

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»

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

Слушатель

Иванова Дарья Александровна

Москва, 2023

## Содержание

<b>Введение .....</b>	<b>3</b>
<b>1. Аналитическая часть .....</b>	<b>4</b>
<b>1.1 Постановка задачи.....</b>	<b>4</b>
<b>1.2 Описание используемых методов.....</b>	<b>10</b>
<b>1.2.1 Линейная регрессия .....</b>	<b>10</b>
<b>1.2.2 Лассо (LASSO) и гребневая (Ridge) регрессия.....</b>	<b>11</b>
<b>1.2.3 ElasticNet регрессия .....</b>	<b>12</b>
<b>1.2.4 Метод k-ближайших соседей .....</b>	<b>12</b>
<b>1.2.5 Случайный лес .....</b>	<b>13</b>
<b>1.2.6 Нейронная сеть.....</b>	<b>14</b>
<b>1.2.7 Разведочный анализ данных .....</b>	<b>15</b>
<b>2. Практическая часть .....</b>	<b>16</b>
<b>2.1 Предобработка данных .....</b>	<b>16</b>
<b>2.2 Разработка и обучение модели .....</b>	<b>17</b>
<b>2.3 Тестирование модели .....</b>	<b>24</b>
<b>2.4 Нейронная сеть, которая будет рекомендовать соотношение матрица-наполнитель.....</b>	<b>27</b>
<b>2.5 Разработка приложения .....</b>	<b>32</b>
<b>3. Создание удаленного репозитория .....</b>	<b>33</b>
<b>Библиографический список .....</b>	<b>35</b>

## **Введение**

Тема данной работы - прогнозирование конечных свойств новых материалов (композиционных материалов).

Композиционные материалы характеризуются совокупностью свойств, не присущих каждому в отдельности взятому компоненту. За счет выбора армирующих элементов, варьирования их объемной доли в матричном материале, а также размеров, формы, ориентации и прочности связи по границе «матрица-наполнитель», свойства композиционных материалов можно регулировать в значительных пределах.

Возможно получить композиты с уникальными эксплуатационными свойствами. Этим обусловлено широкое применение композиционных материалов в различных областях техники.

Использование КМ в технике связано с разработкой новых принципов конструирования ряда ответственных высоконагруженных изделий и повышением их технологичности.

В настоящее время КМ широко используется при производстве летательных аппаратов, в машиностроении, приборостроении, энергетике, электронной, радиотехнической, электротехнической промышленности, транспорте, медицине, строительстве и других отраслях народного хозяйства.

В последние годы в Российской Федерации проводятся исследования по разработке новых КМ с металлическими, полимерными и углеродными матрицами, организовано производство их полуфабрикатов и конечных изделий.

Углепластики успешно применяются так же для изготовления демпферов вибротранспортеров, загрузочных устройств и др. механизмов.

## 1. Аналитическая часть

### 1.1 Постановка задачи

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 строки и 4 столбцов) рисунок 1.

**Цель работы:** разработать несколько моделей для прогноза модуля упругости при растяжении и прочности при растяжении. Для этого нужно объединить 2 файла X\_br.xlsx и X\_nup.xls по индексу тип объединения – INNER рисунок 2.

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

```
[11] print('The scikit-learn version is {}'.format(sklearn.__version__))

The scikit-learn version is 1.2.2.

Загрузка данных

[ ] #Загружаем первый dataset (базальтопластик) и посмотрим на название столбца
df_br = pd.read_excel(r"/content/sample_data/X_br.xlsx")
df_br.shape

(1023, 11)

[ ] #удаляем первый неинформативный столбец
df_br.drop(['Unnamed: 0'], axis=1, inplace=True)
#посмотрим на первые 5 строк первого датасета и убедимся, что первый столбец удалился
df_br.head()
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.г	Содержание эпоксидных групп, %	Температура всплытия, 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

```
# Проверим размерность первого файла
df_br.shape

(1023, 10)

[ ] # Загружаем второй dataset (углепластик)
df_nup = pd.read_excel(r"/content/sample_data/X_nup.xlsx")
df_nup.shape

(1040, 4)
```

Рисунок 1 - Загрузка файлов X\_br.xlsx и X\_nup.xls

Вот таблица данных, отображенная в ноутбуке:

	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.л	30.000000	50.000000	49.900000	129.000000	111.860000
Содержание эпоксидных групп, %	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

Рисунок 2 - Объединение по типу INNER

Часть информации (17 строк таблицы способов компоновки композитов) не имеют соответствующих строк в таблице соотношений и свойств используемых компонентов композитов, поэтому были удалены.

В качестве входных данных приняты данные о начальных свойствах компонентов композиционных материалов:

- Соотношение матрица-наполнитель;
- Плотность;
- Модуль упругости;
- Количество отвердителя;
- Содержание эпоксидных групп;
- Температура вспышки;
- Поверхностная плотность;
- Модуль упругости при растяжении;
- Прочность при растяжении;
- Потребление смолы;
- Угол нашивки;
- Шаг нашивки;
- Плотность нашивки.

Общее количество параметров для анализа – 13.

На выходе необходимо спрогнозировать ряд конечных свойств получаемых композиционных материалов. Кейс основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана).

```
[ ] #Количество пропусков по каждому столбцу датасета
df.isna().sum()

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
```

Рисунок 3 –Количество пропусков в датасете

```
#Количество уникальных значений по каждому столбцу датасета
df.nunique()

#Анализ показал, что столбец "Угол нашивки, град" имеет всего 2 уникальных значения,
# что позволяет нам рассматривать его значения как категориальный тип.

Соотношение матрица-наполнитель      1014
Плотность, кг/м3                     1013
модуль упругости, ГПа                1020
Количество отвердителя, м.%          1005
Содержание эпоксидных групп,%_2     1004
Температура вспышки, С_2             1003
Поверхностная плотность, г/м2        1004
Модуль упругости при растяжении, ГПа 1004
Прочность при растяжении, МПа        1004
Потребление смолы, г/м2              1003
Угол нашивки, град                    2
Шаг нашивки                           989
Плотность нашивки                     988
dtype: int64
```

Рисунок 4 –Количество уникальных значений

Пропусков в данных нет. Все признаки, кроме «Угол нашивки», являются непрерывными, количественными. «Угол нашивки» принимает только два значения и будет рассматриваться как категориальный признак. Так же нас интересует описательная статистика датасета, которая представлена на рисунке 5.

```
#Описательная статистика датасета
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп, %_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Рисунок 5 – Описательная статистика датасета

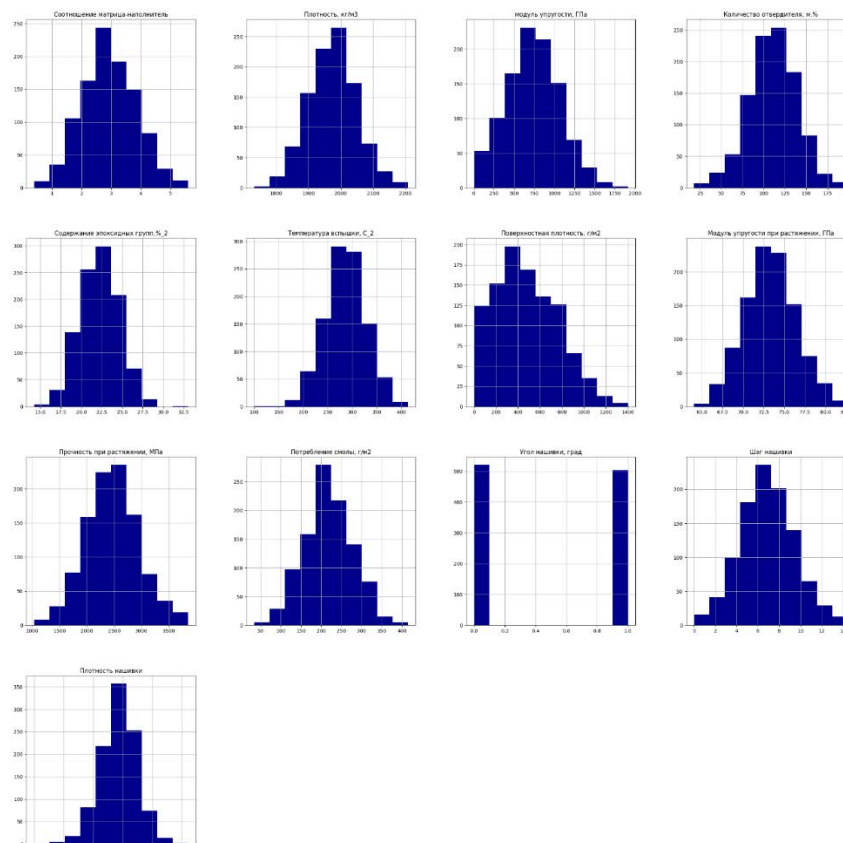


Рисунок 6 - Гистограммы распределения переменных

Гистограммы распределения переменных и диаграммы «ящик с усами» приведены на рисунке 6. По ним видно, что все признаки, кроме «Угол нашивки», имеют нормальное распределение и принимают неотрицательные значения. «Угол нашивки» принимает значения: 0, 90.



На рисунке 7 мы видим график попарного рассеяния точек. По форме «облаков точек» мы не заметили зависимостей, которые станут основой работы моделей.

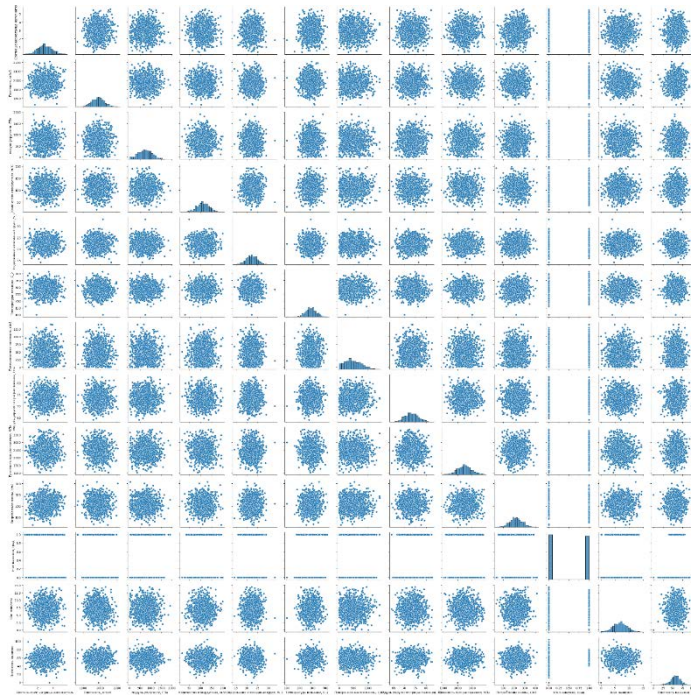


Рисунок 7 – график попарного рассеяния точек

Помочь выявить связь между признаками может матрица корреляции, приведенная на рисунке 8.

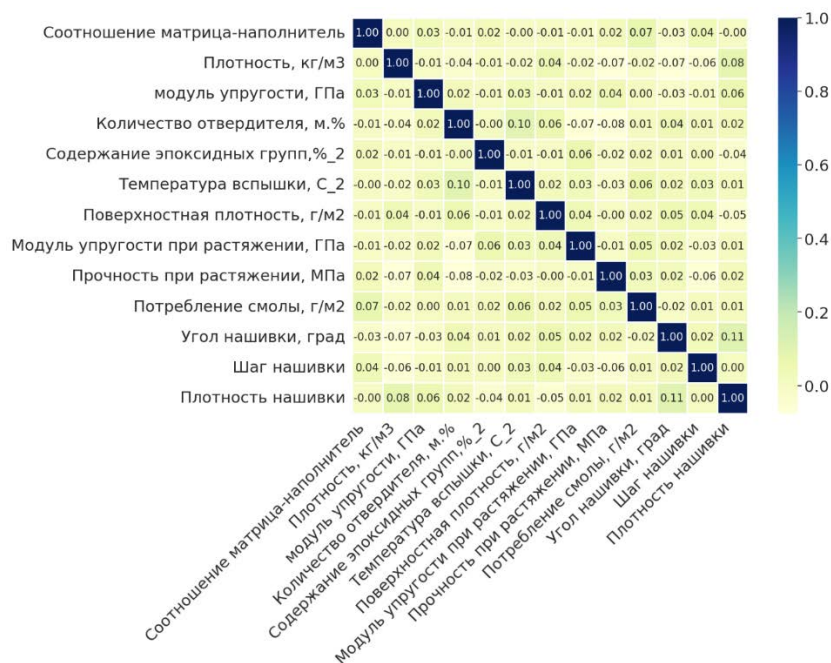


Рисунок 8 — Матрица корреляции

По матрице корреляции мы видим, что все коэффициенты корреляции близки к нулю, что означает отсутствие линейной зависимости между признаками.



В задании целевыми переменными указаны:

- модуль упругости при растяжении, ГПа;
- прочность при растяжении, МПа;
- соотношение матрица-наполнитель.

График боксплот рисунок 9, построенный на масштабированных данных, гораздо лучше отображает необходимость удаления выбросов.



Рисунок 9 — График боксплот «Ящик с усами» до очистки от выбросов

Для удаления выбросов существует 2 основных метода - метод 3-х сигм и межквартильных расстояний. Сравним эти 2 метода рисунок 10.

```
# методом 3-х сигм
zscore = (df[column] - df[column].mean()) / df[column].std()
d['3s'] = zscore.abs() > 3
metod_3s += d['3s'].sum()
count_3s.append(d['3s'].sum())
print(column, '3s', ': ', d['3s'].sum())

# методом межквартильных расстояний
q1 = np.quantile(df[column], 0.25)
q3 = np.quantile(df[column], 0.75)
iqr = q3 - q1
lower = q1 - 1.5 * iqr
upper = q3 + 1.5 * iqr
d['iq'] = (df[column] <= lower) | (df[column] >= upper)
metod_iq += d['iq'].sum()
count_iq.append(d['iq'].sum())
print(column, ': ', d['iq'].sum())
print('Метод 3-х сигм, выбросов:', metod_3s)
print('Метод межквартильных расстояний, выбросов:', metod_iq)
```

Рисунок 10 — Метод 3-х сигм и межквартильных расстояний

Применив эти методы на нашем датасете было найдено:

- методом 3-х сигм — 24 выброса;
- методом межквартильных расстояний — 93 выброса.

После обработки датасета количество строк, подлежащих передаче моделям на обучение составляет 936 строк и 13 столбцов, а ящик с усами стал выглядеть гораздо «чище» рисунок 11.

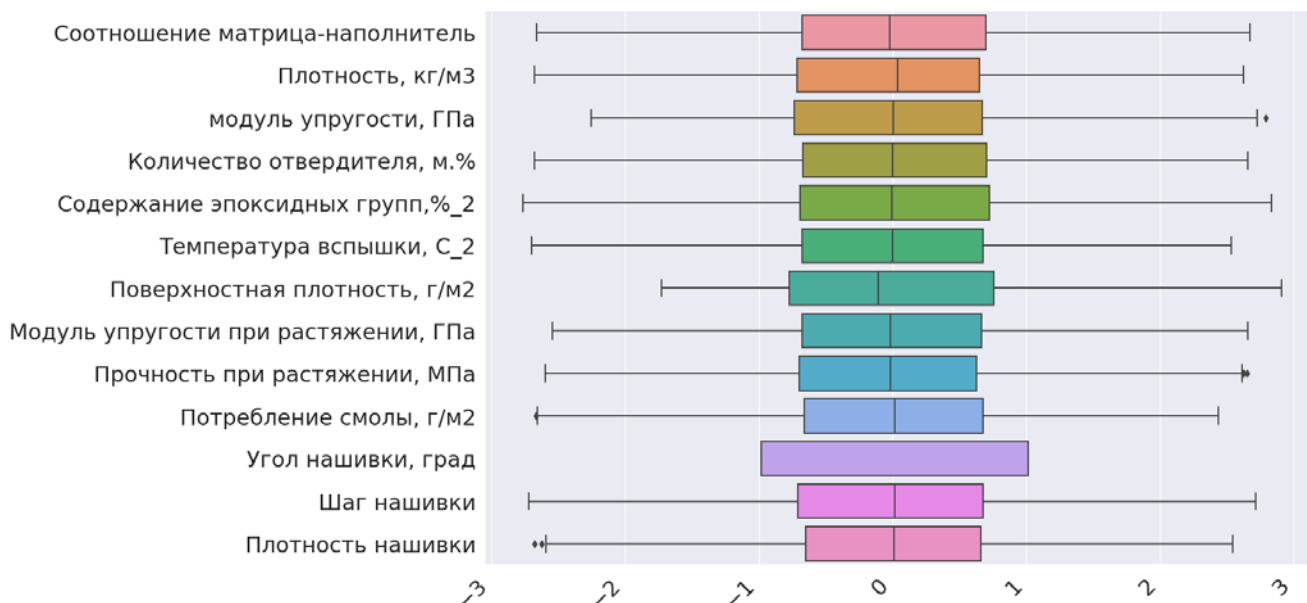


Рисунок 11 — График боксплот «Ящик с усами» после очистки от выбросов

## 1.2 Описание используемых методов

Предсказание значений вещественной, непрерывной переменной — это задача регрессии. Эта зависимая переменная должна иметь связь с одной или несколькими независимыми переменными, называемых также предикторами или регрессорами. Регрессионный анализ помогает понять, как «типичное» значение зависимой переменной изменяется при изменении независимых переменных.

### 1.2.1 Линейная регрессия

Простая линейная регрессия имеет место, если рассматривается зависимость между одной входной и одной выходной переменными. Для этого определяется уравнение регрессии (1) и строится соответствующая прямая, известная как линия регрессии.

$$y=ax+b \quad (1)$$

Коэффициенты  $a$  и  $b$ , называемые также параметрами модели, определяются таким образом, чтобы сумма квадратов отклонений точек, соответствующих

реальным наблюдениям данных, от линии регрессии была бы минимальной. Коэффициенты обычно оцениваются методом наименьших квадратов.

Если ищется зависимость между несколькими входными и одной выходной переменными, то имеет место множественная линейная регрессия. Соответствующее уравнение имеет вид (2).

$$Y=b_0+b_1*x_1+b_2*x_2+\dots+b_n*x_n, \quad (2)$$

где  $n$  - число входных переменных.

Линейная регрессия — первый тщательно изученный метод регрессионного анализа. Его главное достоинство — простота. Такую модель можно построить и рассчитать даже без мощных вычислительных средств. Простота является и главным недостатком этого метода. Тем не менее, именно с линейной регрессии целесообразно начать подбор подходящей модели.

На языке python линейная регрессия реализована в `sklearn.linear_model.LinearRegression`.

### 1.2.2 Лассо (LASSO) и гребневая (Ridge) регрессия

Метод регрессии лассо (LASSO, Least Absolute Shrinkage and Selection Operator) — это вариация линейной регрессии, специально адаптированная для данных, которые имеют сильную корреляцию признаков друг с другом.

LASSO использует сжатие коэффициентов (shrinkage) и этим пытается уменьшить сложность данных, искривляя пространство, на котором они лежат. В этом процессе лассо автоматически помогает устранить или исказить сильно коррелированные и избыточные функции в методе с низкой дисперсией.

Регрессия лассо использует регуляризацию  $L_1$ , то есть взвешивает ошибки по их абсолютному значению.

Гребневая регрессия или ридж-регрессия — так же вариация линейной регрессии, очень похожая на регрессию LASSO. Она так же применяет сжатие и хорошо работает для данных, которые демонстрируют сильную мультиколлинеарность.

Самое большое различие между ними в том, что гребневая регрессия использует регуляризацию L2, которая взвешивает ошибки по их квадрату, чтобы сильнее наказывать за более значительные ошибки.

Регуляризация позволяет интерпретировать модели. Если коэффициент стал 0 (для Lasso) или близким к 0 (для Ridge), значит данный входной признак не является значимым.

Эти методы реализованы в `sklearn.linear_model.Lasso` и `sklearn.linear_model.Ridge`.

### 1.2.3 ElasticNet регрессия

Эластичная сетевая регрессия - это алгоритм классификации, который преодолевает ограничения метода lasso (оператор наименьшей абсолютной усадки и выбора), который использует штрафную функцию в своей регуляризации L1. Эластичная сетевая регрессия - это гибридный подход, который сочетает в себе оба наказания за регуляризацию L2 и L1 методов lasso и ridge.

Он находит оценщик в двухэтапной процедуре, т. е. сначала для каждого фиксированного  $\lambda_2$  он находит коэффициенты регрессии гребня, а затем выполняет сжатие типа регрессии лассо, которое увеличивает усадку в два раза, что в конечном итоге приводит к увеличению смещения и плохим прогнозам. Изменение масштаба коэффициентов наивной версии эластичной сети путем умножения оцененных коэффициентов на  $(1+2)$  выполняется для улучшения производительности прогнозирования.

### 1.2.4 Метод k-ближайших соседей

Еще один метод классификации, который адаптирован для регрессии - метод k-ближайших соседей (k Nearest Neighbors). На интуитивном уровне суть метода проста: посмотри на соседей вокруг, какие из них преобладают, таковым ты и являешься.

В случае использования метода для регрессии, объекту присваивается среднее значение по k ближайшим к нему объектам, значения которых уже известны.

Для реализации метода необходима метрика расстояния между объектами. Используется, например, евклидово расстояние для количественных признаков или расстояние Хэмминга для категориальных.

Этот метод — пример непараметрической регрессии.

Он реализован в `sklearn.neighbors.KNeighborsRegressor`.

### 1.2.5 Случайный лес

Случайный лес (RandomForest) — представитель ансамблевых методов.

Если точность дерева решений оказалась недостаточной, мы можем множество моделей собрать в коллектив. Формула итогового решателя (3) — это усреднение предсказаний отдельных деревьев.

$$a(x) = \frac{1}{N} \sum_{i=1}^N b_i(x) \quad (3),$$

где

- N – количество деревьев;
- i – счетчик для деревьев;
- b – решающее дерево;
- x – сгенерированная нами на основе данных выборка.

Для определения входных данных каждому дереву используется метод случайных подпространств. Базовые алгоритмы обучаются на различных подмножествах признаков, которые выделяются случайным образом.

Преимущества случайного леса:

- высокая точность предсказания;
- редко переобучается;
- практически не чувствителен к выбросам в данных;
- одинаково хорошо обрабатывает как непрерывные, так и дискретные признаки, данные с большим числом признаков;
- высокая параллелизуемость и масштабируемость.

Из недостатков можно отметить, что его построение занимает больше времени. Так же теряется интерпретируемость.

Метод реализован в `sklearn.ensemble.RandomForestRegressor`.

### 1.2.6 Нейронная сеть

Нейронная сеть — это последовательность нейронов, соединенных между собой связями. Структура нейронной сети пришла в мир программирования из биологии. Вычислительная единица нейронной сети — нейрон или персептрон.

У каждого нейрона есть определённое количество входов, куда поступают сигналы, которые суммируются с учётом значимости (веса) каждого входа.

Смещение – это дополнительный вход для нейрона, который всегда равен 1 и, следовательно, имеет собственный вес соединения.

Так же у нейрона есть функция активации, которая определяет выходное значение нейрона. Она используется для того, чтобы ввести нелинейность в нейронную сеть. Примеры активационных функций: `relu`, сигмоида.

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

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

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

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

Для обновления весов в модели используются различные оптимизаторы.

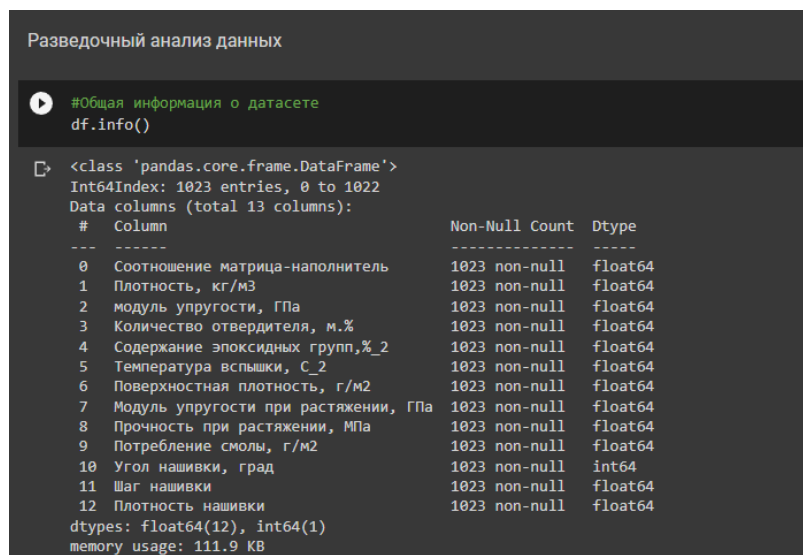
Количество эпох показывает, сколько раз выполнялся проход для всех примеров обучения.

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

### 1.2.7 Разведочный анализ данных

Цель разведочного анализа данных — выявить закономерности в данных. Для корректной работы большинства моделей желательна сильная зависимость выходных переменных от входных и отсутствие зависимости между входными переменными.

Описание признаков объединенного датасета приведены на рисунке 11. Все признаки имеют тип float64, то есть вещественный. Пропусков в данных нет. Все признаки, кроме «Угол нашивки», являются непрерывными, количественными. «Угол нашивки» принимает только два значения и будет рассматриваться как категориальный признак.



```
Разведочный анализ данных

#Общая информация о датасете
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 1023 entries, 0 to 1022
Data columns (total 13 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Соотношение матрица-наполнитель          1023 non-null   float64
1   Плотность, кг/м3                          1023 non-null   float64
2   модуль упругости, ГПа                     1023 non-null   float64
3   Количество отвердителя, м.%              1023 non-null   float64
4   Содержание эпоксидных групп, %_2         1023 non-null   float64
5   Температура вспышки, C_2                 1023 non-null   float64
6   Поверхностная плотность, г/м2            1023 non-null   float64
7   Модуль упругости при растяжении, ГПа     1023 non-null   float64
8   Прочность при растяжении, МПа            1023 non-null   float64
9   Потребление смолы, г/м2                  1023 non-null   float64
10  Угол нашивки, град                       1023 non-null   int64
11  Шаг нашивки                              1023 non-null   float64
12  Плотность нашивки                         1023 non-null   float64
dtypes: float64(12), int64(1)
memory usage: 111.9 KB
```

Рисунок 11 – Общая информация о датасете



## 2. Практическая часть

### 2.1 Предобработка данных

В ходе проведённого анализа принимаем решение столбец "Угол нашивки" привести к виду «0» и «1» рисунок 12.

```
[ ] #Переведем данные столбца в категориальный тип
le = LabelEncoder()
df['Угол нашивки, град'] = le.fit_transform(df['Угол нашивки, град'].values)
df['Угол нашивки, град']

0      0
1      0
2      0
3      0
4      0
..
1018   1
1019   1
1020   1
1021   1
1022   1
Name: Угол нашивки, град, Length: 1023, dtype: int64
```

Рисунок 12 - Преобразование столбца "Угол нашивки"

По условиям задания нормализуем значения. Для этого применим `MinMaxScaler()`, а затем применим `Normalizer()` графики представлены на рисунках 13-14.

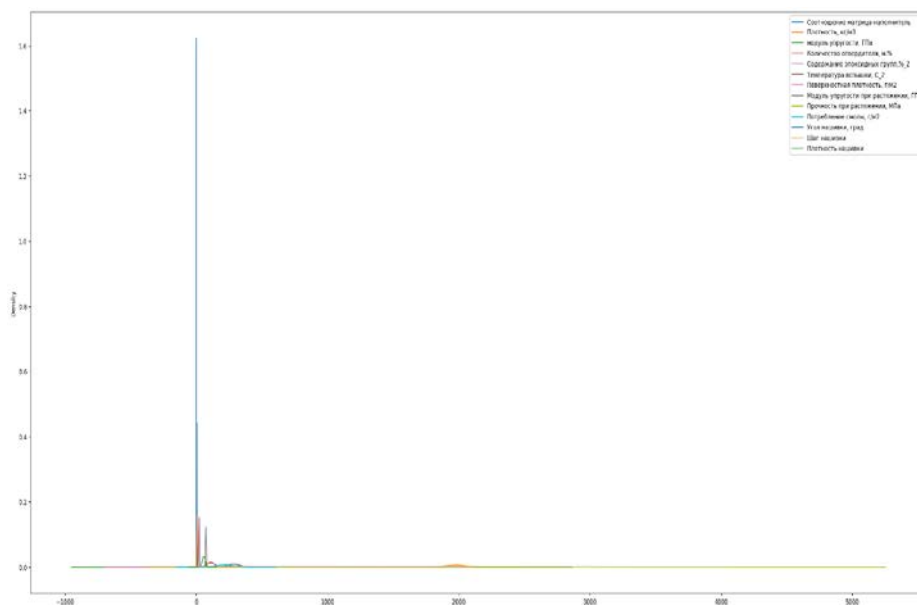


Рисунок 13 - Визуализированные данные с помощью `MinMaxScaler()`,

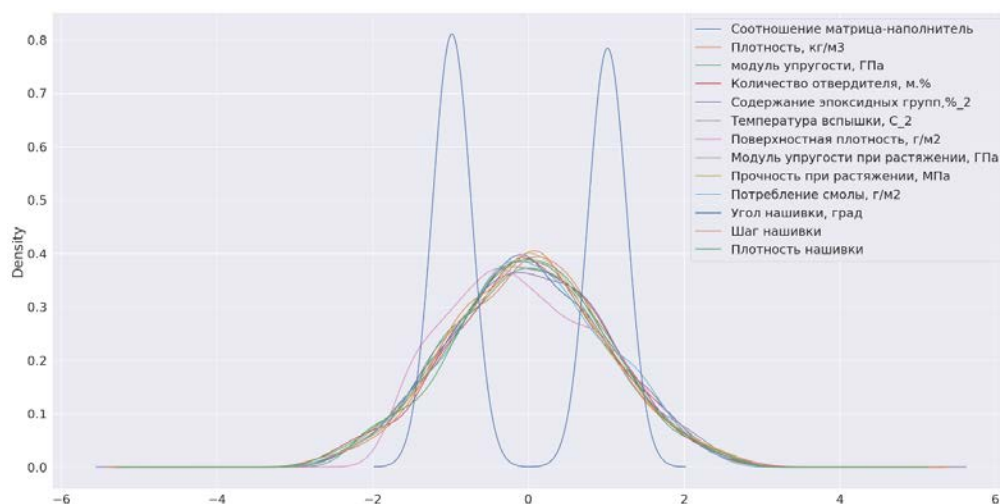


Рисунок 14 - Визуализированные данные с помощью Normalizer().

## 2.2 Разработка и обучение модели

Прогнозируемые величины «Модуль упругости при растяжении» и «Модуль прочности при растяжении», а также все признаки, за исключением столбца «Угол нашивки», являются непрерывными. Следовательно, для решения задачи такого типа будем использовать различные модели регрессии. Но так как в данных нет явной корреляции между признаками, регрессионная модель может не дать хорошего прогноза. В таком случае можно использовать более сложные модели, способные выявить взаимосвязь между признаками. Для таких целей хорошо подойдет модель случайного леса. В виду большого числа признаков, находящихся в разных измерениях неплохим выбором будет метод k-ближайших соседей. По условию задачи, необходимо реализовать многослойную нейронную сеть, которая также должна хорошо обобщиться на многомерном датасете, попытавшись выявить новые зависимости между признаками.

Для начала обучения необходимо выделить из датасета целевые признаки, которые будем прогнозировать, а также разобьем его на обучающий и тестовый набор рисунок 15.

```
#Для упрощения реализации задачи объединим целевые признаки в единый датафрейм.
train = df_norm.drop(['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа'], axis=1)
target = df_norm[['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа']]

#Разбиение на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(
    train,
    target,
    test_size=0.2,
    random_state=1)
```

Рисунок 15 – Целевые признаки

Признаки датасета были разделены на входные и выходные, а строки - на тренировочное и тестовое множество. Размерности полученных наборов данных показаны на рисунке 16.

```
#Проверяем размерность получившихся выборок
print(X_train.shape)
print(X_test.shape)

print(y_train.shape)
print(y_test.shape)

#Размерности совпадают
```

```
(748, 11)
(188, 11)
(748, 2)
(188, 2)
```

Рисунок 16 – Размерности тренировочного и тестового множеств после разбиения для 1-й задачи

При просмотре получившихся выборок, можно убедиться в том, что размерности -совпадают.

Для оценки качества моделей будем использовать метрику R2. Она представляет собой долю вариации зависимой переменной, которая предсказуема из независимой переменной (переменных). Создадим хранилище для оценок, чтобы в дальнейшем выбрать лучшую модель

scores = { } - Словарь коэффициентов детерминации для каждой модели.

Далее будут рассмотрены следующие регрессии:

- **Линейная регрессия**

```

#Обучаем модель для предсказания признака "Модуль упругости при растяжении"
lr.fit(X_train, y_train.iloc[:, 0])
y_pred.append(lr.predict(X_test))
r2_score_feature_1 = r2_score(y_test.iloc[:, 0], y_pred[0])
print(f'Коэффициент детерминации для признака "{y_train.columns[0]}": {r2_score(y_test.iloc[:, 0], y_pred[0])}')
lr_scores["lr_r2_score_feature_1"] = r2_score_feature_1

#Обучаем модель для предсказания признака "Прочность при растяжении, МПа"
lr.fit(X_train, y_train.iloc[:, 1])
y_pred.append(lr.predict(X_test))
r2_score_feature_2 = r2_score(y_test.iloc[:, 1], y_pred[1])
print(f'Коэффициент детерминации для признака "{y_train.columns[1]}": {r2_score(y_test.iloc[:, 1], y_pred[1])}')
lr_scores["lr_r2_score_feature_2"] = r2_score_feature_2

scores['LinearRegression'] = lr_scores

Коэффициент детерминации для признака "Модуль упругости при растяжении, ГПа": 0.7826749174271698
Коэффициент детерминации для признака "Прочность при растяжении, МПа": 0.9552506002027553

```

Рисунок 17 – Обучение моделей

Визуализация полученного результата для обоих признаков показана на рисунках 18-19

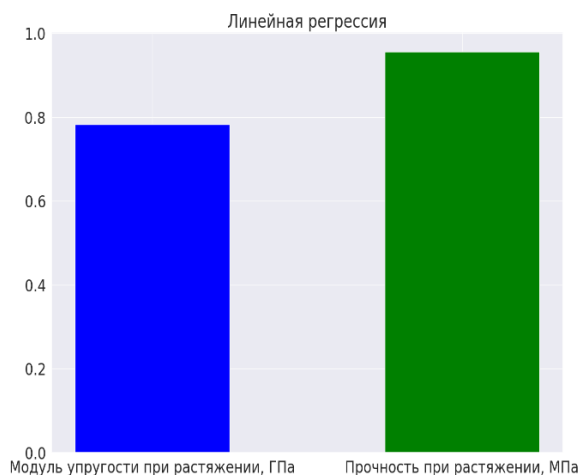


Рисунок 18 – Гистограмма линейной регрессии

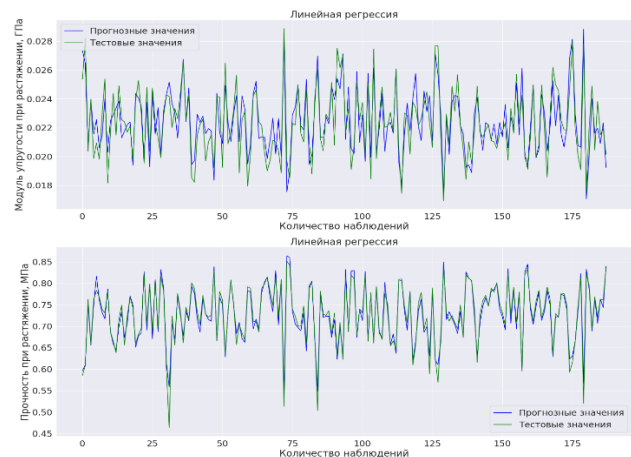


Рисунок 19 – График линейной регрессии

## • LASSO-регрессия

```

#Обучаем модель для предсказания признака "Модуль упругости при растяжении"
LCV.fit(X_train, y_train.iloc[:, 0])
y_pred.append(LCV.predict(X_test))
r2_score_feature_1 = r2_score(y_test.iloc[:, 0], y_pred[0])
print(f'Коэффициент детерминации для признака "{y_train.columns[0]}": {r2_score(y_test.iloc[:, 0], y_pred[0])}')
LCV_scores["LCV_r2_score_feature_1"] = r2_score_feature_1

#Обучаем модель для предсказания признака "Прочность при растяжении, МПа"
LCV.fit(X_train, y_train.iloc[:, 1])
y_pred.append(LCV.predict(X_test))
r2_score_feature_2 = r2_score(y_test.iloc[:, 1], y_pred[1])
print(f'Коэффициент детерминации для признака "{y_train.columns[1]}": {r2_score(y_test.iloc[:, 1], y_pred[1])}')
LCV_scores["LCV_r2_score_feature_2"] = r2_score_feature_2

scores['LassoCV'] = LCV_scores

Коэффициент детерминации для признака "Модуль упругости при растяжении, ГПа": 0.7735465492598215
Коэффициент детерминации для признака "Прочность при растяжении, МПа": 0.9563546747418563

```

Рисунок 20 – Обучение моделей

Визуализация полученного результата для обоих признаков показана на рисунках 21-22

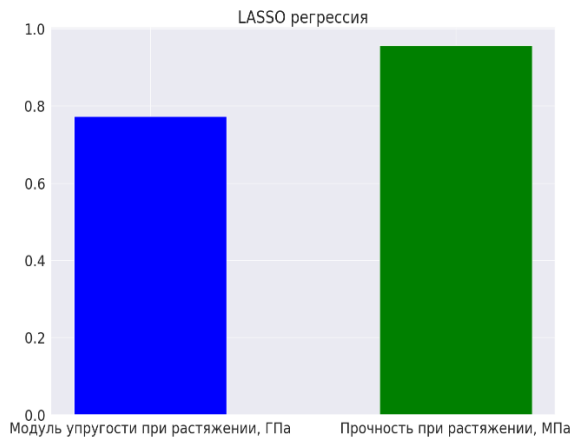


Рисунок 21 – Гистограмма LASSO-регрессия

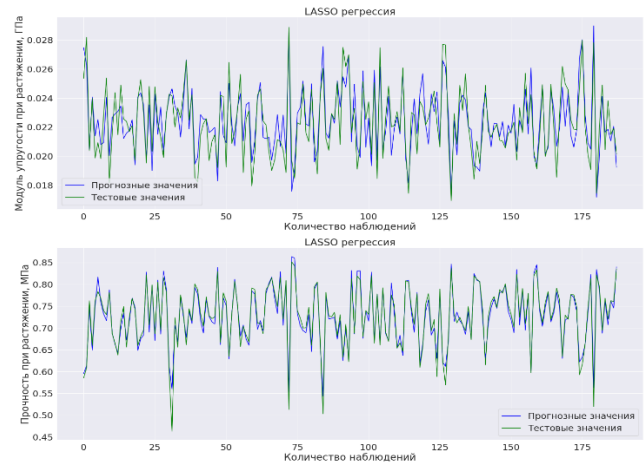


Рисунок 22 – График LASSO-регрессия

- Гребневая регрессия

```
#Обучаем модель для предсказания признака "Модуль упругости при растяжении"
ridge.fit(X_train, y_train.iloc[:, 0])
y_pred.append(ridge.predict(X_test))
r2_score_feature_1 = r2_score(y_test.iloc[:, 0], y_pred[0])
print(f'Коэффициент детерминации для признака "{y_train.columns[0]}": {r2_score(y_test.iloc[:, 0], y_pred[0])}')
ridge_scores["ridge_r2_score_feature_1"] = r2_score_feature_1

#Обучаем модель для предсказания признака "Прочность при растяжении, МПа"
ridge.fit(X_train, y_train.iloc[:, 1])
y_pred.append(ridge.predict(X_test))
r2_score_feature_2 = r2_score(y_test.iloc[:, 1], y_pred[1])
print(f'Коэффициент детерминации для признака "{y_train.columns[1]}": {r2_score(y_test.iloc[:, 1], y_pred[1])}')
ridge_scores["ridge_r2_score_feature_2"] = r2_score_feature_2

scores['Ridge'] = ridge_scores
```

Коэффициент детерминации для признака "Модуль упругости при растяжении, ГПа": 0.7689546971074563  
Коэффициент детерминации для признака "Прочность при растяжении, МПа": 0.9547639267605037

Рисунок 23 – Обучение моделей

Визуализация полученного результата для обоих признаков показана на рисунках 24-25

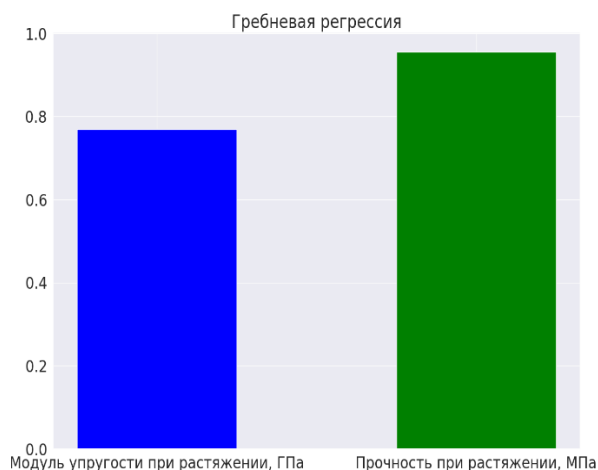


Рисунок 24 – Гистограмма Гребневая регрессия

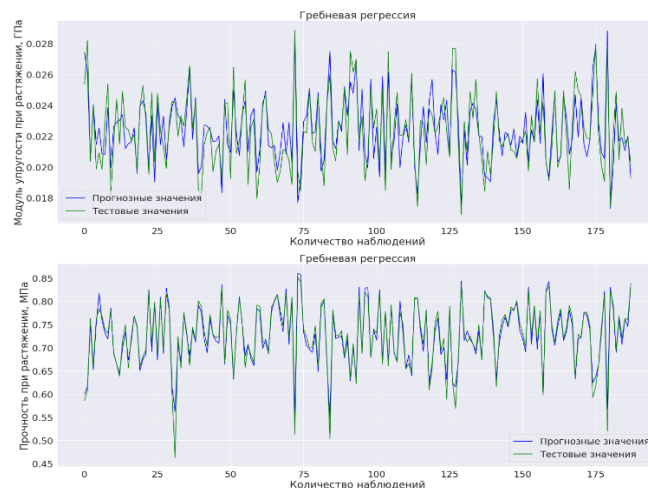


Рисунок 25 – График Гребневая регрессия

- **ElasticNet регрессия**

```
#Обучаем модель для предсказания признака "Модуль упругости при растяжении"
elastic.fit(X_train, y_train.iloc[:, 0])
y_pred.append(elastic.predict(X_test))
r2_score_feature_1 = r2_score(y_test.iloc[:, 0], y_pred[0])
print(f'Коэффициент детерминации для признака "{y_train.columns[0]}": {r2_score(y_test.iloc[:, 0], y_pred[0])}')
elastic_scores["elastic_r2_score_feature_1"] = r2_score_feature_1

#Обучаем модель для предсказания признака "Прочность при растяжении, МПа"
elastic.fit(X_train, y_train.iloc[:, 1])
y_pred.append(elastic.predict(X_test))
r2_score_feature_2 = r2_score(y_test.iloc[:, 1], y_pred[1])
print(f'Коэффициент детерминации для признака "{y_train.columns[1]}": {r2_score(y_test.iloc[:, 1], y_pred[1])}')
elastic_scores["elastic_r2_score_feature_2"] = r2_score_feature_2

scores['ElasticNetCV'] = elastic_scores

Коэффициент детерминации для признака "Модуль упругости при растяжении, ГПа": 0.77247495713938
Коэффициент детерминации для признака "Прочность при растяжении, МПа": 0.9563303066887645
```

Рисунок 26 – Обучение моделей

Визуализация полученного результата для обоих признаков показана на рисунках 27-28.

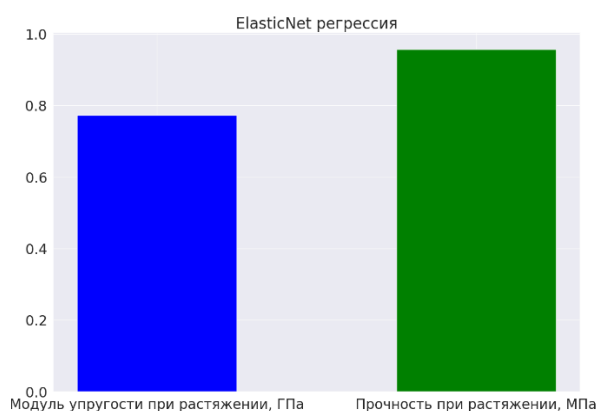


Рисунок 27 – Гистограмма ElasticNet регрессия

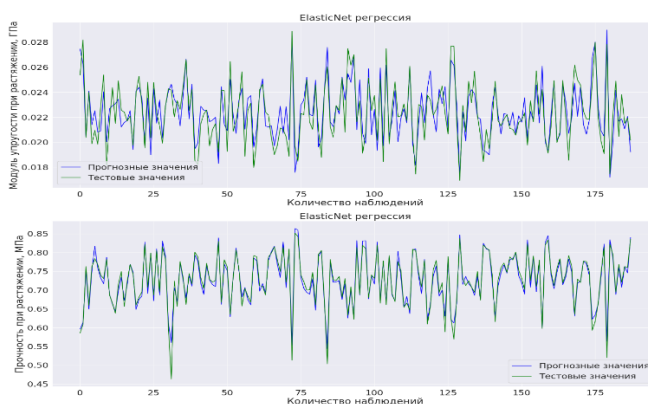


Рисунок 28 – График ElasticNet регрессия

- Случайный лес

Из-за специфической и высокодисперсной природы регрессии просто как задачи машинного обучения, регрессоры дерева решений следует тщательно обрезать. Для поиска оптимальной глубины дерева необходимо применить инструмент оптимизации поиска параметров GridSearchCV. Данный метод потребляет значительное количество ресурсов и его следует применять не для каждой задачи.

```
# Параметры, подаваемые на перекрестную проверку методом поиска по сетке
param_grid = { 'n_estimators': [100, 150, 200, 250, 300, 400, 500],
               'max_depth': [5, 7, 10, 15],
               'criterion': ['squared_error'] }

# Поиск лучшей комбинации гиперпараметров
clf = GridSearchCV(RandomForestRegressor(), param_grid, cv=10,
                  n_jobs=-1, verbose=1)

# Для первого признака
clf.fit(X_train, y_train.iloc[:, 0])
print(f"Лучшие параметры для предсказания признака 'Модуль упругости при растяжении' {clf.best_params_}")

# Для второго признака
clf.fit(X_train, y_train.iloc[:, 1])
print(f"Лучшие параметры для предсказания признака 'Прочность при растяжении, МПа' {clf.best_params_}")

Fitting 10 folds for each of 28 candidates, totalling 280 fits
Лучшие параметры для предсказания признака 'Модуль упругости при растяжении' {'criterion': 'squared_error', 'max_depth': 7, 'n_estimators': 100}
Fitting 10 folds for each of 28 candidates, totalling 280 fits
Лучшие параметры для предсказания признака 'Прочность при растяжении, МПа' {'criterion': 'squared_error', 'max_depth': 15, 'n_estimators': 500}
```

Рисунок 29 – Поиск лучшей комбинации гиперпараметров

Наилучшими параметрами для предсказания признака 'Модуль упругости при растяжении' являются:

- количество деревьев: 100

- глубина дерева: 7

для предсказания признака 'Прочность при растяжении, МПа' являются:

- количество деревьев: 500

- глубина дерева: 15

```
y_pred = []
rfr_scores = {} #Список коэффициентов детерминации для случайного леса

#Обучаем модель для предсказания признака "Модуль упругости при растяжении"

rfr = RandomForestRegressor(n_estimators=100, max_depth=7, random_state=1)
rfr.fit(X_train, y_train.iloc[:, 0])
y_pred.append(rfr.predict(X_test))
r2_score_feature_1 = r2_score(y_test.iloc[:, 0], y_pred[0])
print(f"Коэффициент детерминации для признака "{y_train.columns[0]}": {r2_score(y_test.iloc[:, 0], y_pred[0])}")
rfr_scores["rfr_r2_score_feature_1"] = r2_score_feature_1

#Обучаем модель для предсказания признака "Прочность при растяжении, МПа"

rfr = RandomForestRegressor(n_estimators=500, max_depth=15, random_state=1)
rfr.fit(X_train, y_train.iloc[:, 1])
y_pred.append(rfr.predict(X_test))
r2_score_feature_2 = r2_score(y_test.iloc[:, 1], y_pred[1])
print(f"Коэффициент детерминации для признака "{y_train.columns[1]}": {r2_score(y_test.iloc[:, 1], y_pred[1])}")
rfr_scores["rfr_r2_score_feature_2"] = r2_score_feature_2

scores['RandomForestRegressor'] = rfr_scores

Коэффициент детерминации для признака "Модуль упругости при растяжении, ГПа": 0.7929458253154105
Коэффициент детерминации для признака "Прочность при растяжении, МПа": 0.9508222000242285
```

Рисунок 30 – Коэффициенты детерминации для случайного леса



Визуализация полученного результата для обоих признаков показана на рисунках 31-32.

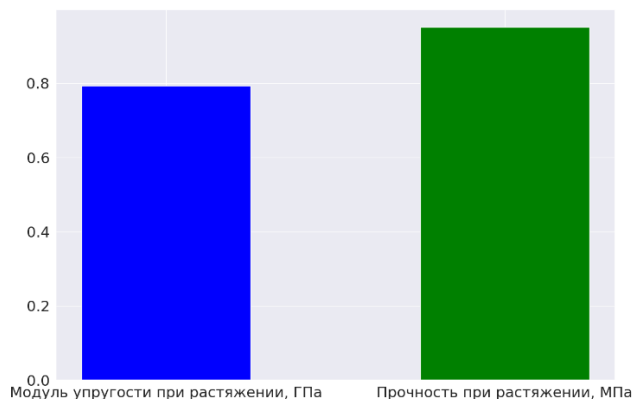


Рисунок 31 – Гистограмма Случайный лес

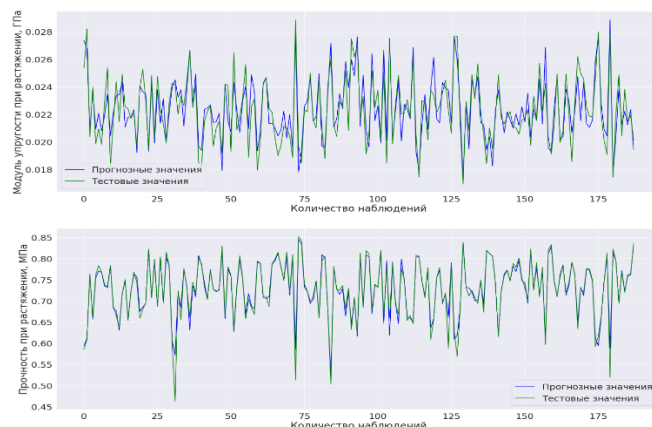


Рисунок 32 – График Случайный лес

- **к-ближайших соседей**

```
KNeighbors = list(range(1, 31))
param_grid = {'n_neighbors': range(1, 31)}

clf = GridSearchCV(KNeighborsRegressor(), param_grid, cv=10, n_jobs=-1, verbose=1)

#Для первого признака
clf.fit(X_train, y_train.iloc[:, 0])
print(f"Лучшие параметры для предсказания признака 'Модуль упругости при растяжении' {clf.best_params_}")

#Для второго признака
clf.fit(X_train, y_train.iloc[:, 1])
print(f"Лучшие параметры для предсказания признака 'Прочность при растяжении, МПа' {clf.best_params_}")

Fitting 10 folds for each of 30 candidates, totalling 300 fits
Лучшие параметры для предсказания признака 'Модуль упругости при растяжении' {'n_neighbors': 10}
Fitting 10 folds for each of 30 candidates, totalling 300 fits
Лучшие параметры для предсказания признака 'Прочность при растяжении, МПа' {'n_neighbors': 4}
```

Наилучшими параметрами для предсказания признака 'Модуль упругости при растяжении' являются:

-количество соседей: 10

для предсказания признака 'Прочность при растяжении, МПа' являются:

-количество соседей: 4

```

y_pred = []
knn_scores = {} #Список коэффициентов детерминации для k-ближайших соседей

#Обучаем модель для предсказания признака "Модуль упругости при растяжении"
knn = KNeighborsRegressor(n_neighbors=10)
knn.fit(X_train, y_train.iloc[:, 0])
y_pred.append(knn.predict(X_test))
r2_score_feature_1 = r2_score(y_test.iloc[:, 0], y_pred[0])
print(f'Коэффициент детерминации для признака "{y_train.columns[0]}": {r2_score(y_test.iloc[:, 0], y_pred[0])}')
knn_scores["knn_r2_score_feature_1"] = r2_score_feature_1

#Обучаем модель для предсказания признака "Прочность при растяжении, МПа"
knn = KNeighborsRegressor(n_neighbors=4)
knn.fit(X_train, y_train.iloc[:, 1])
y_pred.append(knn.predict(X_test))
r2_score_feature_2 = r2_score(y_test.iloc[:, 1], y_pred[1])
print(f'Коэффициент детерминации для признака "{y_train.columns[1]}": {r2_score(y_test.iloc[:, 1], y_pred[1])}')
knn_scores["knn_r2_score_feature_2"] = r2_score_feature_2

scores['KNeighborsRegressor'] = knn_scores

Коэффициент детерминации для признака "Модуль упругости при растяжении, ГПа": 0.7326668096474398
Коэффициент детерминации для признака "Прочность при растяжении, МПа": 0.9458498207697015

```

Рисунок 33 – Коэффициенты детерминации для k-ближайших соседей

Визуализация полученного результата для обоих признаков показана на рисунках 34-35.

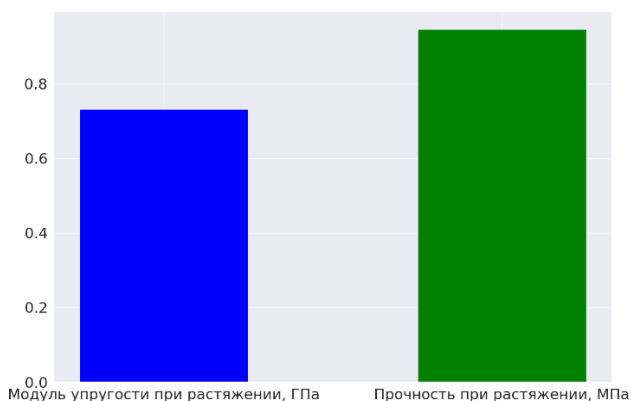


Рисунок 34 – Гистограмма Случайный лес

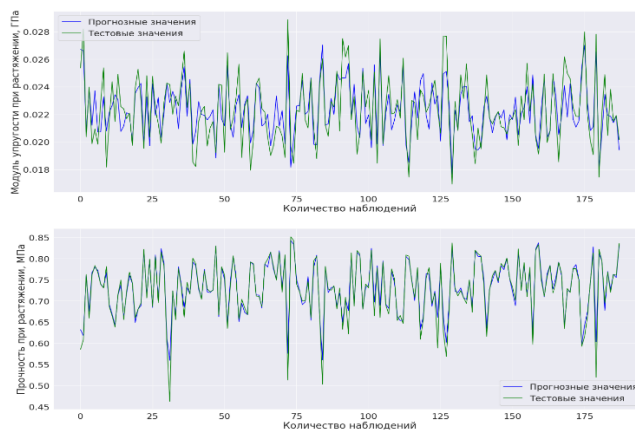


Рисунок 35 – График Случайный лес

## 2.3 Тестирование модели

Выбор лучшей модели из ранее представленных с помощью команды `sorted(scores)` для обоих признаков:

```

#Выведем оценки для предсказаний двух целевых признаков для каждой из моделей,
# где первым признаком является "Модуль упругости при растяжении", вторым - "Прочность при растяжении, МПа"
for key in sorted(scores):
    print(scores[key])

#Если бы моделей было больше, определить лучшую модель на глаз не представилось бы простой задачей. Найдем лучшую модель для
# каждого из признаков путем визуализации оценок предсказаний.

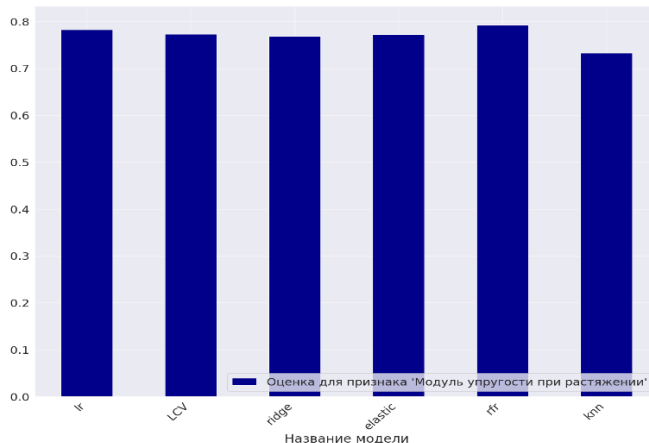
{'elastic_r2_score_feature_1': 0.772474957713938, 'elastic_r2_score_feature_2': 0.9563303066887645}
{'knn_r2_score_feature_1': 0.7326668096474398, 'knn_r2_score_feature_2': 0.9458498207697015}
{'LCV_r2_score_feature_1': 0.7735465492598215, 'LCV_r2_score_feature_2': 0.9563546747418563}
{'lr_r2_score_feature_1': 0.7826749174271698, 'lr_r2_score_feature_2': 0.9552506002027553}
{'rfr_r2_score_feature_1': 0.7929458253154105, 'rfr_r2_score_feature_2': 0.9588222000242285}
{'ridge_r2_score_feature_1': 0.7689546971074563, 'ridge_r2_score_feature_2': 0.9547639267605037}

```

Рисунок 36 – Тестирование моделей

```
#Создаем датафрейм для визуализации разделения
feature_1 = pd.DataFrame.from_dict(feature_1, orient='index',
                                   columns=["Оценка для признака 'Модуль упругости при растяжении'"])
feature_1
```

Оценка для признака 'Модуль упругости при растяжении'	
lr	0.782675
LCV	0.773547
ridge	0.768955
elastic	0.772475
rfr	0.792946
knn	0.732667

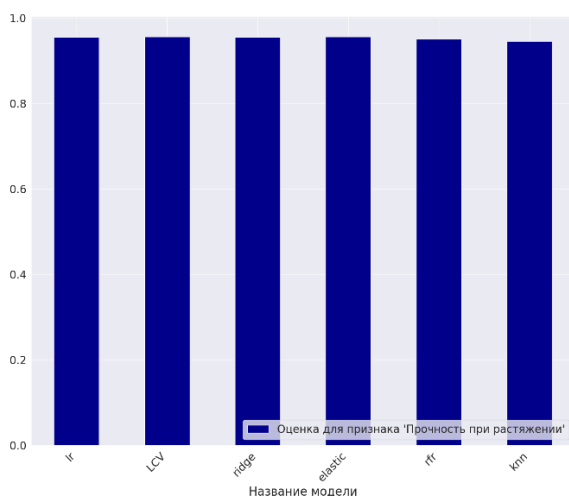


Визуализируем оценки по каждой модели для первого признака. Лучшей моделью для предсказания признака "Модуль упругости при растяжении" является модель случайного леса рисунок 37.

Рисунок 37- Оценка тестирования моделей для признака «Модуль упругости при растяжении»

```
feature_2 = pd.DataFrame.from_dict(feature_2, orient='index',
                                   columns=["Оценка для признака 'Прочность при растяжении'"])
feature_2
```

Оценка для признака 'Прочность при растяжении'	
lr	0.955251
LCV	0.956355
ridge	0.954764
elastic	0.956330
rfr	0.950822
knn	0.945850



Визуализируем оценки по каждой модели для второго признака. Так как с предсказанием второго признака модели справились практически одинаково, оптимальной моделью, для удобства будем считать также модель случайного леса рисунок 38.

Рисунок 38- Оценка тестирования моделей для признака «Прочность при растяжении»

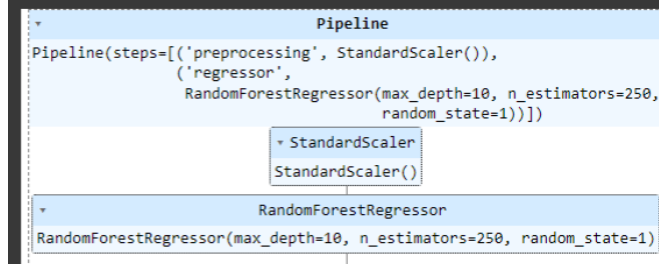
## Сохранение модели на диск

Еще раз обучим каждую из моделей перед сохранением на диск.

Так как данные, которые будут поступать в модель машинного обучения, встроенную в приложение будут являться "сырыми", а модель будет обучена на нормализованных данных, модель будет плохо предсказывать целевые признаки. Для решения этой проблемы необходимо создать рабочий поток обработки данных и обучения модели. Будем использовать метод `make_pipeline`. В качестве метода преобразования данных будем использовать метод `StandardScaler()` для лучшей сходимости алгоритма на новых данных рисунок 39.

```
# Набор параметров для предсказания первого признака "Модуль упругости при растяжении"
pipe_rfr_elasticity = Pipeline([('preprocessing', StandardScaler()),
                                ('regressor', RandomForestRegressor(n_estimators=500, max_depth=7, random_state=1))])
pipe_rfr_elasticity.fit(X_train, y_train.iloc[:, 0])

# Набор параметров для предсказания второго признака "Прочность при растяжении, МПа"
pipe_rfr_strength = Pipeline([('preprocessing', StandardScaler()),
                               ('regressor', RandomForestRegressor(n_estimators=250, max_depth=10, random_state=1))])
pipe_rfr_strength.fit(X_train, y_train.iloc[:, 1])
```



```
[86] #Проверяем работоспособность моделей для каждого из признаков
y_pred_elasticity_before = pipe_rfr_elasticity.predict(X_test)
y_pred_elasticity_before[0]

0.027424349836562213

[87] y_pred_strength_before = pipe_rfr_strength.predict(X_test)
y_pred_strength_before[0]

0.5945697610494617
```

```
# Сохраняем первую модель
with open('/models/pipe_rfr_elasticity.pkl', 'wb') as outp:
    pickle.dump(pipe_rfr_elasticity, outp)

# Сохраняем вторую модель
with open('/models/pipe_rfr_strength.pkl', 'wb') as outp:
    pickle.dump(pipe_rfr_strength, outp)

[95] # Загружаем модели и проверяем их работоспособность
with open('/models/pipe_rfr_elasticity.pkl', 'rb') as inp:
    pipe_rfr_elasticity = pickle.load(inp)

y_pred_elasticity_after = pipe_rfr_elasticity.predict(X_test)

# В данной строке мы проверяем каждое старое прогнозируемое значение с новым. Функция all возвращает true, если логическая маска
# целиком состоит из значений true.
print((y_pred_elasticity_before == y_pred_elasticity_after).all())

True

with open('/models/pipe_rfr_strength.pkl', 'rb') as inp:
    pipe_rfr_strength = pickle.load(inp)

y_pred_strength_after = pipe_rfr_strength.predict(X_test)
print((y_pred_strength_before == y_pred_strength_after).all())

# Обе модели работают корректно после сохранения и загрузки в ноутбук с локального диска.

True
```

Рисунок 39- Использование методов StandardScaler, сохранение и выгрузка моделей

## **2.4 Нейронная сеть, которая будет рекомендовать соотношение матрица-наполнитель.**

### **2.4.1 Обучение ИНС для целевого признака "Модуль упругости при растяжении"**

По условию задачи, также, необходимо реализовать алгоритм на основе искусственных нейронных сетей. Нейронные сети обычно используют для классификации. Сигналы проходят через слои нейронов и обобщаются в один из нескольких классов. Однако их можно адаптировать в регрессионные модели, если изменить последнюю функцию активации.

Заменив последнюю функцию активации (выходной нейрон) линейной функцией активации, выходной сигнал можно отобразить на множество значений, выходящих за пределы фиксированных классов.

Для решения поставленной задачи с большим количеством регрессоров (независимых переменных) следует использовать полносвязную нейронную сеть, с целью попытаться выявить дополнительные связи между признаками, ввиду низкой коллинеарности между ними.

Для построения полносвязной искусственной нейронной сети будем использовать библиотеку Keras рисунок 40.

```
model = Sequential()
```

Так как преобразованные данные в датасете находятся в диапазоне [0..1], т.е. не имеют отрицательных значений, при построении архитектуры нейронной сети будем использовать активационную функцию ReLu.

```
model.add(Dense(128, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(16, activation='softmax'))
```

Выходной слой с одним линейным нейроном для решения задачи регрессии

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

#Архитектура построенной нейронной сети  
model.summary()

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	1536
dense_1 (Dense)	(None, 128)	16512
dense_2 (Dense)	(None, 64)	8256
dense_3 (Dense)	(None, 32)	2080
dense_4 (Dense)	(None, 16)	528
dense_5 (Dense)	(None, 1)	17

=====  
Total params: 28,929  
Trainable params: 28,929  
Non-trainable params: 0

Рисунок 40 — Архитектура нейросети в виде summary

```
[101] #Оценка обученной модели на тестовом наборе для первого целевого признака
result = model.evaluate(X_test, y_test.iloc[:, 0])
print(f"loss: {result[0]} | mae: {result[1]}")

6/6 [=====] - 2s 5ms/step - loss: 0.0042 - mae: 0.0645
loss: 0.00416568061336875 | mae: 0.06450531631708145

#Обучаем модель для первого признака
history_first = model.fit(X_train,
                          y_train.iloc[:, 0],
                          epochs=100,
                          validation_split=0.1,
                          verbose=1)

22/22 [=====] - 0s 5ms/step - loss: 4.4300e-06 - mae: 0.0018 - val_loss: 2.3139e-06 - val_mae: 0.0011
Epoch 4/100
22/22 [=====] - 0s 4ms/step - loss: 3.7194e-06 - mae: 0.0015 - val_loss: 1.9124e-06 - val_mae: 0.0010
Epoch 5/100
22/22 [=====] - 0s 4ms/step - loss: 3.3739e-06 - mae: 0.0015 - val_loss: 1.6809e-06 - val_mae: 9.6592e-04
Epoch 6/100
22/22 [=====] - 0s 4ms/step - loss: 2.7986e-06 - mae: 0.0013 - val_loss: 1.6119e-06 - val_mae: 0.0010
Epoch 7/100
22/22 [=====] - 0s 5ms/step - loss: 2.3925e-06 - mae: 0.0012 - val_loss: 2.3179e-06 - val_mae: 0.0013
Epoch 8/100
22/22 [=====] - 0s 5ms/step - loss: 2.5503e-06 - mae: 0.0013 - val_loss: 1.6482e-06 - val_mae: 0.0010
Epoch 9/100
22/22 [=====] - 0s 5ms/step - loss: 2.0731e-06 - mae: 0.0012 - val_loss: 1.3862e-06 - val_mae: 9.3862e-04
Epoch 10/100
22/22 [=====] - 0s 4ms/step - loss: 1.9099e-06 - mae: 0.0011 - val_loss: 1.3975e-06 - val_mae: 9.3401e-04
Epoch 11/100
22/22 [=====] - 0s 4ms/step - loss: 1.8009e-06 - mae: 0.0011 - val_loss: 1.3501e-06 - val_mae: 9.2131e-04
Epoch 12/100
22/22 [=====] - 0s 4ms/step - loss: 1.7909e-06 - mae: 0.0011 - val_loss: 1.3473e-06 - val_mae: 9.2712e-04
Epoch 13/100
22/22 [=====] - 0s 4ms/step - loss: 1.6822e-06 - mae: 0.0010 - val_loss: 1.5746e-06 - val_mae: 9.8915e-04
Epoch 14/100
22/22 [=====] - 0s 4ms/step - loss: 1.6410e-06 - mae: 0.0010 - val_loss: 1.6034e-06 - val_mae: 0.0010
0 сек. выполнено в 00:34
```

Рисунок 41 — Обучение модели для первого признака

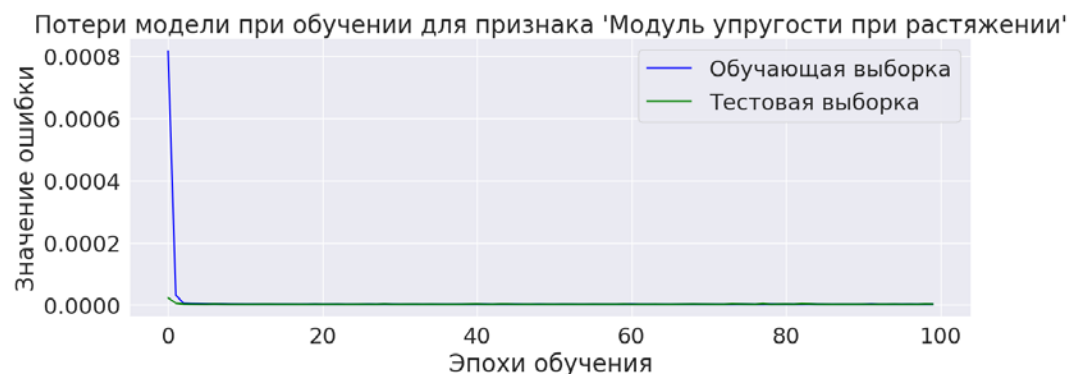


Рисунок 42 – Потери модели при обучении для первого признака

Визуализируем график потерь модели при обучении для первого признака рисунок 43.

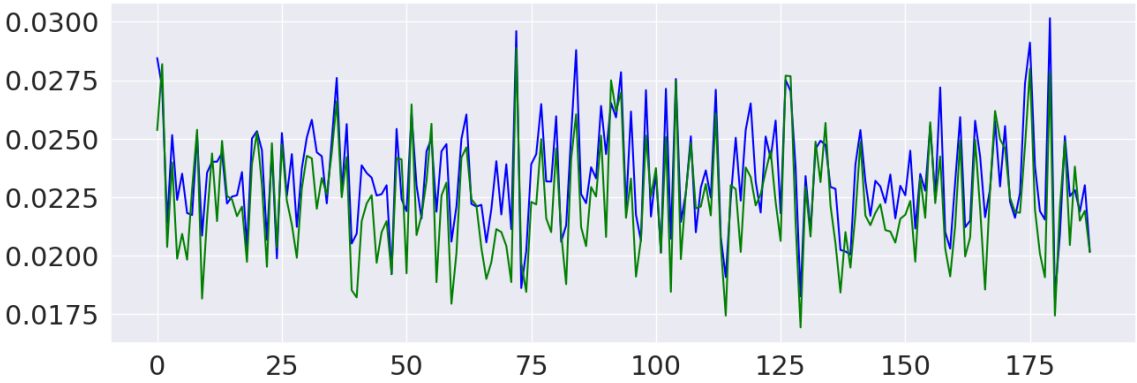


Рисунок 43 - График потерь модели при обучении для первого признака

Визуализируем качество обобщения искусственной нейронной сети на тестовых данных для каждого признака рисунок 44

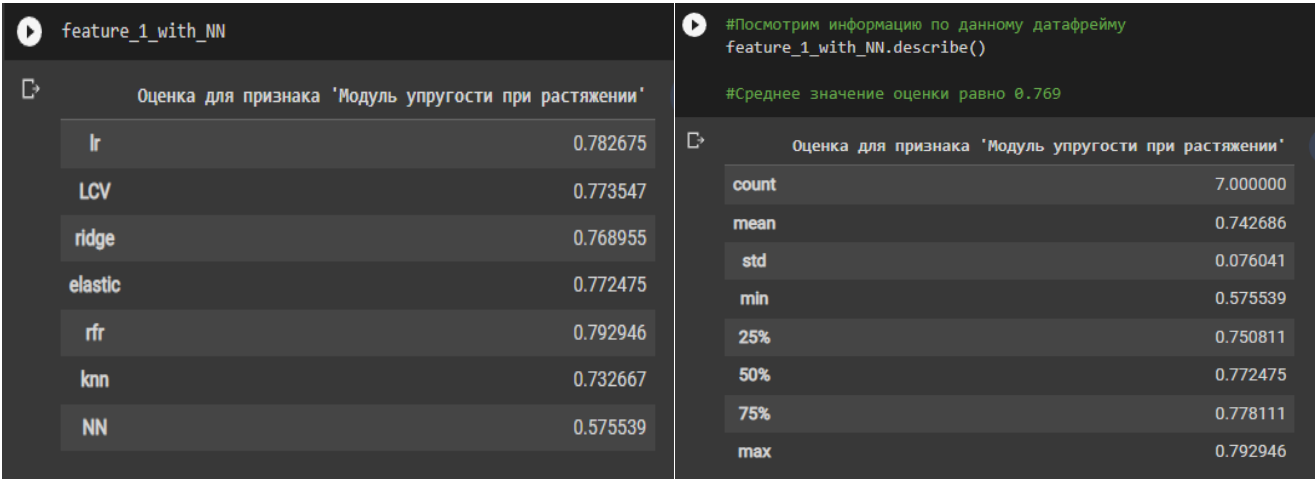


Рисунок 44 - Качество обобщения для ИНС для первого признака

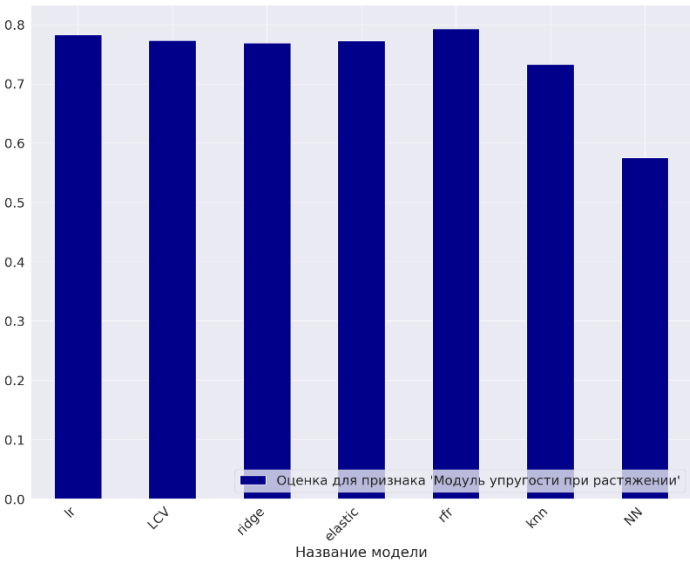


Рисунок 45- Оценка тестирования моделей для первого признака



После визуализации оценки по каждой модели для первого признака, лидером по-прежнему остается модель случайного леса.

## 2.4.2 Обучение ИНС для целевого признака "Прочность при растяжении"

Для оценки второго признака создадим отдельную архитектуру:

```
model2 = Sequential()
```

```
model2.add(Dense(128, input_dim=X_train.shape[1], activation='relu'))
```

```
model2.add(Dense(128, activation='relu'))
```

```
model2.add(Dense(64, activation='relu'))
```

```
model2.add(Dense(32, activation='relu'))
```

```
model2.add(Dense(16, activation='relu'))
```

```
model2.add(Dense(1, activation='linear'))
```

```
[116] # Оценка обученной модели на тестовом наборе для второго целевого признака
result = model2.evaluate(X_test, y_test.iloc[:, 1])
print(f'loss: {result[0]} | mae: {result[1]}')

6/6 [=====] - 0s 2ms/step - loss: 0.5640 - mae: 0.7479
loss: 0.5639968514442444 | mae: 0.7478866577148438

Обучаем модель для второго признака
history_second = model2.fit(X_train,
                             y_train.iloc[:, 1],
                             epochs=100,
                             validation_split=0.1)

Epoch 72/100
22/22 [=====] - 0s 4ms/step - loss: 9.9747e-06 - mae: 0.0022 - val_loss: 2.3415e-05 - val_mae: 0.0036
Epoch 73/100
22/22 [=====] - 0s 5ms/step - loss: 9.0989e-06 - mae: 0.0021 - val_loss: 1.5951e-05 - val_mae: 0.0026
Epoch 74/100
22/22 [=====] - 0s 4ms/step - loss: 1.0086e-05 - mae: 0.0023 - val_loss: 6.8906e-05 - val_mae: 0.0072
Epoch 75/100
22/22 [=====] - 0s 4ms/step - loss: 1.7588e-05 - mae: 0.0031 - val_loss: 2.0022e-05 - val_mae: 0.0027
Epoch 76/100
22/22 [=====] - 0s 4ms/step - loss: 8.7216e-06 - mae: 0.0021 - val_loss: 1.6766e-05 - val_mae: 0.0025
Epoch 77/100
22/22 [=====] - 0s 4ms/step - loss: 9.0869e-06 - mae: 0.0023 - val_loss: 2.7550e-05 - val_mae: 0.0037
Epoch 78/100
22/22 [=====] - 0s 4ms/step - loss: 1.6670e-05 - mae: 0.0033 - val_loss: 1.8211e-05 - val_mae: 0.0024
Epoch 79/100
22/22 [=====] - 0s 4ms/step - loss: 8.9504e-06 - mae: 0.0022 - val_loss: 1.4810e-05 - val_mae: 0.0022
Epoch 80/100
22/22 [=====] - 0s 4ms/step - loss: 9.6363e-06 - mae: 0.0023 - val_loss: 1.7141e-05 - val_mae: 0.0030
Epoch 81/100
22/22 [=====] - 0s 4ms/step - loss: 9.7163e-06 - mae: 0.0023 - val_loss: 1.4099e-05 - val_mae: 0.0021
Epoch 82/100
22/22 [=====] - 0s 4ms/step - loss: 8.8231e-06 - mae: 0.0022 - val_loss: 2.0670e-05 - val_mae: 0.0029
Epoch 83/100
22/22 [=====] - 0s 4ms/step - loss: 8.8623e-06 - mae: 0.0022 - val_loss: 1.5203e-05 - val_mae: 0.0028
Epoch 84/100
22/22 [=====] - 0s 4ms/step - loss: 8.1634e-06 - mae: 0.0021 - val_loss: 2.2470e-05 - val_mae: 0.0030
Epoch 85/100
0 сек. выполнено в 00:34
```

Рисунок 46 — Обучение модели для второго признака



Рисунок 47 – Потери модели при обучении для второго признака

Визуализируем график потерь модели при обучении для второго признака  
рисунок 48

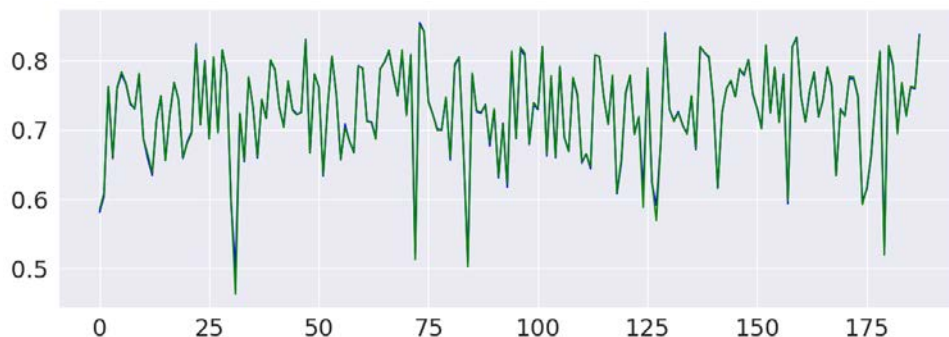


Рисунок 48- График потерь модели при обучении для второго признака

Визуализируем качество обобщения искусственной нейронной сети на тестовых данных для каждого признака рисунок 49.

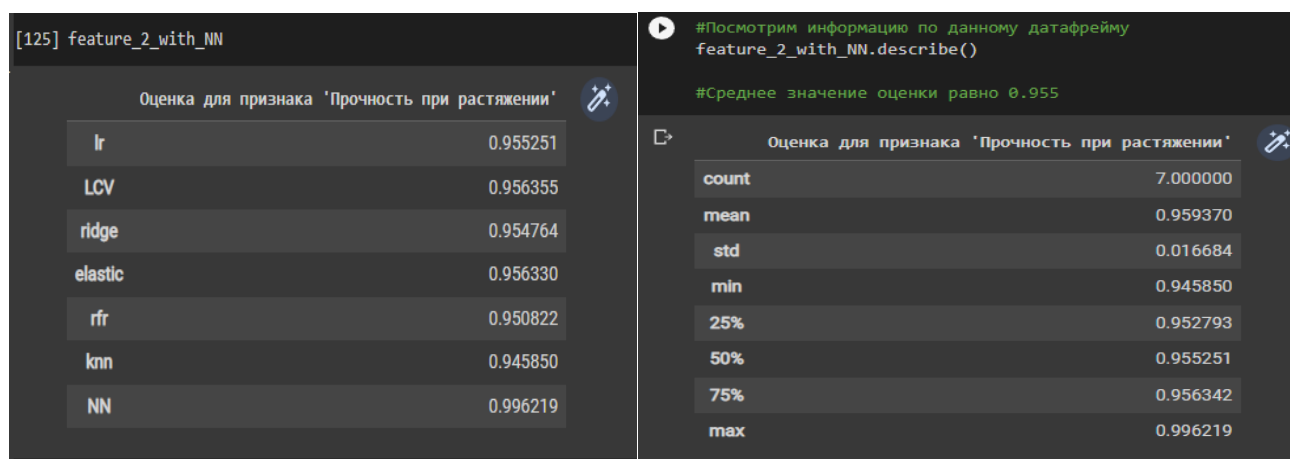
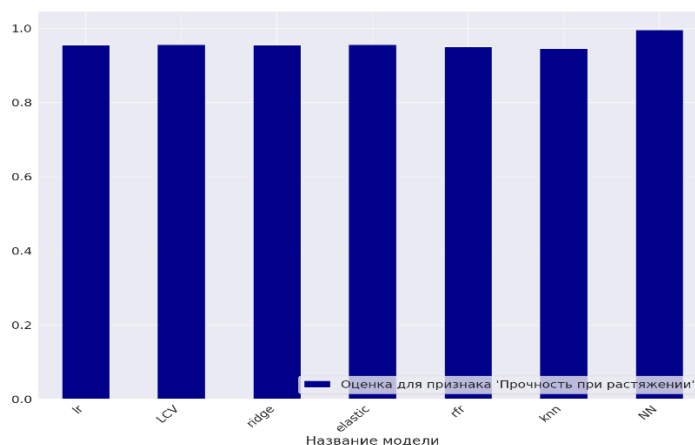


Рисунок 49 - Качество обобщения для ИНС для второго признака

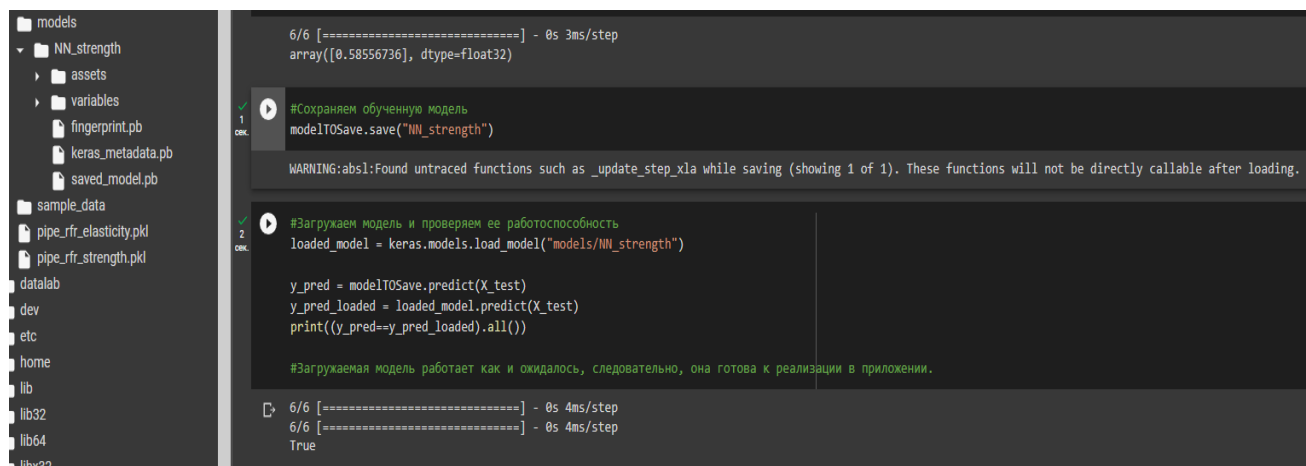
Визуализируем оценки по каждой модели для второго признака  
рисунок 50.



## Рисунок 50- Оценка тестирования моделей для второго признака

Визуально анализируя оценку тестирования моделей для второго признака можно отметить, что искусственная нейронная сеть выдает более высокие показатели, чем модель случайного леса.

После обучение ИНС, необходимо сохранить полученные результаты по моделям, затем загрузить ее и проверить работоспособность. На рисунке 51 видны манипуляции, что модели работают и можно приступать к созданию приложения.



```
6/6 [=====] - 0s 3ms/step
array([[0.58556736], dtype=float32])

1
OK
#Сохраняем обученную модель
modelToSave.save("NN_strength")

WARNING:absl:Found untraced functions such as _update_step_xla while saving (showing 1 of 1). These functions will not be directly callable after loading.

2
OK
#Загружаем модель и проверяем ее работоспособность
loaded_model = keras.models.load_model("models/NN_strength")

y_pred = modelToSave.predict(X_test)
y_pred_loaded = loaded_model.predict(X_test)
print((y_pred==y_pred_loaded).all())

#Загружаемая модель работает как и ожидалось, следовательно, она готова к реализации в приложении.

6/6 [=====] - 0s 4ms/step
6/6 [=====] - 0s 4ms/step
True
```

## Рисунок 51 – Часть кода, сохранения обучения ИНС

На данном этапе исследовательскую работу можно считать завершенной.

На основании созданных и обученных моделей будет реализовано веб-приложение для предсказания двух приведенных выше целевых признаков, с интуитивно-понятным пользователю интерфейсом.

## 2.5 Разработка приложения

В приложении необходимо реализовать следующие функции:

- выбор целевой переменной для предсказания;
- ввод входных параметров;
- проверка введенных параметров;
- загрузка сохраненной модели, получение и отображение прогноза выходных параметров.

Решено разработать веб-приложение с помощью языка Python, фреймворка Flask и VSCode.

Эту задачу получилось решить.

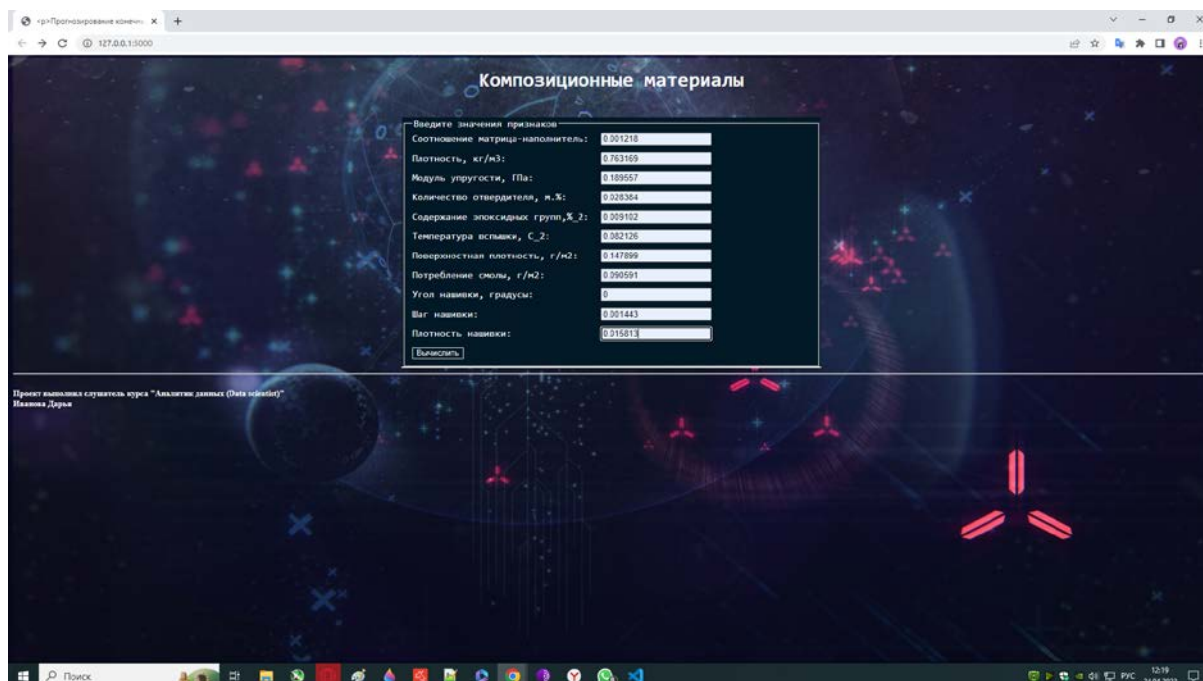


Рисунок 52 – Созданное веб-приложение

### 3. Создание удаленного репозитория

Для данного исследования был создан удаленный репозиторий на GitHub, который находится по адресу <https://github.com/DariaIvanova007>. На него были загружены результаты работы: исследовательский notebook, код приложения.

## **Заключение**

По результатам исследования удалось выяснить, что входной датасет имеет некоторое количество выбросов, что означает необходимость тщательной проверки входных данных на превышение граничных значений в дальнейшем. Результат обучения моделей на первом целевом признаке - 'Модуль упругости при растяжении' можно считать удовлетворительным, но недостаточным для использования в промышленных целях, так как модели хоть и предсказывают значение признака лучше, чем простое подбрасывание монеты, но делают это плохо (Средний коэффициент детерминации равен 0.77, где максимальное значение равняется 1). Обучившись предсказывать второй признак, каждая из моделей показала хороший результат, со средним значением коэффициента равным 0.95. Из этого следует, что для применения в промышленной среде можно использовать модель предсказания второго целевого признака - "Прочность при растяжении".

В ходе выполнения ВКР были изучены способы анализа и предобработки данных. Построенные модели показали, что исходный датасет является предобработанным и не содержит реальных значений для отработки обучения и тренировки моделей.

Полученная модель нейронной сети не идеальна, но позволяет предсказывать значения, близкие к средним значениям параметров.

## Библиографический список

1. Костиков, В. И. Технология композиционных материалов: учебное пособие / В. И. Костиков, Ж. В. Еремеева. – Москва; Вологда : Инфра-Инженерия, 2021. – 484 с. ISBN 978-5-9729-0520-1.
2. Документация по языку программирования python: – Режим доступа: <https://docs.python.org/3.8/index.html>.
3. Документация по библиотеке numpy: – Режим доступа: <https://numpy.org/doc/1.22/user/index.html#user>.
4. Документация по библиотеке pandas: – Режим доступа: [https://pandas.pydata.org/docs/user\\_guide/index.html#user-guide](https://pandas.pydata.org/docs/user_guide/index.html#user-guide).
5. Документация по библиотеке matplotlib: – Режим доступа: <https://matplotlib.org/stable/users/index.html>.
6. Документация по библиотеке seaborn: – Режим доступа: <https://seaborn.pydata.org/tutorial.html>.
7. Документация по библиотеке sklearn: – Режим доступа: [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html).
8. Документация по библиотеке keras: – Режим доступа: <https://keras.io/api/>.
9. Руководство по быстрому старту в flask: – Режим доступа: <https://flask-russian-docs.readthedocs.io/ru/latest/quickstart.html>.
10. Программирование, тестирование, проектирование, нейросети, технологии аппаратно-программных средств (практические задания и способы их решения): учебник: [16+] / С. В. Веретехина, К. С. Кармицкий, Д. Д. Лукашин [и др.]. – Москва: Директ-Медиа, 2022. – 144 с. ISBN 978-5-4499-3321-8. – Текст: электронный.
11. Вошиков, А. Д. Автоматизация получения данных и выбора в исключительных выборках на территории ФГБУ "НЦБРП": [16+] / А. Д. Вошиков ; Московский финансово-промышленный университет «Синергия». – Москва: б.и., 2022. – 121 с. по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=694003> (дата обращения: 23.04.2023). – Текст: электронный.