что данная модель больше подходит для того чтобы двигаться дальше.

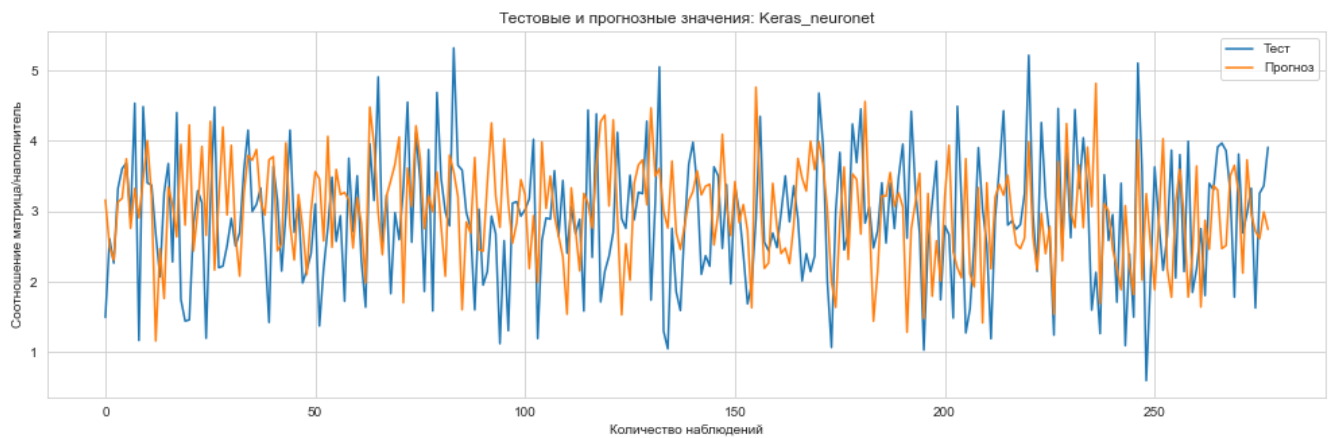


Рисунок 29 - тестовые и прогнозные значения второй модели

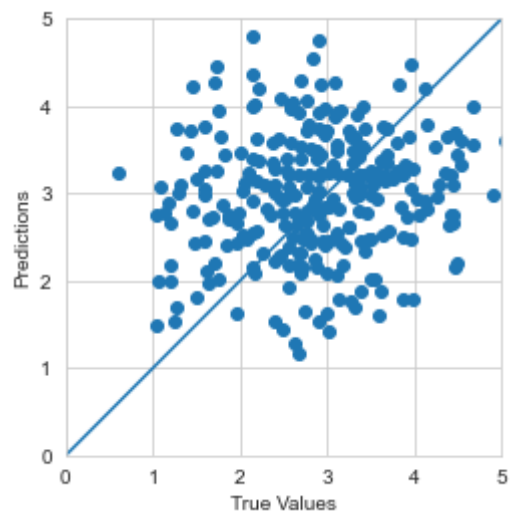


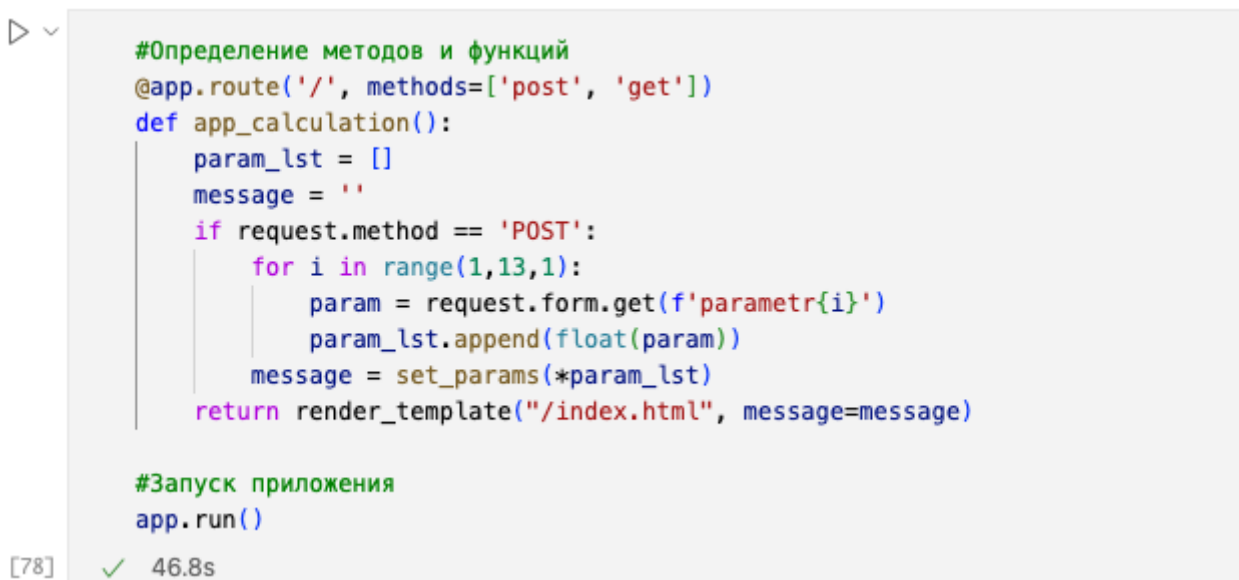
Рисунок 30- график прогнозных и реальных значений второй модели

2.5. Разработка приложения

На данном этапе было разработано графическое приложение с использованием фреймворка Flask.

Приложение состоит из следующих файлов и директорий:

- app.ipynb (Файл проекта приложения)
- templates (Директория с шаблоном приложения в формате HTML)
- model (Директория с сохраненной моделью нейронной сети)



```
#Определение методов и функций
@app.route('/', methods=['post', 'get'])
def app_calculation():
    param_lst = []
    message = ''
    if request.method == 'POST':
        for i in range(1,13,1):
            param = request.form.get(f'parametr{i}')
            param_lst.append(float(param))
            message = set_params(*param_lst)
        return render_template("/index.html", message=message)

#Запуск приложения
app.run()
```

[78] ✓ 46.8s

Рисунок 31 - пример кода для запуска приложения

Использовать данное приложение очень просто, после его запуска в исполняемой среде Python необходимо перейти по следующему адресу:

<http://127.0.0.1:5000/>

Далее пользователю следует ввести данные в поля открывшейся формы и нажать на кнопку “Рассчитать”.

На выходе пользователь получает результат прогноза для значения параметра «Соотношение матрица – наполнитель».

Прогнозное значение «Соотношение матрица-наполнитель»	
Плотность, кг/м3	2030
Модуль упругости, ГПа	753
Количество отвердителя, м.%	111
Содержание эпоксидных групп, %_2	22
Температура вспышки, С_2	284
Поверхностная плотность, г/м2	210
Модуль упругости при растяжении, ГПа	70
Прочность при растяжении, МПа	3000
Потребление смолы, г/м2	220
Угол нашивки, град	0
Шаг нашивки	5
Плотность нашивки	57

Результат прогноза:
2.9983335

Расчитать

Рисунок 32 - скриншот приложения

2.6. Создание удаленного репозитория и загрузка

Репозиторий был создан на github.com по адресу:
<https://github.com/mm7ru/baumana>

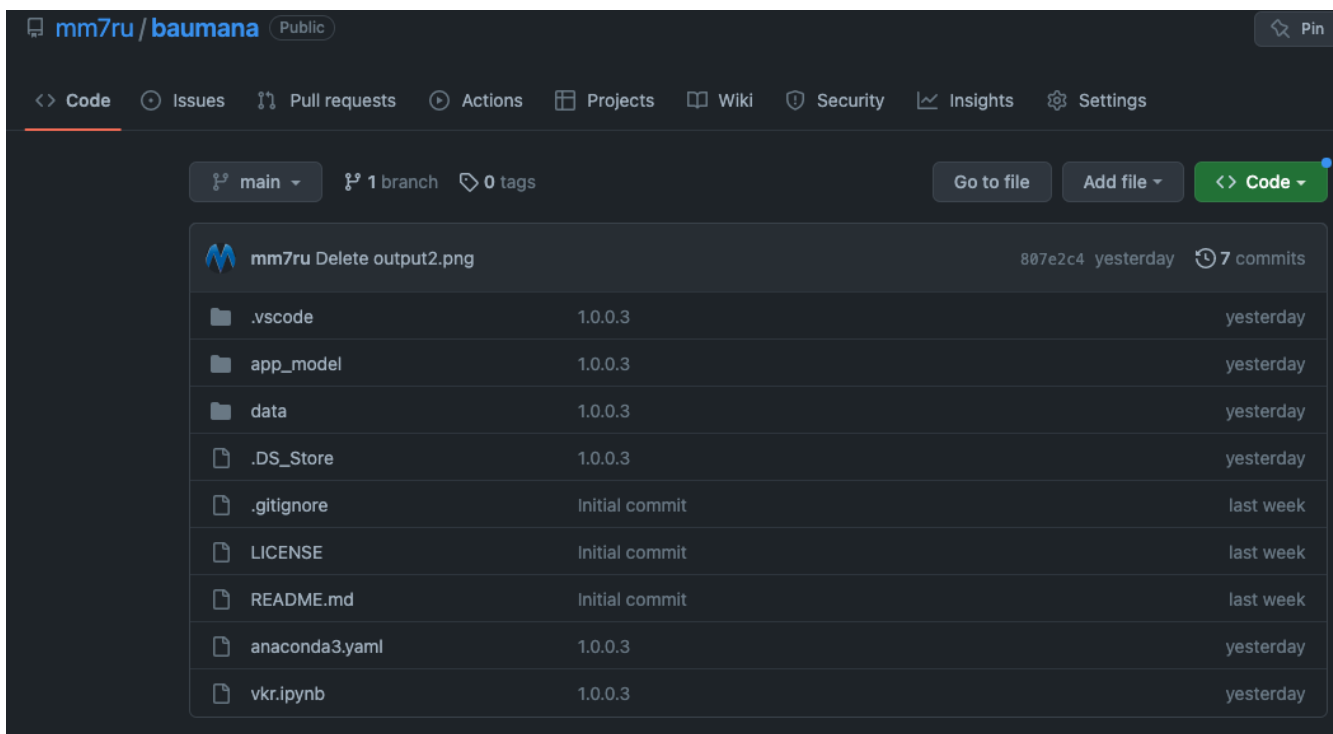


Рисунок 33 - скриншот репозитория

2.7. Заключение

Машинное обучение в задачах моделей прогнозирования – сложный процесс, требующий не только навыков программирования, но и профессиональных знаний в сфере работы с композитными материалами.

В ходе работы был задействован датасет с реальными данными, произведена его подробная опись и анализ; построено множество графиков; осуществлено разбиение данных на обучающую и тестовую выборки. Для реализации моделей машинного обучения и поиска гиперпараметров были задействованы несколько алгоритмов: метод К ближайших соседей, линейная регрессия, градиентный бустинг, деревья решений, стохастический градиентный спуск, многослойный перцептрон, опорные вектора, случайный лес и лассо регрессия.

Поиск гиперпараметров осуществлялся при помощи таких методов, как «GridSearch», были составлены отчеты, оценивающие качество проводимого обучения.

Было представлено сравнение результатов оценок работы моделей, графики и диаграммы, позволяющие оценить итоги проведенного обучения.

Обучена нейронная сеть и разработано пользовательское приложение, предсказывающее вероятный прогноз по заданным параметрам.

В процессе выполнения данной работы получилось сделать следующие выводы:

- распределение полученных данных в объединенном датасете близко к нормальному;
- коэффициенты корреляции между парами признаков стремятся к нулю;
- использованные модели не всегда позволяют получить достоверные прогнозы;
- лучшая метрика для прочности при растяжении, МПа - лассо-регрессия;
- лучшая метрика для модуля упругости при растяжении, ГПа – метод опорных векторов;

- из свойств материалов невозможно точно определить соотношение «матрица – наполнитель».

Из этого можно сделать вывод, для того чтобы определить на сколько верны полученные решения необходимо собрать дополнительные данные и провести дополнительные исследования.

2.8. Библиографический список

1. Андерсон, Карл Аналитическая культура. От сбора данных до бизнес-результатов / Карл Андерсон ; пер. с англ. Юлии Константиновой ; [науч. ред. Руслан Салахияев]. — М. : Манн, Иванов и Фербер, 2017. — 336 с
2. Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru. 2020. - 412 с.: ил.
3. Гафаров, Ф.М., Галимянов А.Ф. Искусственные нейронные сети и приложения: учеб. пособие /Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Издательство Казанского университета, 2018. – 121 с.
4. Джулли, Пал: Библиотека Keras - инструмент глубокого обучения / пер. с англ. А. А. Слинкин.- ДМК Пресс, 2017. – 249 с.
5. Грас, Джоэл. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.:
6. Д. Рутковская, М. Пилиньский, Л. Рутковский Нейронные сети, генетические алгоритмы и нечеткие системы. - М.: Горячая Линия - Телеком. - 2013. - 384 с. ISBN: 978-5-9912-0320-3
7. Д. Фостер Генеративное глубокое обучение. Творческий потенциал нейронных сетей. - СПб.: Питер. - 2020. - 336 с. - ISBN: 978-5-4461-1566-2
8. С. Николенко, А. Кадурын, Е. Архангельская Глубокое обучение. Погружение в мир нейронных сетей. - СПб.: Питер. - 2020. - 480 с. ISBN: 978-5-4461-1537-2

9. Иванов Д.А., Ситников А.И., Шляпин С.Д – Композиционные материалы: учебное пособие для вузов, 2019. 13 с.
10. Траск Эндрю. Грокаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.: ил.
11. Реутов Ю.А.: Прогнозирование свойств полимерных композиционных материалов и оценка надёжности изделий из них, Диссертация на соискание ученой степени кандидата физико-математических наук, Томск 2016.