

**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

Моисеева Мария Алексеевна

Москва, 2022

Содержание

Введение	3
1. Аналитическая часть	
1.1. Постановка задачи	5
1.2. Описание используемых методов	7
1.3. Разведочный анализ данных	11
2. Практическая часть	
2.1. Предобработка данных	12
2.2. Разработка и обучение модели	18
2.3. Тестирование модели	27
2.4. Написать нейронную сеть, которая будет рекомендовать соотношение матрица	29
2.5. Разработка приложения	35
2.6. Создание удаленного репозитория и загрузка результатов работы на него	38
Заключение	40
Список используемой литературы	41

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 — это материалы, состоящие из двух или более компонентов, нерастворимых друг с другом, с четко обозначенной границей раздела и сильным взаимодействием по всей зоне контакта. Одним из компонентов композитных материалов является непрерывная фаза, он называется матрица, в которой нерастворимые материалы помещаются в другую природу, называемую арматурой или наполнителем.

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.

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).

Актуальность: 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 соотношение «матрица - наполнитель». Было создано пользовательское веб - приложение на фреймворке Flask.

1. Аналитическая часть

1.1. Постановка задачи

Цель работы - разработать модели для прогноза модуля упругости при растяжении, прочности при растяжении и соотношения «матрица-наполнитель». Для этого нужно объединить 2 файла. Часть информации (17 строк таблицы способов компоновки композитов) не имеют соответствующих строк в таблице соотношений и свойств используемых компонентов композитов, поэтому были удалены при объединении.

Для исследовательской работы были даны 2 файла: X_br.xlsx (с данными о параметрах базальтопластика, состоящий из 1023 строки и 11 столбцов) и X_nup.xlsx (данными нашивок углепластика, состоящий из 1040 строк и 4 столбцов).

```
# загружаем первый датасет
df1 = pd.read_excel('X_br.xlsx')
df1
```

	Unnamed: 0	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	0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000	210.000000	70.000000	3000.000000	220.000000
1	1	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	210.000000	70.000000	3000.000000	220.000000
2	2	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385	210.000000	70.000000	3000.000000	220.000000
3	3	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	210.000000	70.000000	3000.000000	220.000000
4	4	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385	210.000000	70.000000	3000.000000	220.000000
...
1018	1018	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576	209.198700	73.090961	2387.292495	125.007669
1019	1019	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401	350.660830	72.920827	2360.392784	117.730099
1020	1020	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047	740.142791	74.734344	2662.906040	236.606764
1021	1021	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840	641.468152	74.042708	2071.715856	197.126067
1022	1022	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708	758.747882	74.309704	2856.328932	194.754342

1023 rows x 11 columns

Рисунок 1 - Пример начала работы с файлом X_br.xlsx

```
# загрузим второй датасет
df2 = pd.read_excel('X_nup.xlsx')
df2
```

Unnamed: 0	Угол нашивки, град	Шаг нашивки	Плотность нашивки	
0	0	0	4.000000	57.000000
1	1	0	4.000000	60.000000
2	2	0	4.000000	70.000000
3	3	0	5.000000	47.000000
4	4	0	5.000000	57.000000
...
1035	1035	90	8.088111	47.759177
1036	1036	90	7.619138	66.931932
1037	1037	90	9.800926	72.858286
1038	1038	90	10.079859	65.519479
1039	1039	90	9.021043	66.920143

1040 rows × 4 columns

Рисунок 2 - Пример начала работы с файлом X_nup.xlsx

Объединение делаем по индексу, тип объединения INNER.

```
# Объединяем по индексу, тип объединения INNER
df3 = pd.merge(df1, df2, how = 'inner', on = 'Index')
df3
```

Index	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	Угол нашивки, град
0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000	210.000000	70.000000	3000.000000	220.000000	0
1	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	210.000000	70.000000	3000.000000	220.000000	0
2	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385	210.000000	70.000000	3000.000000	220.000000	0
3	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000	210.000000	70.000000	3000.000000	220.000000	0
4	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385	210.000000	70.000000	3000.000000	220.000000	0
...
1018	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576	209.198700	73.090961	2387.292495	125.007669	90
1019	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401	350.660830	72.920827	2360.392784	117.730099	90
1020	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047	740.142791	74.734344	2662.906040	236.606764	90
1021	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840	641.468152	74.042708	2071.715856	197.126067	90
1022	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708	758.747882	74.309704	2856.328932	194.754342	90

1023 rows × 13 columns

Рисунок 3 – Объединение файлов

1.2. Описание используемых методов

Данная задача в рамках классификации категорий машинного обучения относится к машинному обучению с учителем и традиционно это задача регрессии. Цель любого алгоритма обучения с учителем — определить функцию потерь и минимизировать её, поэтому для наилучшего решения в процессе исследования были применены следующие методы:

- линейная регрессия;
- дерево решений;
- К-ближайших соседей;
- случайный лес;
- градиентный бустинг.

Линейная регрессия (Linear regression) — это алгоритм машинного обучения, основанный на контролируемом обучении, рассматривающий зависимость между одной входной и выходными переменными. Это один из самых простых и эффективных инструментов статистического моделирования. 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.

Достоинства метода: быстр и прост в реализации; легко интерпретируем; 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 огромное влияние, а границы линейны.

Дерево принятия решений (DecisionTreeRegressor) – метод автоматического анализа больших массивов данных. 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. Дерево принятия решений - эффективный инструмент интеллектуального анализа данных и предсказательной аналитики. Алгоритм дерева решений подпадает под категорию контролируемых алгоритмов обучения. Он работает как для непрерывных, так и для категориальных выходных переменных. Правила генерируются за счёт обобщения множества отдельных наблюдений (обучающих примеров), описывающих предметную область. Регрессия дерева решений отслеживает особенности объекта и обучает модель в структуре дерева прогнозированию данных в будущем для получения значимого непрерывного вывода. Дерево решений один из вариантов решения регрессионной задачи, в случае если зависимость в данных не имеет очевидной корреляции.

Достоинства метода: помогают визуализировать процесс принятия решения и сделать правильный выбор в ситуациях, когда результаты одного решения влияют на результаты следующих решений; создаются по понятным правилам; просты в применении и интерпретации; заполняют пропуски в данных наиболее вероятным решением; работают с разными переменными; выделяют наиболее важные поля для прогнозирования;

Недостатки метода: ошибаются при классификации с большим количеством классов и небольшой обучающей выборкой; имеют нестабильный процесс (изменение в одном узле может привести к построению совсем другого дерева);

имеет затратные вычисления; необходимо обращать внимание на размер; ограниченное число вариантов решения проблемы.

Метод ближайших соседей - K-ближайших соседей (kNN - k Nearest Neighbours) 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 голосует за наиболее часто встречающуюся метку (в случае задачи классификации) или усредняет метки (в случае задачи регрессии).

Достоинства метода: 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 метода: 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.

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

Достоинства метода: не переобучается; не требует предобработки входных данных; эффективно обрабатывает пропущенные данные, данные с большим числом классов и признаков; имеет высокую точность предсказания и внутреннюю

оценку обобщающей способности модели, а также высокую параллелизуемость и масштабируемость.

Недостатки метода: построение занимает много времени; сложно интерпретируемый; не обладает возможностью экстраполяции; может недо обучаться; трудоёмко прогнозируемый; иногда работает хуже, чем линейные методы.

Градиентный бустинг (Gradient Boosting) — это ансамбль деревьев решений, обученный с использованием градиентного бустинга. В основе данного алгоритма лежит итеративное обучение деревьев решений с целью минимизировать функцию потерь. 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; требуется меньше итераций, чтобы приблизиться к фактическим прогнозам; 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; слабее и менее гибок чем нейронные сети.

1.3. Разведочный анализ данных

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.

Цель разведочного анализа - получение первоначальных представлений о характерах 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); график «квантиль-квантиль»; тепловая карта (несколько вариантов); описательная статистика для каждой переменной; анализ и полное исключение выбросов; проверка наличия пропусков и дубликатов.

После обнаружения выбросов данные, значительно отличающиеся от выборки, будут полностью удалены. Для расчёта этих данных мы будем использовать методы трех сигм и межквартильного расстояния.

2. Практическая часть

2.1. Предобработка данных

Необходимо провести разведочный анализ предложенных данных. Нарисуем гистограммы распределения каждой из переменных, диаграммы ящика с усами, попарные графики рассеяния точек.

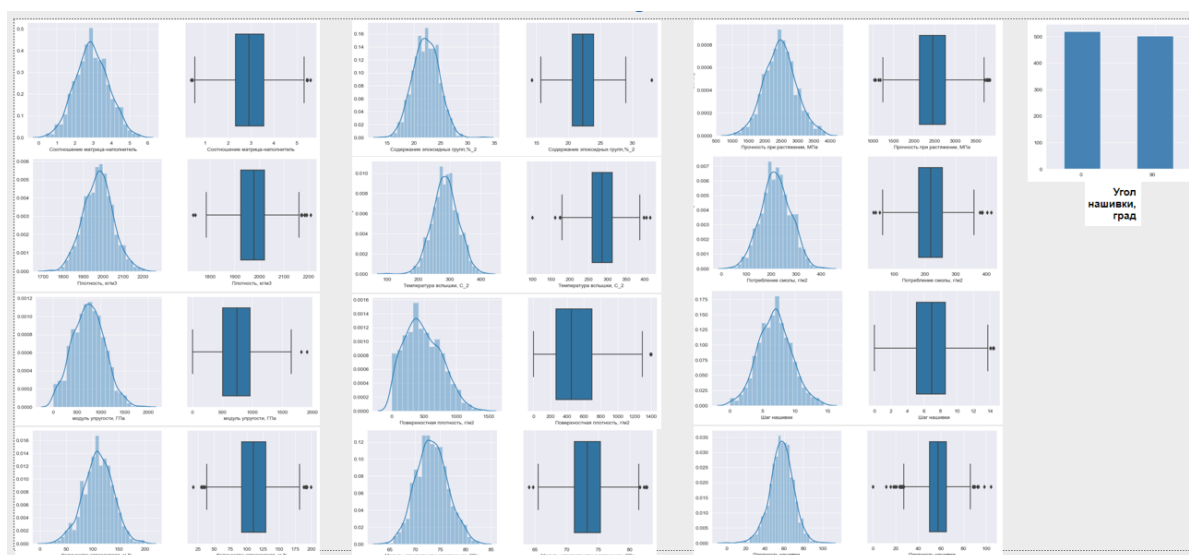


Рисунок 4 – Гистограммы распределения переменных.

Ящики с усами

Построив тепловую карту, видим, что между параметрами очень слабая корреляция. Чем ближе корреляция к 1 или -1, тем она выше, значит, есть влияние и зависимость одних параметров и других. В нашем случае самая высокая корреляция 0.11 обнаруживается между углом нашивки и плотностью нашивки.

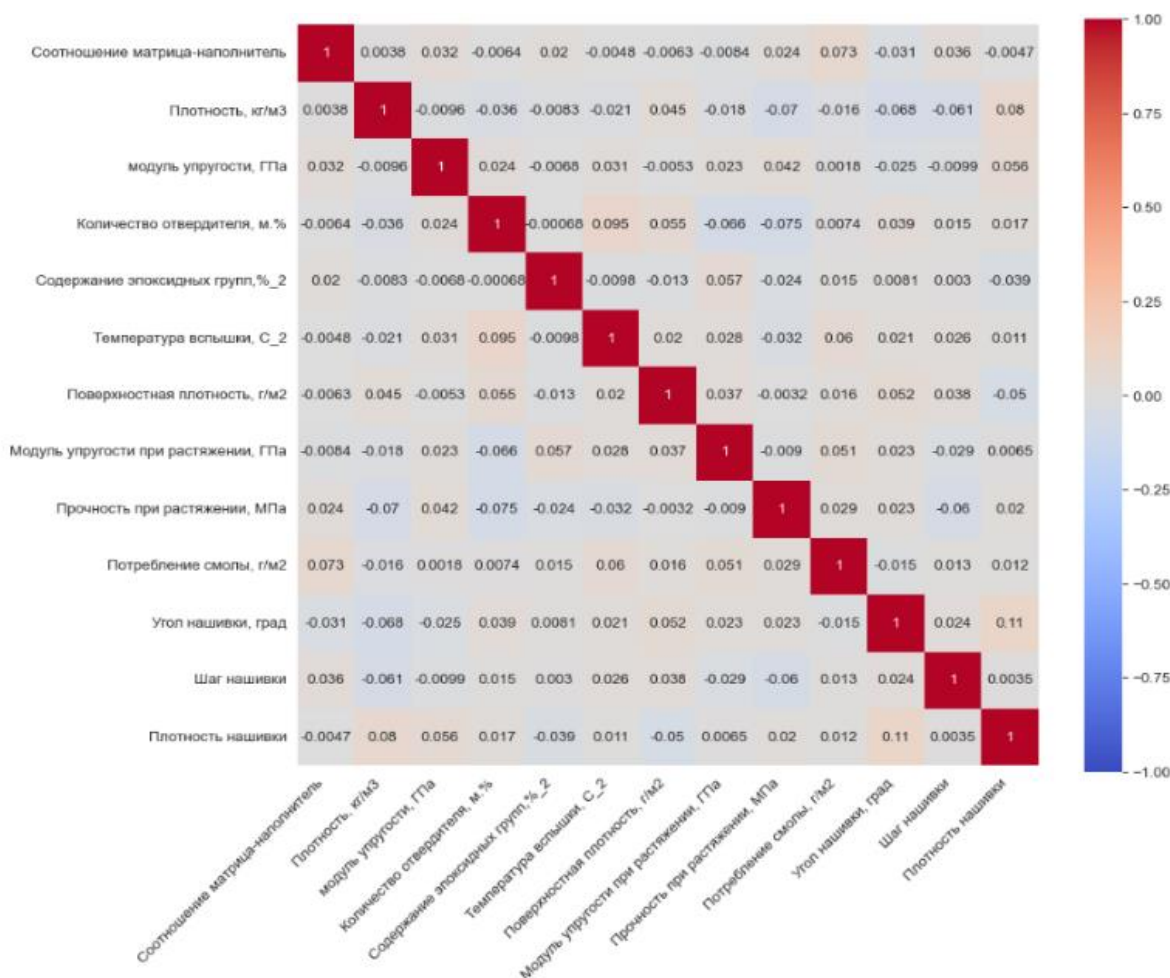


Рисунок 5 – Тепловая карта корреляции

Построим попарные графики рассеяния точек. Распределение каждой переменной показано в виде гистограммы в прямоугольниках по диагонали. Во всех остальных полях отображается диаграмма рассеяния отношений между каждой парной комбинацией переменных. На данном графике также видим отсутствие четкой зависимости между переменными.

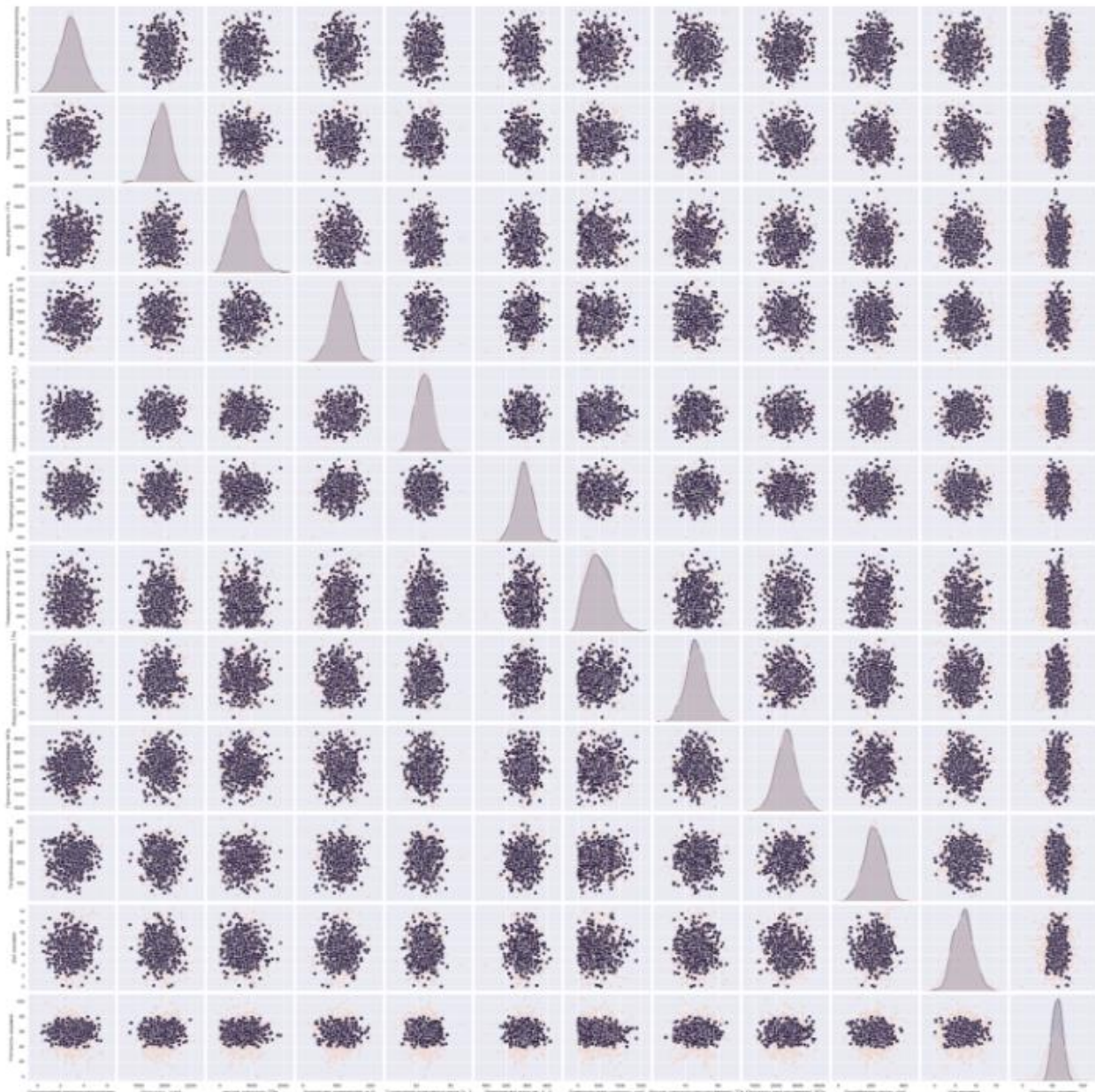


Рисунок 6 – Попарные графики рассеяния точек

Необходимо также для каждой колонки получить среднее, медианное значение. Видим, что сильных отклонений не наблюдается. Значит, выбросы практически не влияют на данные.

1. Соотношение матрица-наполнитель:	mean = 2.93 , median = 2.91
2. Плотность, кг/м3:	mean = 1975.73 , median = 1977.62
3. Модуль упругости, ГПа:	mean = 739.92 , median = 739.66
4. Количество отвердителя, м.%:	mean = 110.57 , median = 110.56
5. Содержание эпоксидных групп,%_2:	mean = 22.24 , median = 22.23
6. Температура вспышки, C_2:	mean = 285.88 , median = 285.9
7. Поверхностная плотность, г/м2:	mean = 482.73 , median = 451.86
8. Модуль упругости при растяжении, ГПа:	mean = 73.33 , median = 73.27
9. Прочность при растяжении, МПа:	mean = 2466.92 , median = 2459.52
10. Потребление смолы, г/м2:	mean = 218.42 , median = 219.2
11. Шаг нашивки:	mean = 6.9 , median = 6.92
12. Плотность нашивки:	mean = 57.15 , median = 57.34

Рисунок 7 – Среднее и медианное значения для каждой колонки

Произведем предобработку данных: удалим выбросы, сделаем нормализацию. Выбросы в данных - это значения, которые значительно отличаются от закономерностей и трендов других значений.

```
# IQR – это разница между Q3 и Q1. IQR = Q3 - Q1 IQR.
IQR = iqr(df3['модуль упругости, ГПа'])
Q3 = np.percentile(df3['модуль упругости, ГПа'], 75,
                    interpolation = 'midpoint')
upper = Q3 + 1.5 * IQR
norm_list = [x for x in df3['модуль упругости, ГПа'] if (x < upper)]
df4 = df3.loc[df3['модуль упругости, ГПа'].isin(norm_list)]
df4.shape
```

(1021, 13)

Рисунок 8 – Удаление выбросов

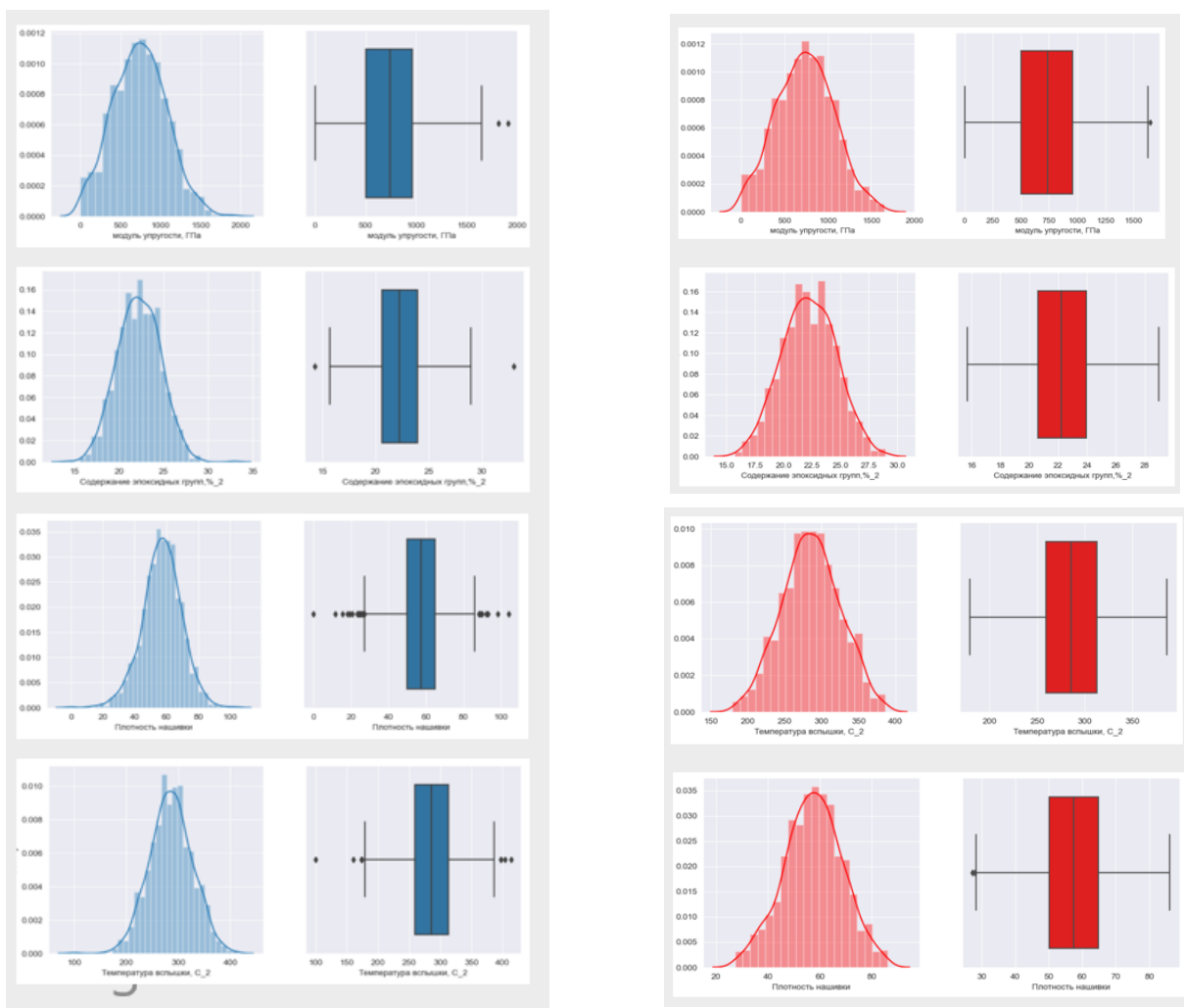


Рисунок 9 – Результаты удаления выбросов

Удалили выбросы из четырех колонок, где они были особенно заметны. В нашем датасете после удаления выбросов осталось 990 строк. Этого достаточно для дальнейшей работы. С выбросами мы удалили всего около 3 процентов данных (33 строки).

В датасете есть значения, которые исчисляются в единицах, а есть значения, исчисляемые в тысячах. Чтобы с ними было удобно работать, нужно провести нормализацию — привести различные данные в самых разных единицах

измерения и диапазонах значений к единому виду, который позволит сравнивать их между собой или использовать для расчёта схожести объектов.

```
minmax_scaler = MinMaxScaler()

# применяем нормализацию
df_norm = minmax_scaler.fit_transform(np.array(df7))
df_norm
```

```
array([[0.28213084, 0.62653324, 0.44706097, ..., 0.        , 0.27510888,
        0.55715613],
       [0.28213084, 0.62653324, 0.44706097, ..., 0.        , 0.34453943,
        0.33583998],
       [0.45785722, 0.62653324, 0.45572116, ..., 0.        , 0.34453943,
        0.50608317],
       ...,
       [0.55575038, 0.50547008, 0.25161199, ..., 1.        , 0.28629789,
        0.68704631],
       [0.63739572, 0.70384225, 0.44872381, ..., 1.        , 0.43571567,
        0.5275521 ],
       [0.65713085, 0.33328967, 0.25190326, ..., 1.        , 0.41944815,
        0.85396608]])
```

Рисунок 10 – Пример нормализации данных

	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	Угол нашивки, град	n
0	0.282131	0.626533	0.447061	0.178021	0.607435	0.509164	0.149682	0.319194	0.698235	0.488979	0.0	(
1	0.282131	0.626533	0.447061	0.613972	0.418887	0.583596	0.149682	0.319194	0.698235	0.488979	0.0	(
2	0.457857	0.626533	0.455721	0.519387	0.495653	0.509164	0.149682	0.319194	0.698235	0.488979	0.0	(
3	0.457201	0.563509	0.452685	0.519387	0.495653	0.509164	0.149682	0.319194	0.698235	0.488979	0.0	(
4	0.419084	0.374437	0.488508	0.519387	0.495653	0.509164	0.149682	0.319194	0.698235	0.488979	0.0	(
...	
985	0.361750	0.462855	0.552781	0.382158	0.333908	0.703458	0.149109	0.485125	0.480312	0.239516	1.0	(
986	0.587163	0.668737	0.268550	0.707685	0.294428	0.362087	0.250230	0.475992	0.470745	0.220404	1.0	(
987	0.555750	0.505470	0.251612	0.512067	0.623085	0.334063	0.528643	0.573346	0.578340	0.532590	1.0	(
988	0.637396	0.703842	0.448724	0.682389	0.267818	0.466417	0.458108	0.536217	0.368070	0.428909	1.0	(
989	0.657131	0.333290	0.251903	0.614984	0.888354	0.588206	0.541942	0.550550	0.647135	0.422680	1.0	(

990 rows × 13 columns

Рисунок 11 – Данные после нормализации

2.2. Разработка и обучение модели

Необходимо обучить нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении. При построении модели необходимо 30 процентов данных оставить на тестирование модели, на остальных происходит обучение моделей. При построении моделей проведем поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10.

Гиперпараметры - это все параметры, которые могут быть произвольно установлены перед началом обучения модели.

Линейная регрессия (Linear regression) — это алгоритм машинного обучения, основанный на контролируемом обучении, рассматривающий зависимость между одной входной и выходными переменными.

```
# создание и обучение модели Линейной регрессии для модуля упругости при растяжении
LR_model1 = LinearRegression()

# поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10
LR_model1_params = {
    'fit_intercept' : ['True', 'False']
}
GSCV_LR_model1 = GridSearchCV(LR_model1, LR_model1_params, n_jobs=-1, cv=10)
GSCV_LR_model1.fit(X1_train, y1_train)
print('Лучшие параметры LinearRegression для предсказания модуля упругости при растяжении: ')
GSCV_LR_model1.best_params_

Лучшие параметры LinearRegression для предсказания модуля упругости при растяжении:
{'fit_intercept': 'True'}

LR_model1_upr = GSCV_LR_model1.best_estimator_
# предсказанные значения нашей модели
y1_pred = LR_model1_upr.predict(X1_test)
```

Рисунок 12 – Построение модели Линейной регрессии

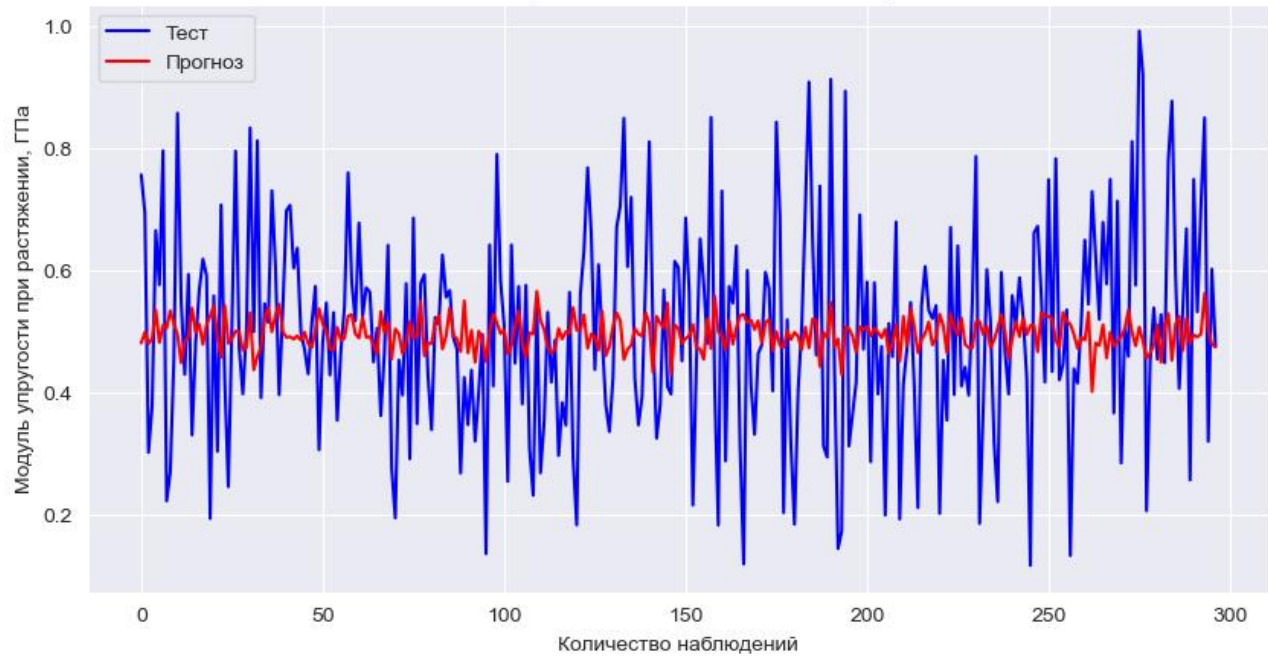


Рисунок 13 – Тестовые и прогнозные значения LinearRegression
 для Модуля упругости при растяжении

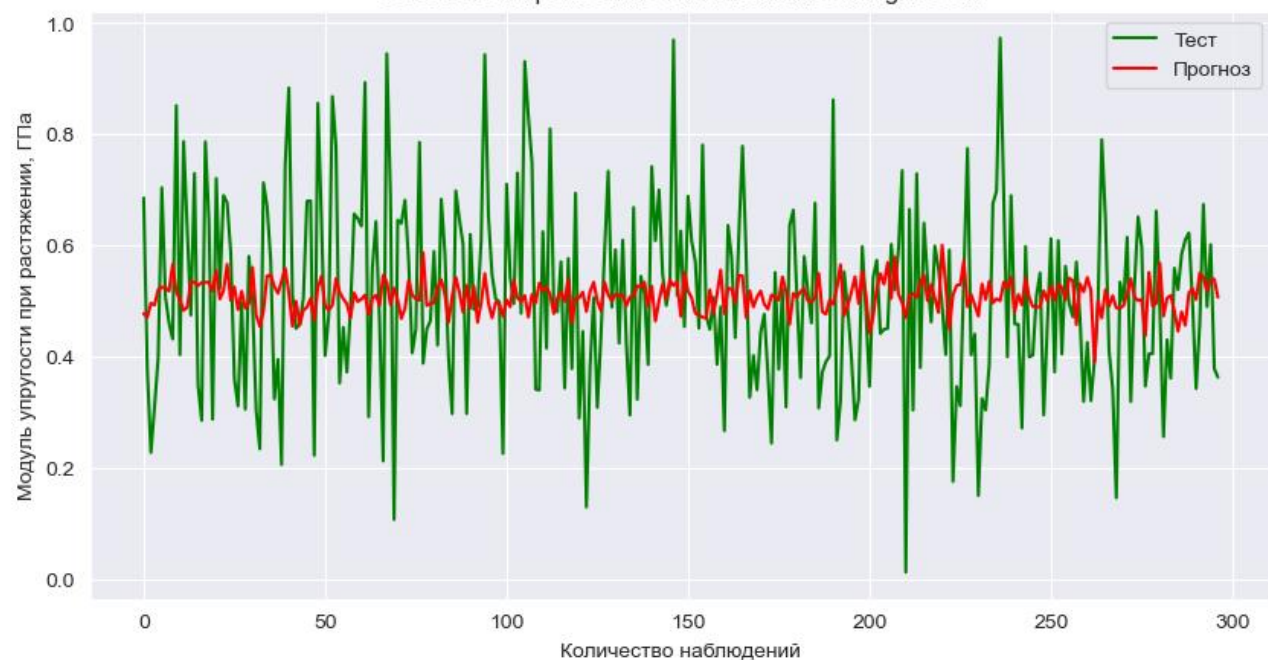


Рисунок 14 – Тестовые и прогнозные значения LinearRegression
 для Прочности при растяжении

Дерево принятия решений (DecisionTreeRegressor) – метод автоматического анализа больших массивов данных. 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.

```
# создание и обучение модели DecisionTreeRegressor для модуля упругости при растяжении
tree1 = DecisionTreeRegressor()

#поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10
tree1_params = {
    'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'],
    'splitter': ['best', 'random'],
    'max_depth': range(1, 20, 2),
    'min_samples_split': range(10, 100, 5),
    'min_samples_leaf': range(1, 200, 50),
    'max_features': ['auto', 'sqrt', 'log2']
}
GSCV_tree1 = GridSearchCV(tree1, tree1_params, n_jobs=-1, cv=10)
GSCV_tree1.fit(X1_train, y1_train)
print('Лучшие параметры DecisionTreeRegressor для предсказания модуля упругости при растяжении: ')
GSCV_tree1.best_params_

Лучшие параметры DecisionTreeRegressor для предсказания модуля упругости при растяжении:
{'criterion': 'squared_error',
 'max_depth': 17,
 'max_features': 'auto',
 'min_samples_leaf': 51,
 'min_samples_split': 90,
 'splitter': 'random'}
```

```
tree1_upr = GSCV_tree1.best_estimator_
# предсказанные значения нашей модели
y3_pred = tree1_upr.predict(X1_test)
```

Рисунок 15 – Построение модели DecisionTreeRegressor

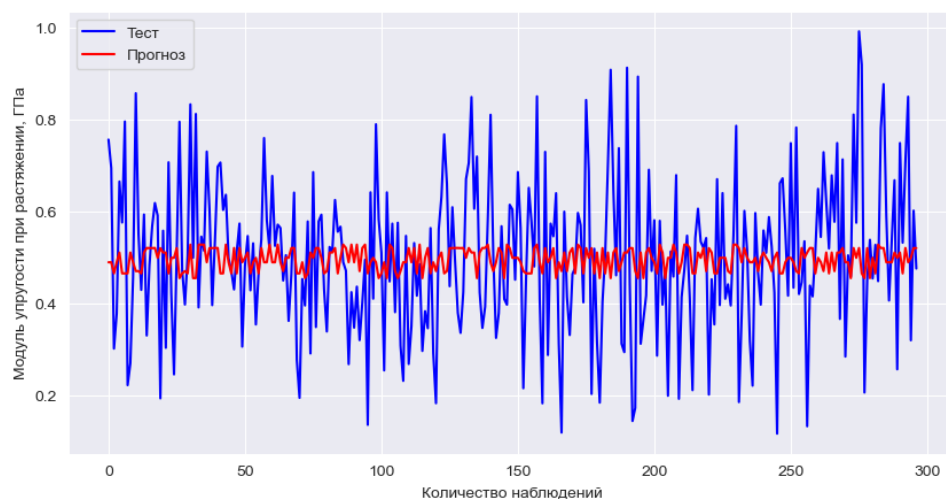


Рисунок 16 – Тестовые и прогнозные значения DecisionTreeRegressor для Модуля упругости при растяжении

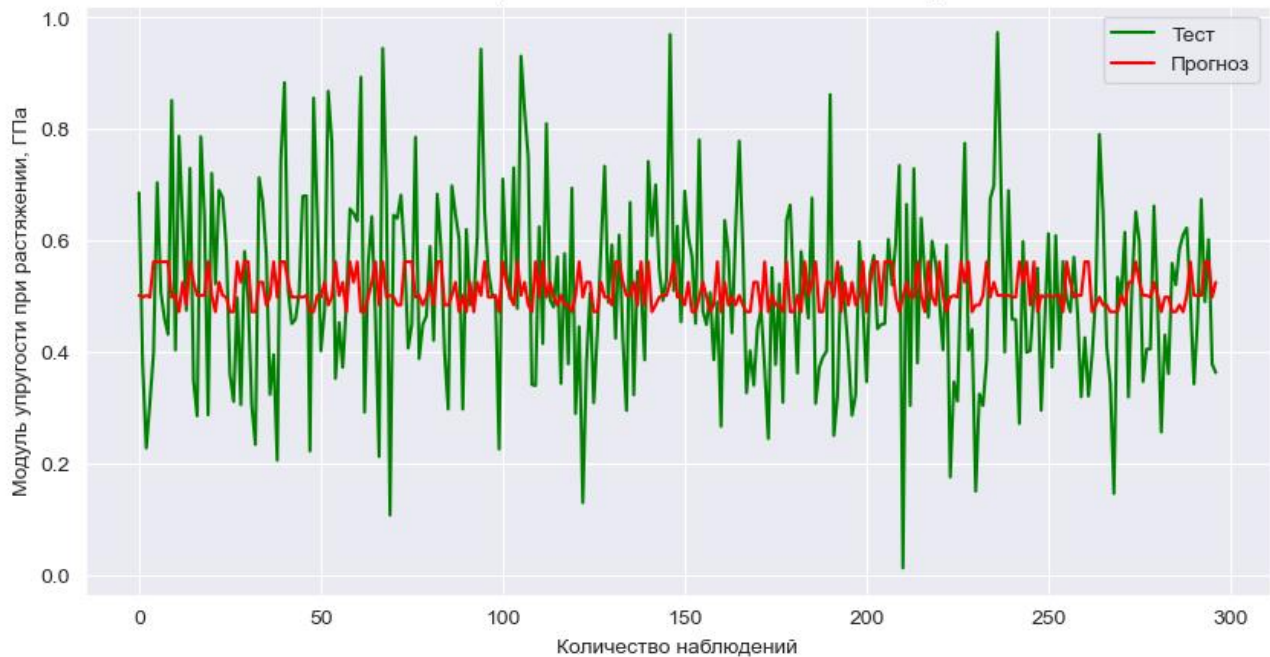


Рисунок 17 – Тестовые и прогнозные значения DecisionTreeRegressor для Прочности при растяжении

Метод ближайших соседей - К-ближайших соседей (kNN - k Nearest Neighbours) ищет ближайшие объекты с известными значения целевой переменной и основывается на хранении данных в памяти для сравнения с новыми элементами.

```
# создание и обучение модели KNeighborsRegressor для модуля упругости при растяжении
knr1 = KNeighborsRegressor()

#поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10
knr1_params = {
    'n_neighbors' : range(1, 100, 2),
    'weights' : ['uniform', 'distance'],
    'algorithm' : ['auto', 'ball_tree', 'kd_tree', 'brute']
}
GSCV_knr1 = GridSearchCV(knr1, knr1_params, n_jobs=-1, cv=10)
GSCV_knr1.fit(X1_train, y1_train)
print('Лучшие параметры KNeighborsRegressor для предсказания модуля упругости при растяжении:')
GSCV_knr1.best_params_

Лучшие параметры KNeighborsRegressor для предсказания модуля упругости при растяжении:
{'algorithm': 'auto', 'n_neighbors': 77, 'weights': 'uniform'}

knr1_upr = GSCV_knr1.best_estimator_
# предсказанные значения нашей модели
y5_pred = knr1_upr.predict(X1_test)
```

Рисунок 18 – Построение модели KNeighborsRegressor

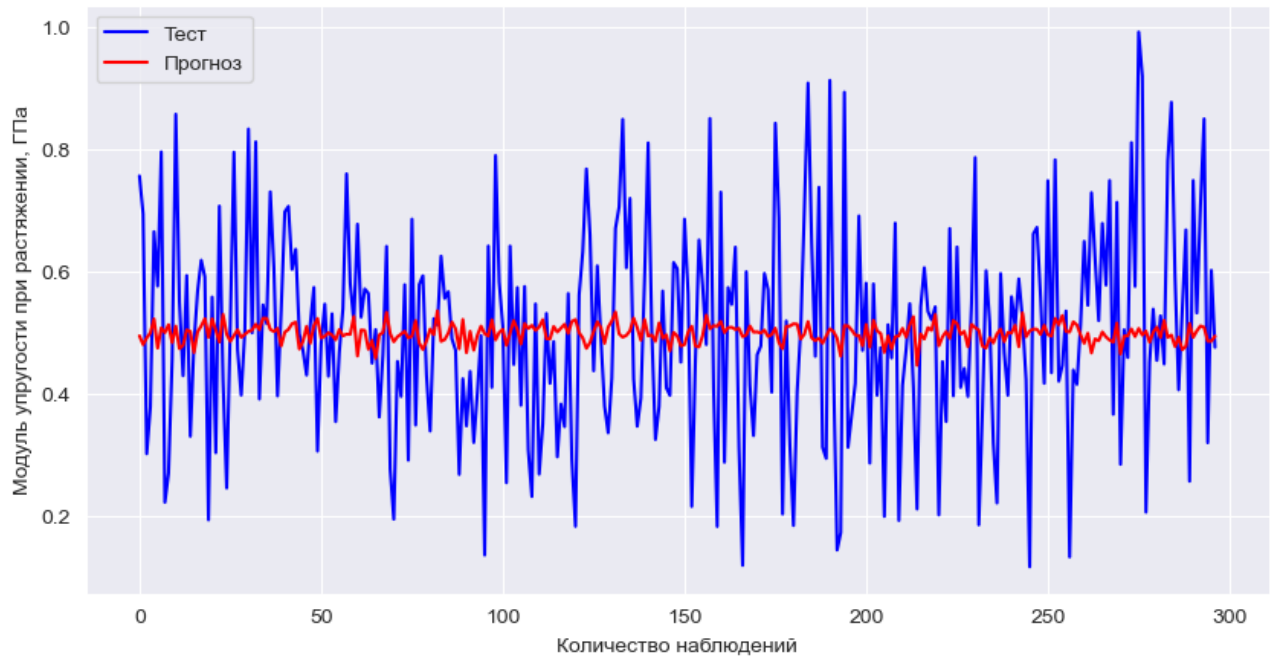


Рисунок 19 – Тестовые и прогнозные значения KNeighborsRegressor для Модуля упругости при растяжении

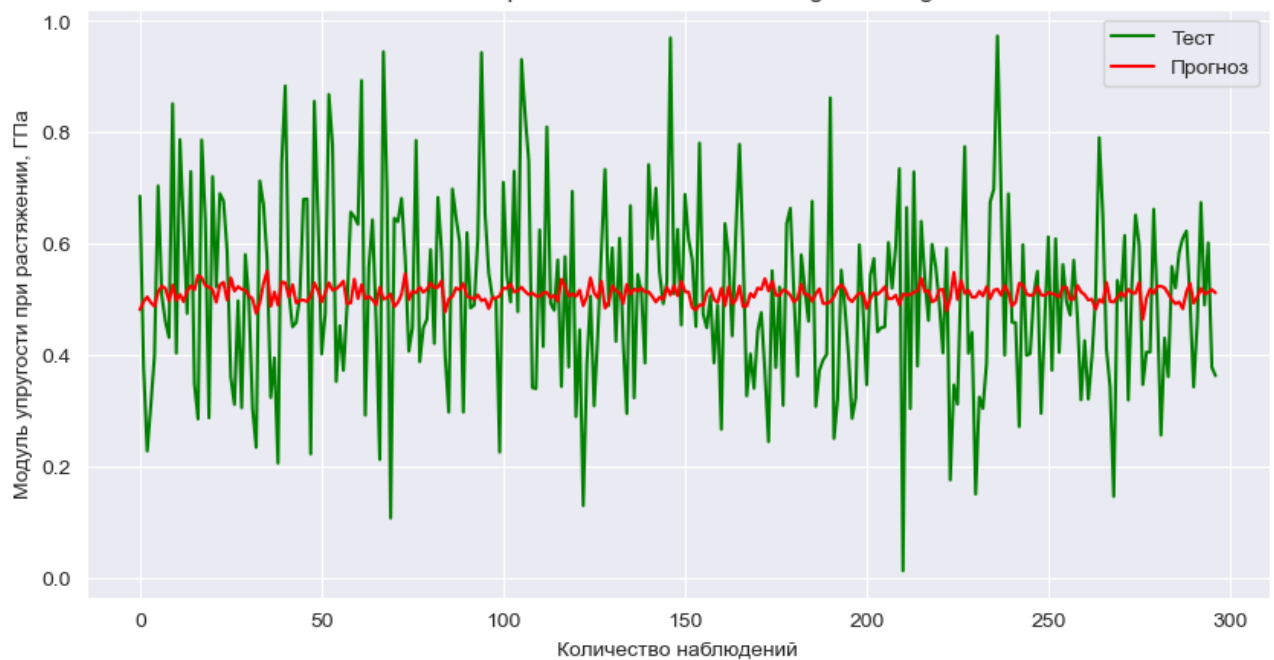


Рисунок 20 – Тестовые и прогнозные значения KNeighborsRegressor для Прочности при растяжении

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

```
# создание и обучение модели Random Forest Regressor для модуля упругости при растяжении
rfr1 = RandomForestRegressor()

#поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10
rfr1_params = {
    'n_estimators' : range(1, 500, 10),
    'criterion' : ['squared_error', 'absolute_error', 'poisson'],
    'max_depth' : range(1, 8),
    'min_samples_split' : range(10, 50, 5),
    'min_samples_leaf' : range(2, 9),
    'bootstrap' : ['True', 'False'],
    'oob_score' : ['True', 'False'],
    'warm_start' : ['True', 'False']
}
GSCV_rfr1 = RandomizedSearchCV(rfr1, rfr1_params, n_jobs=-1, cv=10)
GSCV_rfr1.fit(X1_train, np.ravel(y1_train))
print('Лучшие параметры RandomForestRegressor для предсказания модуля упругости при растяжении: ')
GSCV_rfr1.best_params_

Лучшие параметры RandomForestRegressor для предсказания модуля упругости при растяжении:

{'warm_start': 'False',
 'oob_score': 'False',
 'n_estimators': 451,
 'min_samples_split': 30,
 'min_samples_leaf': 2,
 'max_depth': 2,
 'criterion': 'absolute_error',
 'bootstrap': 'False'}

rfr1_upr = GSCV_rfr1.best_estimator_
# предсказанные значения нашей модели
y7_pred = rfr1_upr.predict(X1_test)
```

Рисунок 21 – Построение модели RandomForestRegressor

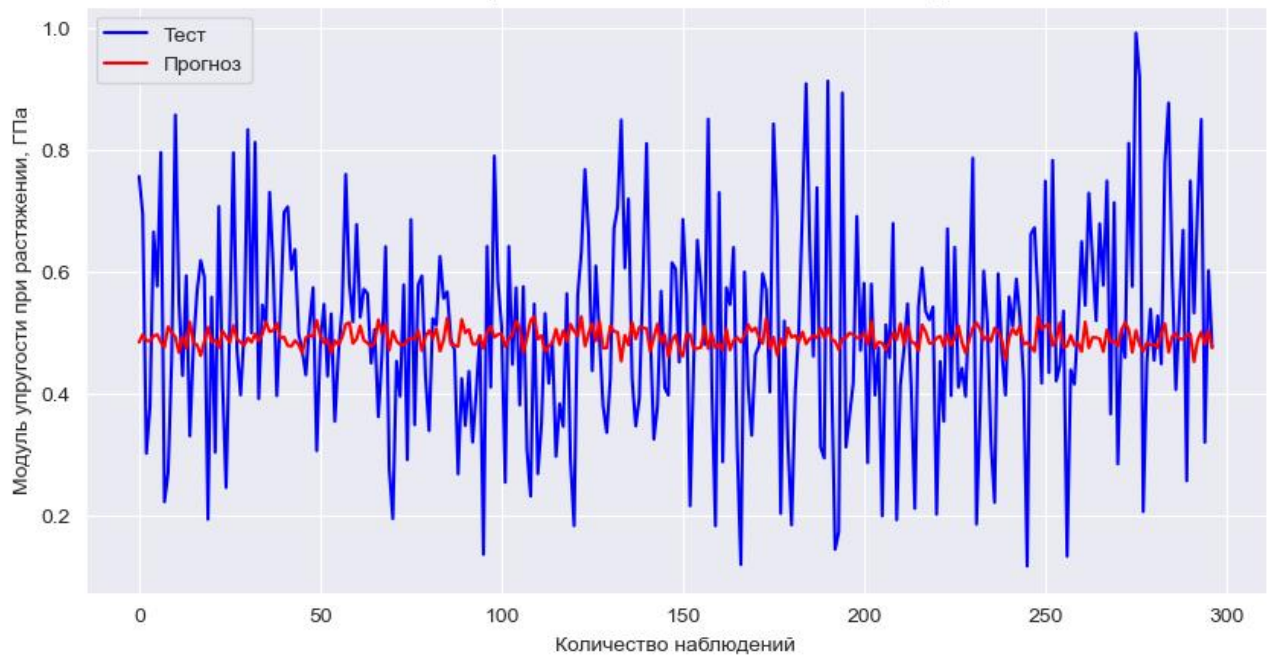


Рисунок 22 – Тестовые и прогнозные значения RandomForestRegressor для Модуля упругости при растяжении

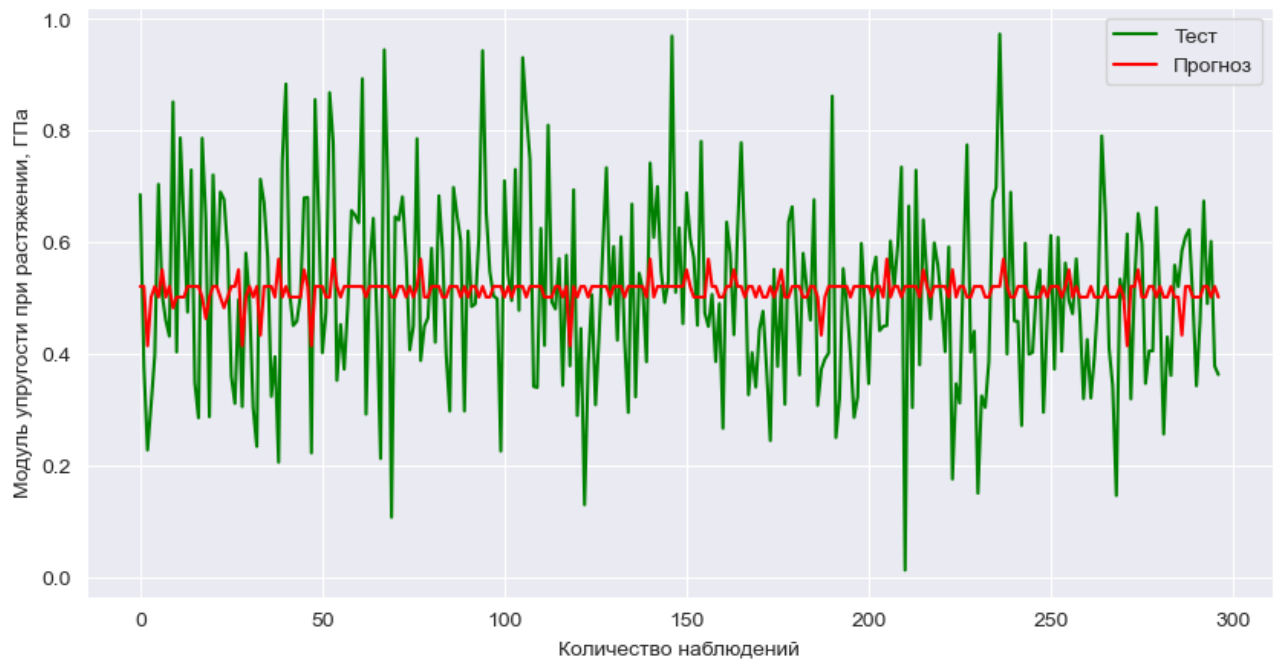


Рисунок 23 – Тестовые и прогнозные значения RandomForestRegressor для Прочности при растяжении

Градиентный бустинг (Gradient Boosting) — это ансамбль деревьев решений, обученный с использованием градиентного бустинга. В основе данного алгоритма лежит итеративное обучение деревьев решений с целью минимизировать функцию потерь.

```
# создание и обучение модели Gradient Boosting Regressor для модуля упругости при растяжении
gbr1 = GradientBoostingRegressor()

#поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой, количество блоков равно 10
gbr1_params = {
    'n_estimators' : range(1, 10),
    'loss': ['squared_error', 'absolute_error', 'huber', 'quantile'],
    'criterion' : ['friedman_mse', 'squared_error', 'mse'],
    'max_depth' : range(1, 5),
    'min_samples_split' : range(1, 10),
    'min_samples_leaf' : range(1, 10, 2),
    'max_features' : range(1, 10)
}
GSCV_gbr1 = RandomizedSearchCV(gbr1, gbr1_params, n_jobs=-1, cv=10)
GSCV_gbr1.fit(X1_train, y1_train)
print('Лучшие параметры Gradient Boosting Regressor для предсказания модуля упругости при растяжении:')
GSCV_gbr1.best_params_

Лучшие параметры Gradient Boosting Regressor для предсказания модуля упругости при растяжении:
{'n_estimators': 7,
 'min_samples_split': 8,
 'min_samples_leaf': 5,
 'max_features': 2,
 'max_depth': 1,
 'loss': 'absolute_error',
 'criterion': 'mse'}

gbr1_upr = GSCV_gbr1.best_estimator_
# предсказанные значения нашей модели
y9_pred = gbr1_upr.predict(X1_test)
```

Рисунок 24 – Построение модели GradientBoostingRegressor

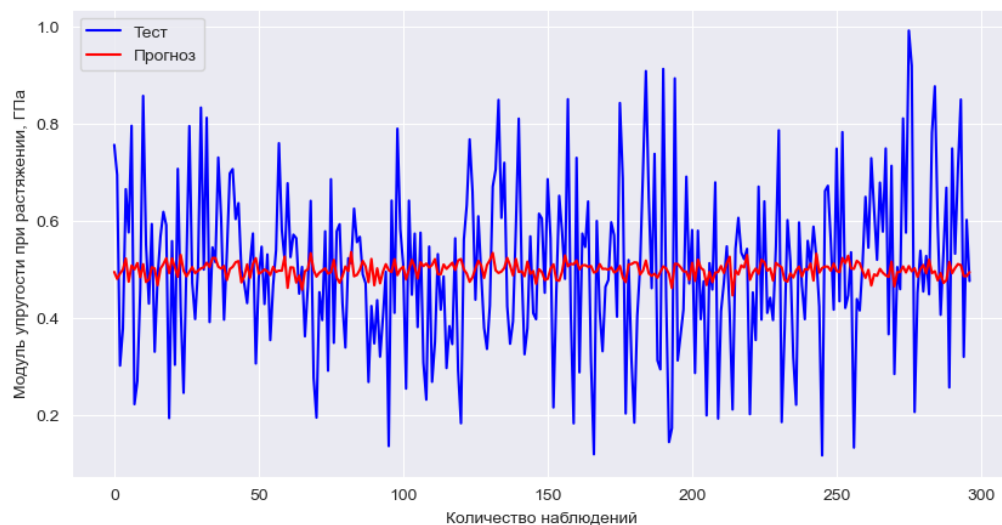


Рисунок 25 – Тестовые и прогнозные значения GradientBoostingRegressor для Модуля упругости при растяжении

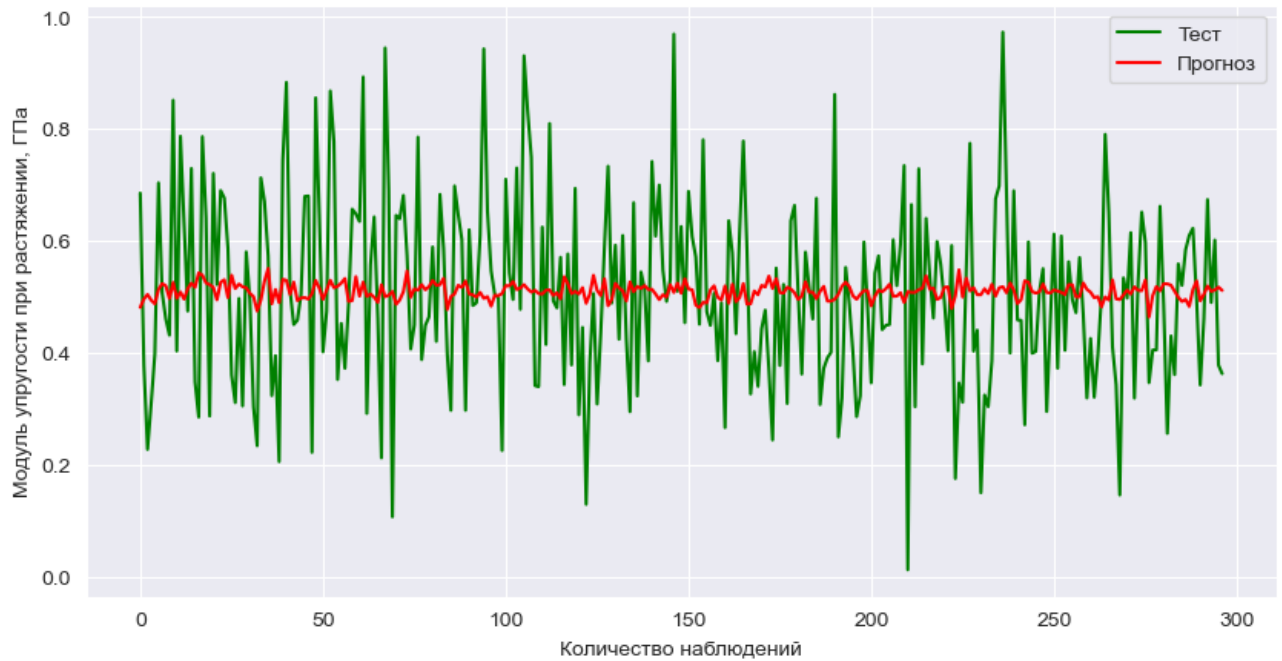


Рисунок 26 – Тестовые и прогнозные значения GradientBoostingRegressor для Прочности при растяжении

2.3. Тестирование модели

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

В качестве метрик для оценки качества моделей использовались: Mean Absolute Error (MAE), Mean Squared Error (MSE), accuracy score и оценка R2.

Mean Absolute Error (MAE): метрика измеряет среднюю сумму абсолютной разницы между фактическим значением и прогнозируемым значением. Чем ниже MAE для данной модели, тем точнее модель способна предсказать фактические значения.

Mean Squared Error (MSE): измеряет среднюю сумму квадратной разности между фактическим значением и прогнозируемым значением для всех точек данных. Выполняется возведение во вторую степень, поэтому отрицательные значения не компенсируются положительными. Чем ниже значение MSE, тем лучше модель способна точно предсказывать значения.

Функция accuracy_score вычисляет точность, либо фракции (по умолчанию) или количество правильных предсказаний.

Оценка R2: один из показателей оценки эффективности моделей машинного обучения на основе регрессии. Также метрика известна как коэффициент детерминации.

Таблица 1. Результаты построения и обучения моделей для Модуля упругости при растяжении

	Модель для модуля упругости при растяжении	MAE	MSE	Score	r2
2	KNeighborsRegressor	0.136126	0.029122	0.004488	0.004488
4	GradientBoostingRegressor	0.136359	0.029413	-0.005487	-0.005487
1	DecisionTree	0.136798	0.029423	-0.005798	-0.005798
0	LinearRegression	0.136385	0.029613	-0.012293	-0.012293
3	RandomForestRegressor	0.137106	0.029656	-0.013782	-0.013782

Для предсказания Модуля упругости при растяжении лучший результат показала модель KNeighborsRegressor: MAE и MSE минимальны, а R2 максимален из всех полученных результатов. В целом, все модели плохо справились с задачей.

Таблица 2. Результаты построения и обучения моделей для Прочности при растяжении

	Модель для прочности при растяжении	MAE	MSE	Score	r2
2	KNeighborsRegressor	0.134416	0.027857	0.007242	0.007242
3	RandomForestRegressor	0.134000	0.028000	0.005194	0.005194
0	LinearRegression	0.136221	0.028019	0.001459	0.001459
4	GradientBoostingRegressor	0.135191	0.028066	-0.000191	-0.000191
1	DecisionTree	0.137633	0.028216	-0.005560	-0.005560

Для предсказания Прочности при растяжении лучший результат также показала модель KNeighborsRegressor: MSE минимально, а R2 максимален из всех полученных результатов. В целом, все модели плохо справились с задачей.

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

Обучение нейронной сети— это такой процесс, при котором происходит подбор оптимальных параметров модели, с точки зрения минимизации функционала ошибки.

```
# нормализуем данные для подачи в нейросеть
normalizer = tf.keras.layers.Normalization(axis=-1)
X_train_ns_norm = normalizer.adapt(np.array(X_train_ns))

# воспользуемся классом Sequential библиотеки Keras, который укажет,
# что мы задаём последовательно связанные между собой слои
model_mn = Sequential(X_train_ns_norm)
model_mn.add(Dense(128))
model_mn.add(BatchNormalization())
model_mn.add(LeakyReLU())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(128, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(64, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(32, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(LeakyReLU())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(16, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(Dense(1))
model_mn.add(Activation('sigmoid'))

# Настроек будет три:
# 1) тип функции потерь (loss function) определяет, как мы будем считать отклонение прогнозного значения от истинного
# 2) способ или алгоритм оптимизации этой функции (optimizer) поможет снизить потерю или ошибку
# и подобрать правильные веса в процессе back propagation
# 3) метрика (metric) покажет, насколько точна наша модель
model_mn.compile(
    optimizer='Adam',
    loss='mse', metrics=['mae'])

history = model_mn.fit(X_train_ns,
                      y_train_ns,
                      epochs=100,
                      verbose=1,
                      validation_split=0.2)
```

Рисунок 27 – 1 вариант нейронной сети

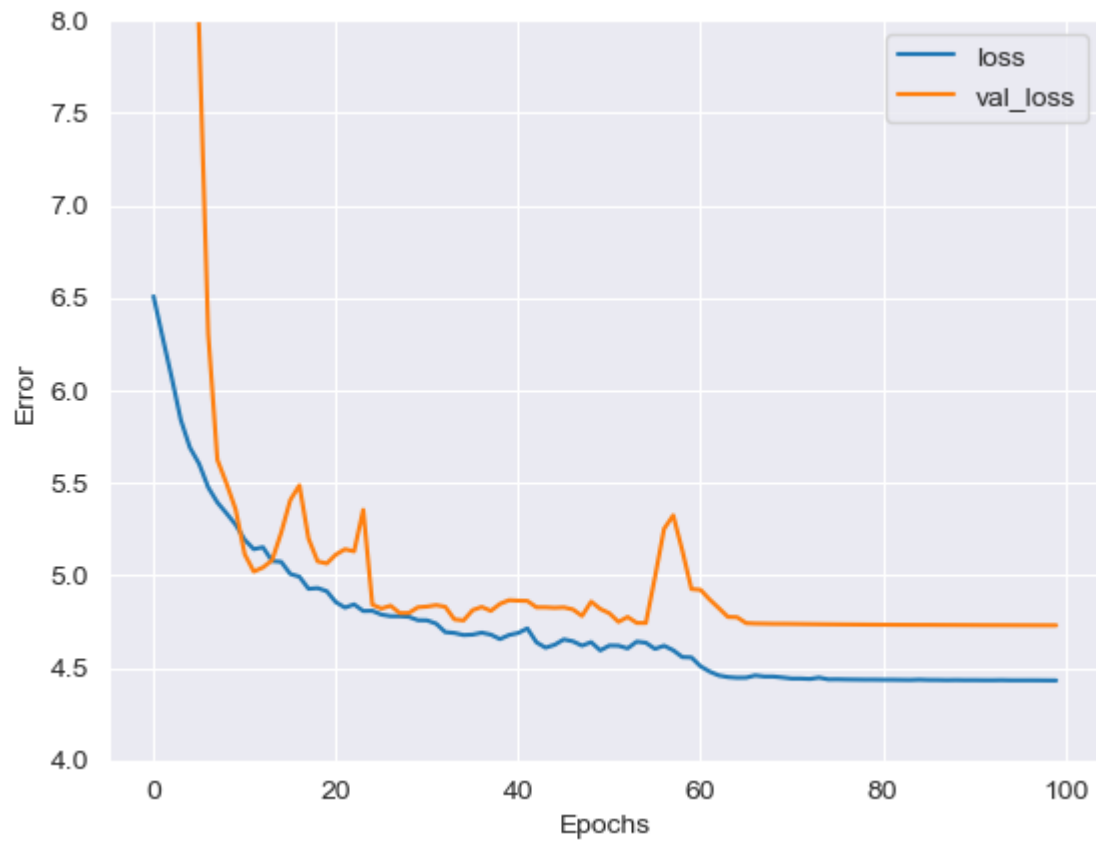


Рисунок 28 – График потерь первой модели

```
# увеличим количество слоев, нейронов и уменьшим количество эпох
model_mn = Sequential(X_train_ns_norm)

model_mn.add(Dense(256))
model_mn.add(BatchNormalization())
model_mn.add(LeakyReLU())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(256, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(128))
model_mn.add(BatchNormalization())
model_mn.add(LeakyReLU())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(128, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(64, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(32, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(LeakyReLU())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(16, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(Dense(1))
model_mn.add(Activation('sigmoid'))

model_mn.compile(
    optimizer='Adam',
    loss='mean_absolute_error', metrics = ['mae'])

history = model_mn.fit( X_train_ns,
                        y_train_ns,
                        epochs = 35,
                        verbose = 1,
                        validation_split = 0.2)
```

Рисунок 29 – 2 вариант нейронной сети

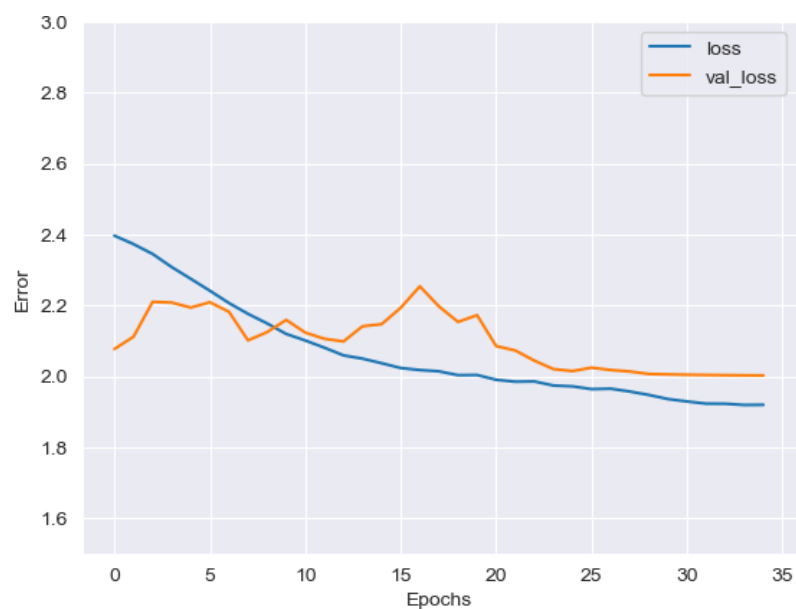


Рисунок 30 – График потерь второй модели

```
# уменьшим количество слоев, нейронов и эпох. Изменим оптимайзер
model_mn = Sequential(X_train_ns_norm)
```

```
model_mn.add(Dense(64, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(32, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(LeakyReLU())
model_mn.add(Dropout(0.18))
model_mn.add(Dense(16, activation='relu'))
model_mn.add(BatchNormalization())
model_mn.add(Dense(1))
model_mn.add(Activation('sigmoid'))
```

```
model_mn.compile(
    optimizer='rmsprop',
    loss='mse', metrics = ['mae'])
```

```
history = model_mn.fit( X_train_ns,
                        y_train_ns,
                        epochs = 20,
                        verbose = 1,
                        validation_split = 0.2)
```

Рисунок 31 – 3 вариант нейронной сети

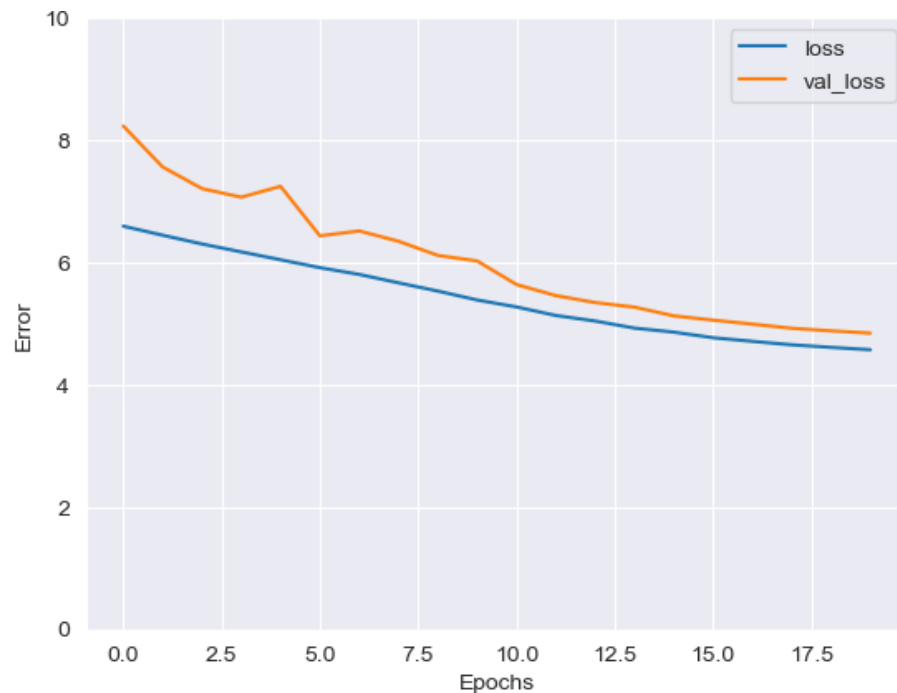


Рисунок 32 – График потерь третьей модели


```
# подадим в нейросеть данные после нормализации после MinMaxScaler
#определяем X y
y_ns = np.array(df_norm_df['Соотношение матрица-наполнитель'])
X_ns = np.array(df_norm_df.drop(['Соотношение матрица-наполнитель'], axis = 1))

#train_test_split
X_train_ns, X_test_ns, y_train_ns, y_test_ns = train_test_split(X_ns, y_ns,
                                                                test_size=0.3,
                                                                random_state = 1)

#архитектура модели
model_ns = Sequential()
model_ns.add(Dense(128, input_shape = (X_train_ns.shape[1],), activation = 'relu'))
model_ns.add(BatchNormalization())
model_ns.add(Dense(128, activation='relu'))
model_ns.add(Dropout(0.18))
model_ns.add(Dense(64, activation='relu'))
model_ns.add(Dropout(0.18))
model_ns.add(Dense(32, activation='relu'))
model_ns.add(Dense(16, activation='relu'))
model_ns.add(BatchNormalization())
model_ns.add(Dense(1, activation = 'sigmoid'))

#компиляция
model_ns.compile(
    optimizer='Adam',
    loss='mean_absolute_error', metrics = ['mae'])

history = model_ns.fit( X_train_ns,
                        y_train_ns,
                        epochs = 100,
                        verbose = 1,
                        validation_split = 0.2)
```

Рисунок 33 – 4 вариант нейронной сети

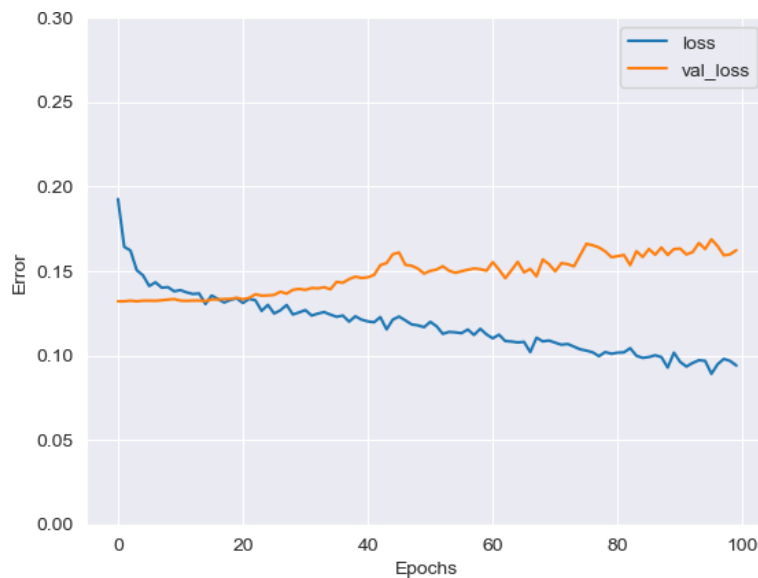


Рисунок 34 – График потерь четвертой модели

```
# увеличим количество слоев и нейронов и уменьшим количество эпох
#архитектура модели
model_ns = Sequential()
model_ns.add(Dense(256, input_shape = (X_train_ns.shape[1],), activation = 'relu'))
model_ns.add(BatchNormalization())
model_ns.add(Dense(128, activation='relu'))
model_ns.add(Dropout(0.18))
model_ns.add(Dense(128, activation='relu'))
model_ns.add(Dropout(0.18))
model_ns.add(Dense(64, activation='relu'))
model_ns.add(Dropout(0.18))
model_ns.add(Dense(64, activation='relu'))
model_ns.add(Dropout(0.18))
model_ns.add(Dense(32, activation='relu'))
model_ns.add(Dense(16, activation='relu'))
model_ns.add(BatchNormalization())
model_ns.add(Dense(1, activation = 'sigmoid'))

#компиляция
model_ns.compile(
    optimizer='Adam',
    loss='mean_absolute_error', metrics = ['mae'])

history = model_ns.fit( X_train_ns,
                        y_train_ns,
                        epochs = 30,
                        verbose = 1,
                        validation_split = 0.2)
```

Рисунок 35 – 5 вариант нейронной сети

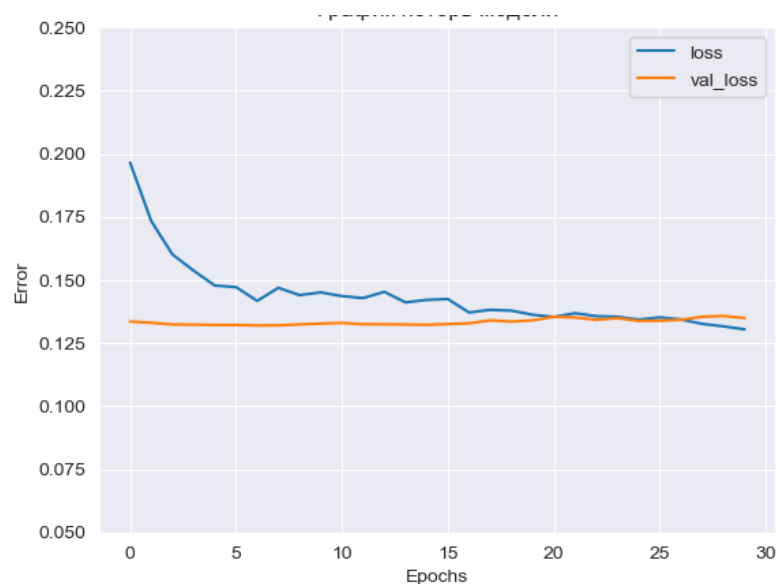
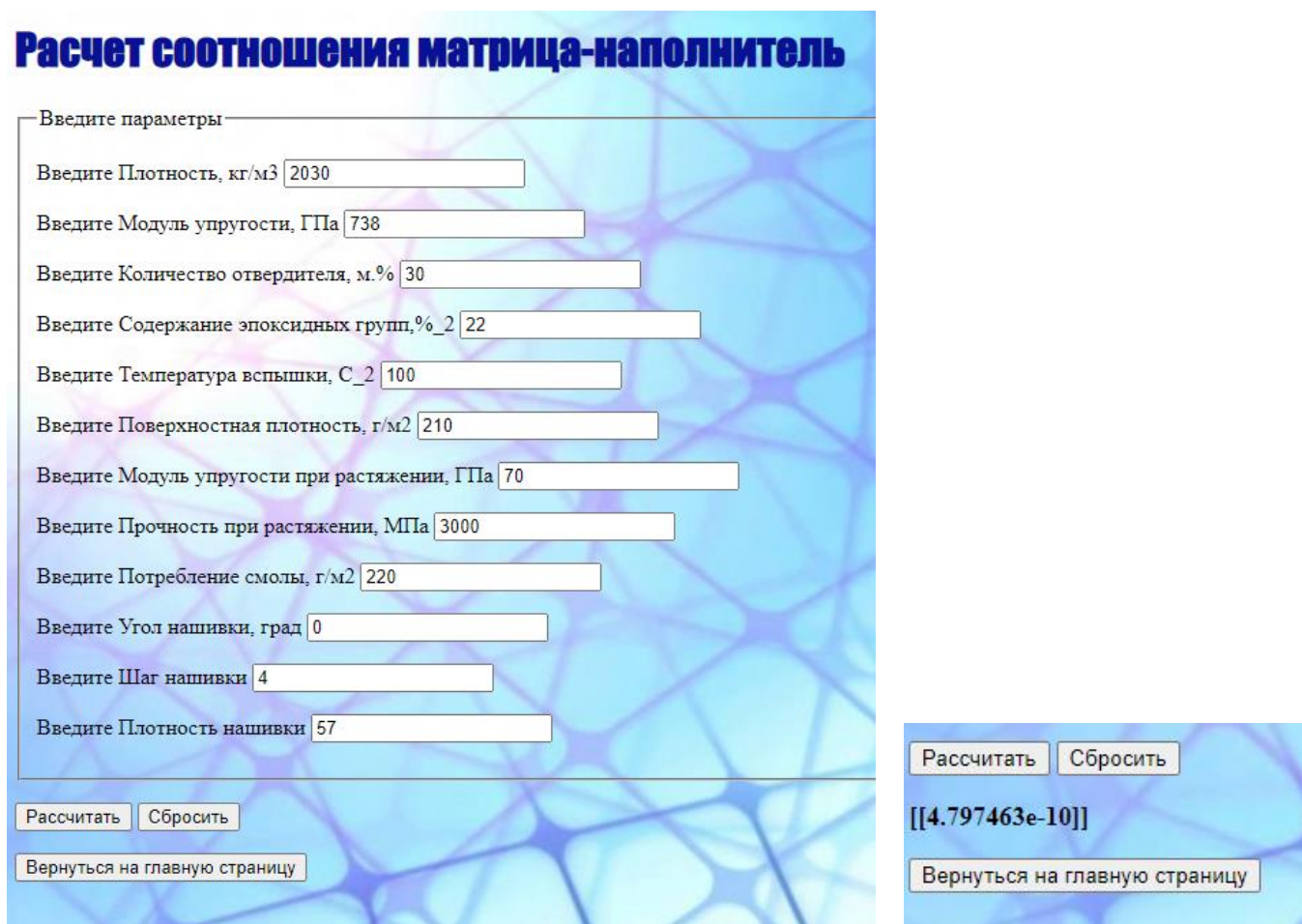


Рисунок 36 – График потерь пятой модели

Проанализировав полученные результаты, сможем сделать вывод, что лучше всего справилась нейронная сеть под номером четыре с активационной функцией Sigmoid и оптимайзером Adam. Хотя после 20-й эпохи сеть начала переобучаться, уменьшение количества эпох не улучшило результатов.

2.5. Разработка приложения

Приложение успешно работает и показывает результат прогноза для соотношения «матрица – наполнитель».



Расчет соотношения матрица-наполнитель

Введите параметры

Введите Плотность, кг/м³

Введите Модуль упругости, ГПа

Введите Количество отвердителя, м.%

Введите Содержание эпоксидных групп, %_2

Введите Температура вспышки, С_2

Введите Поверхностная плотность, г/м²

Введите Модуль упругости при растяжении, ГПа

Введите Прочность при растяжении, МПа

Введите Потребление смолы, г/м²

Введите Угол нашивки, град

Введите Шаг нашивки

Введите Плотность нашивки

[[4.797463e-10]]

Рисунок 37 - Пример результата работы приложения

Данное приложение — это основной файл Flask, папка templates, с шаблоном html - страницы, папка model_mn с сохранённой моделью для данных.

```
app.py > main
1  import flask
2  from flask import render_template
3  import tensorflow as tf
4  from tensorflow import keras
5  import sklearn
6  import keras
7
8  app = flask.Flask(__name__, template_folder='templates')
9
10 @app.route('/', methods=['GET', 'POST'])
11 @app.route('/index', methods=['GET', 'POST'])
12
13 def main():
14     temp = 1
15     param_lst = []
16
17     if flask.request.method == 'GET':
18         return render_template('mn.html' )
19
20     if flask.request.method == 'POST':
21         loaded_model = keras.models.load_model("model_mn")
22         for i in range(1,13,1):
23
24
25             experience = flask.request.form.get(f'experience{i}')
26
27             param_lst.append(float(experience))
28
29         temp = loaded_model.predict([param_lst])
30         return render_template('mn.html', message = temp)
31
32
33 if __name__ == '__main__':
34     app.run()
```

Рисунок 38 - Код приложения

При запуске приложения пользователь переходит на: <http://127.0.0.1:5000/>.

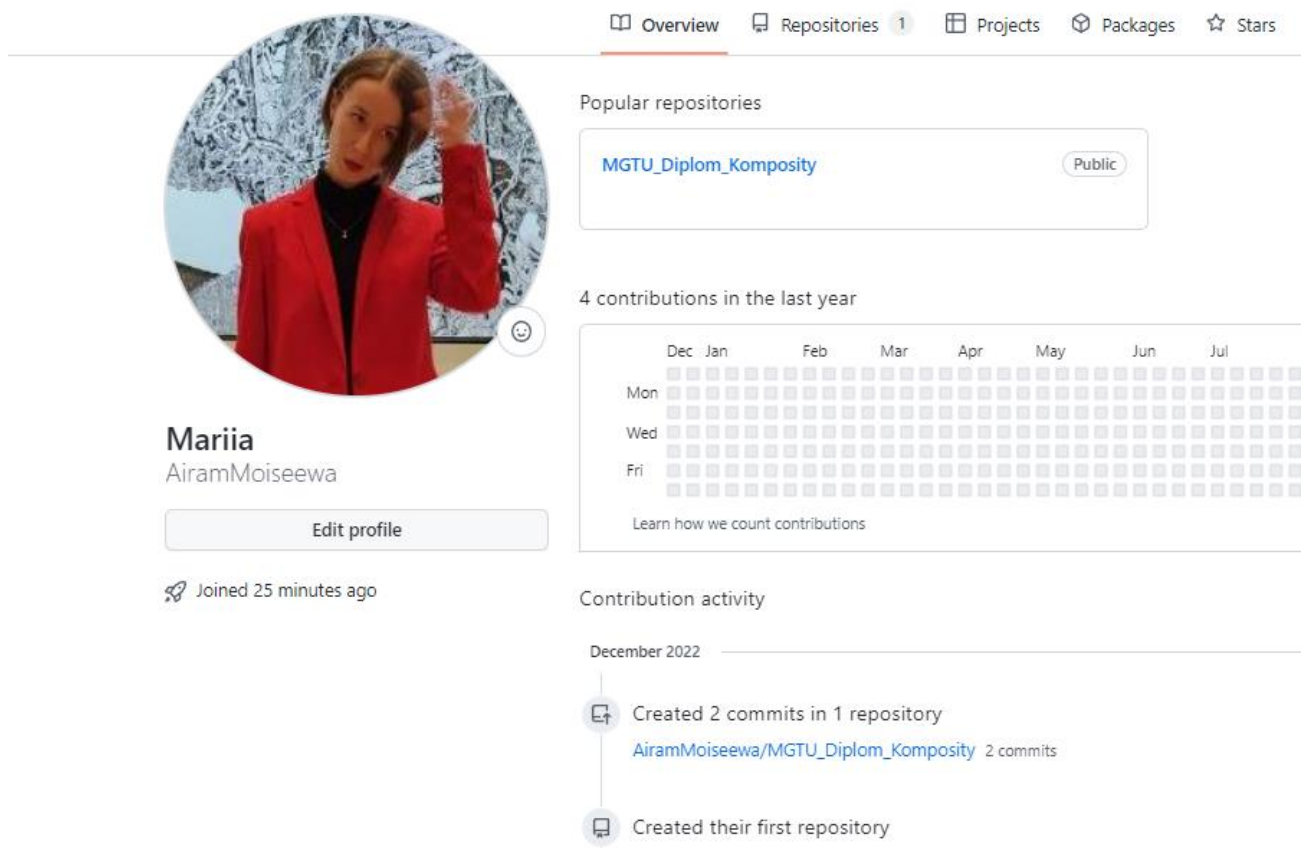
```
(base) m1@WIN-67VNPTIGQOB:/mnt/c/Users/user/Desktop/Diplom/ML_FLASK$ conda activate ml_flask
(ml_flask) m1@WIN-67VNPTIGQOB:/mnt/c/Users/user/Desktop/Diplom/ML_FLASK$ python3 app.py
2022-12-19 00:01:30.031991: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow bin
al operations: AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-12-19 00:01:30.930457: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.c
file or directory
2022-12-19 00:01:30.930584: I tensorflow/compiler/xla/stream_executor/cuda/cudart_stub.cc:29] Ignore
2022-12-19 00:01:34.424674: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.c
or directory
2022-12-19 00:01:34.424822: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.c
: No such file or directory
2022-12-19 00:01:34.424872: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning:
ies mentioned above are installed properly.
* Serving Flask app 'app'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production W
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
□
```

Рисунок 39 - Ссылка для открытия html – файла

В открывшемся окне пользователю необходимо ввести в соответствующие ячейки требуемые значения и нажать на кнопку «Рассчитать». На выходе пользователь получает результат прогноза для значения параметра «Соотношение «матрица – наполнитель».

2.6. Создание удалённого репозитория и загрузка результатов работы на него

Репозиторий был создан на github.com по адресу:
https://github.com/AiramMoiseewa/MGTU_Diplom_Komposity



Overview Repositories 1 Projects Packages Stars

Popular repositories

[MGTU_Diplom_Komposity](#) Public

4 contributions in the last year

	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul
Mon								
Wed								
Fri								


Learn how we count contributions


Contribution activity

December 2022

- Created 2 commits in 1 repository
[AiramMoiseewa/MGTU_Diplom_Komposity](#) 2 commits
- Created their first repository

Рисунок 40 - Часть страницы на github.com


AiramMoiseewa Update README.md
f17e764 1 minute ago 2 commits

 README.md
Update README.md
1 minute ago

☰ README.md

MGTU_Diplom_Komposity

Прогнозирование конечных свойств новых композиционных материалов

Выпускная квалификационная работа по курсу «Data Science»

в Образовательном Центре МГТУ им. Н.Э. Баумана по теме:

"Прогнозирование конечных свойств новых материалов (композиционных материалов)".

Целью данной работы является разработка пользовательского приложения для прогнозирования характеристики конечных свойств новых композиционных материалов.

1). В процессе исследования изучены теоретические основы и методы решения поставленной задачи: Спрогнозировать по входным параметрам ряд конечных свойств получаемых композиционных материалов при следующих используемых признаках:

- Соотношение матрица-наполнитель
- Плотность, кг/м3
- Модуль упругости, ГПа
- Количество отвердителя, м.%
- Содержание эпоксидных групп, %_2
- Температура вспышки, C_2
- Поверхностная плотность, г/м2
- Потребление смолы, г/м2
- Прочность при растяжении, МПа
- Потребление смолы, г/м2
- Угол нашивки, град
- Шаг нашивки
- Плотность нашивки

Рисунок 41 - Часть созданного файла README

Заключение

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 к нулю. Используемые при разработке моделей подходы не позволили получить сколько-нибудь достоверных прогнозов. Применённые модели регрессии не показали высокой эффективности в прогнозировании свойств композитов. Лучшие модели для модуля упругости при растяжении и для прочности при растяжении – метод ближайших соседей.

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 предоставленного 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.

Учитывая отсутствие корреляции между признаками, делаем вывод, что текущим набором алгоритмов задача не решается, возможно, решается трудно или не решается совсем.

Список используемой литературы и веб ресурсы

1. Alex Maszański. Метод k-ближайших соседей (k-nearest neighbour): – Режим доступа: <https://proglib.io/p/metod-k-blizhayshih-sosedey-k-nearest-neighbour-2021-07-19>. (дата обращения: 05.11.2022)
2. Andre Ye. 5 алгоритмов регрессии в машинном обучении, о которых вам следует знать: – Режим доступа: <https://habr.com/ru/company/vk/blog/513842/> (дата обращения: 07.11.2022).
3. Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru. 2020. - 412 с.: ил.
4. Абросимов Н.А.: Методика построения разрешающей системы уравнений динамического деформирования композитных элементов конструкций (Учебно-методическое пособие), ННГУ, 2010
5. Абу-Хасан Махмуд, Масленникова Л. Л.: Прогнозирование свойств композиционных материалов с учётом наноразмера частиц и акцепторных свойств катионов твёрдых фаз, статья 2006 год
6. Бизли Д. Python. Подробный справочник: учебное пособие. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 864 с., ил.
7. 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.
8. Грас Д. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.
9. Документация по библиотеке keras: – Режим доступа: <https://keras.io/api/>. (дата обращения: 18.11.2022).
10. Документация по библиотеке matplotlib: – Режим доступа: <https://matplotlib.org/stable/users/index.html>. (дата обращения: 10.11.2022)

11. Документация по библиотеке numpy: – Режим доступа: <https://numpy.org/doc/1.22/user/index.html#user>. (дата обращения: 03.12.2022).
12. Документация по библиотеке pandas: – Режим доступа: https://pandas.pydata.org/docs/user_guide/index.html#user-guide. (дата обращения: 04.12.2022).
13. Документация по библиотеке scikit-learn: – Режим доступа: https://scikit-learn.org/stable/user_guide.html. (дата обращения: 05.12.2022).
14. Документация по библиотеке seaborn: – Режим доступа: <https://seaborn.pydata.org/tutorial.html>. (дата обращения: 16.12.2022).
15. Документация по библиотеке Tensorflow: – Режим доступа: <https://www.tensorflow.org/overview> (дата обращения: 10.12.2022).
16. Документация по языку программирования python: – Режим доступа: <https://docs.python.org/3.8/index.html>. (дата обращения: 02.12.2022).
17. Иванов Д.А., Ситников А.И., Шляпин С.Д – Композиционные материалы: учебное пособие для вузов, 2019. 13 с.
18. Краткий обзор алгоритма машинного обучения Метод Опорных Векторов (SVM) – Режим доступа: <https://habr.com/ru/post/428503/> (дата обращения 07.12.2022)
19. Ларин А. А., Способы оценки работоспособности изделий из композиционных материалов методом компьютерной томографии, Москва, 2013, 148 с.
20. Материалы конференции: V Всероссийская научно-техническая конференция «Полимерные композиционные материалы и производственные технологии нового поколения», 19 ноября 2021 г.
21. Миронов А.А. Машинное обучение часть I ст.9 – Режим доступа: <http://is.ifmo.ru/verification/machine-learning-mironov.pdf>. (дата обращения 08.12.2022)

22. Плас Дж. Вандер, Python для сложных задач: наука о данных и машинное обучение. Санкт-Петербург: Питер, 2018, 576 с.
23. Реутов Ю.А.: Прогнозирование свойств полимерных композиционных материалов и оценка надёжности изделий из них, Диссертация на соискание учёной степени кандидата физико-математических наук, Томск 2016.
24. Роббинс, Дженнифер. HTML5: карманный справочник, 5-е издание.: Пер. с англ. - М.: ООО «И.Д. Вильямс»: 2015. - 192 с.: ил.
25. Руководство по быстрому старту в flask: – Режим доступа: <https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html>. (дата обращения: 09.12.2022)
26. Силен Дэви, Мейсман Арно, Али Мохамед. Основы Data Science и Big Data. Python и наука о данных. – СПб.: Питер, 2017. – 336 с.: ил.
27. Скиена, Стивен С. C42 Наука о данных: учебный курс.: Пер. с англ. - СПб.: ООО "Диалектика", 2020. - 544 с. : ил.
28. Справочник по композиционным материалам: в 2 - х кн. Кн. 2 / Под ред. Дж. Любина; Пер. с англ. Ф. Б. Геллера, М. М. Гельмонта; Под ред. Б. Э. Геллера - М.: Машиностроение, 1988. - 488 с. : ил;
29. Траск Эндрю. Грокаем глубокое обучение. – СПб.: Питер, 2019. – 352 с.: ил.
30. Чун-Те Чен и Грейс Х. Гу. Машинное обучение для композитных материалов (март 2019г.) – Режим доступа: <https://www.cambridge.org/core/journals/mrs-communications/article/machine-learning-for-composite-materials/F54F60AC0048291BA47E0B671733ED15>. (дата обращения 02.12.2022)