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 Pro»

Слушатель Токарева Людмила Евгеньевна

Москва, 2023

1. Содержание

Введение .......................................................................................................... 4

1. Аналитическая часть .................................................................................. 5

1.1. Постановка задачи ................................................................................... 5

1.2. Описание используемых методов ......................................................... 23

1.3. Разведочный анализ данных .................................................................. 25

1.3.1. Выбор признаков ................................................................................. 26

1.3.2. Ход решения задачи ............................................................................. 26

1.3.3. Препроцессинг ..................................................................................... 27

1.3.4. Метрики качества моделей ................................................................. 27

2. Практическая часть ................................................................................... 28

2.1. Разбиение и предобработка данных ..................................................... 28

2.2. Разработка и обучение модели. Выбор лучшей модели ..................... 28

2.3. Разработка приложения…………………………….............................. 39

2.4. Состав проекта………..…………………………….............................. 41

2.4.1. Наименование и назначение SQL-скриптов...................................... 41

2.4.2. Наименование и назначение python-скриптов.................................. 42

2.5. Создание удаленного репозитория ....................................................... 45

Библиографический список ......................................................................... 46

1. Введение

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

Композиционные материалы — это искусственно созданные материалы, состоящие из нескольких других с четкой границей между ними. Композиты обладают теми свойствами, которые не наблюдаются у компонентов по отдельности. При этом композиты являются монолитным материалом, т.е. компоненты материала неотделимы друг от друга без разрушения конструкции в целом. Яркий пример композита — железобетон. Бетон прекрасно сопротивляется сжатию, но плохо растяжению. Стальная арматура внутри бетона компенсирует его неспособность сопротивляться сжатию, формируя тем самым новые, уникальные свойства. Современные композиты изготавливаются из других материалов: полимеры, керамика, стеклянные и углеродные волокна, но данный принцип сохраняется. У такого подхода есть и недостаток: даже если мы знаем характеристики исходных компонентов, определить характеристики композита, состоящего из этих компонентов, достаточно проблематично. Для решения этой проблемы есть два пути: физические испытания образцов материалов или прогнозирование характеристик. Суть прогнозирования заключается в симуляции представительного элемента объема композита на основе данных о характеристиках входящих компонентов (связующего и армирующего компонента).

* + 1. На входе имеются данные о начальных свойствах компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.). На выходе необходимо спрогнозировать ряд конечных свойств получаемых композиционных материалов.

1. Созданные прогнозные модели помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов возможными новыми характеристиками материалов и цифровыми двойниками новых композитов.
2. 1. Аналитическая часть
   1. 1.1. Постановка задачи
      1. В данной работе исследуется композит, датасет которого основан на реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана). Датасет представлен с заданием в виде архива - hw\_data\_composite.zip

От специалиста Data Scientist (учащегося) требуется:

1. Обучить алгоритм машинного обучения, который будет определять значения:

* Модуль упругости при растяжении, ГПа;
* Прочность при растяжении, МПа.

1. Написать нейронную сеть, которая будет рекомендовать:

* Соотношение матрица-наполнитель.

1. Написать приложение, которое будет выдавать прогноз, полученный в задании 1 или 2 (один или два прогноза, на выбор).
2. Сделать commit приложения на github.com.

Распаковав hw\_data\_composite.zip обнаруживаем, что датасет состоит из двух Excel-файлов X\_bp.xlsx и Х\_nup.xlsx.

Методом визуального наблюдения устанавливаем, что данные в файлах X\_bp.xlsx и Х\_nup.xlsx являются структурированными. Каждое свойство хранится в отдельной колонке (10 и 3 колонок данных соответственно), значения представляет собой числа с десятичным разделителем. Первая колонка – индекс (целое число). Первая строка содержит наименование свойств, указанных в колонках.

Так как данные являются структурированными, то для первичной оценки качества данных и получения статистической информации можем использовать базу данных PostgreSQL и инструмент pgAdmin. Далее будет показано, что для первичной оценки и анализа данных эффективней использовать специализированные библиотеки Python (pandas и seaborn), а возможность использования PostgreSQL указана с целью демонстрации статистических функций базы данных PostgreSQL. Эти две возможности будут рассматриваться в параллель, а использованные методы для этого будут показаны в сравнительной таблице (Таблица 1).

Для использования базы данных PostgreSQL для анализа и получения статистической информации необходимо создать таблицы X\_bp и X\_nup командой CREATE TABLE*.*  Имена полей напишем в английской раскладке, а в описании поля укажем наименование согласно исходному файлу данных.

Так как штанные средства pgAdmin позволяют загружать данные из текстовых файлов csv, то с помощью Excel преобразуем файлы X\_bp.xlsx и X\_nup.xlsx в текстовые файлы данных X\_bp.csv и X\_nup.csv соответственно, и далее в X\_bp4pg.csv и X\_nup4pg.csv (заменив знак десятичного разделителя, что потребовалось в моей среде, учитывая национальные настройки), затем импортируем данные в базу PostgreSQL «в соответствии с рисунком 1 и рисунком 2» с помощью инструмента pgAdmin и с помощью встроенных агрегатных функций проведем анализ каждой из таблиц.

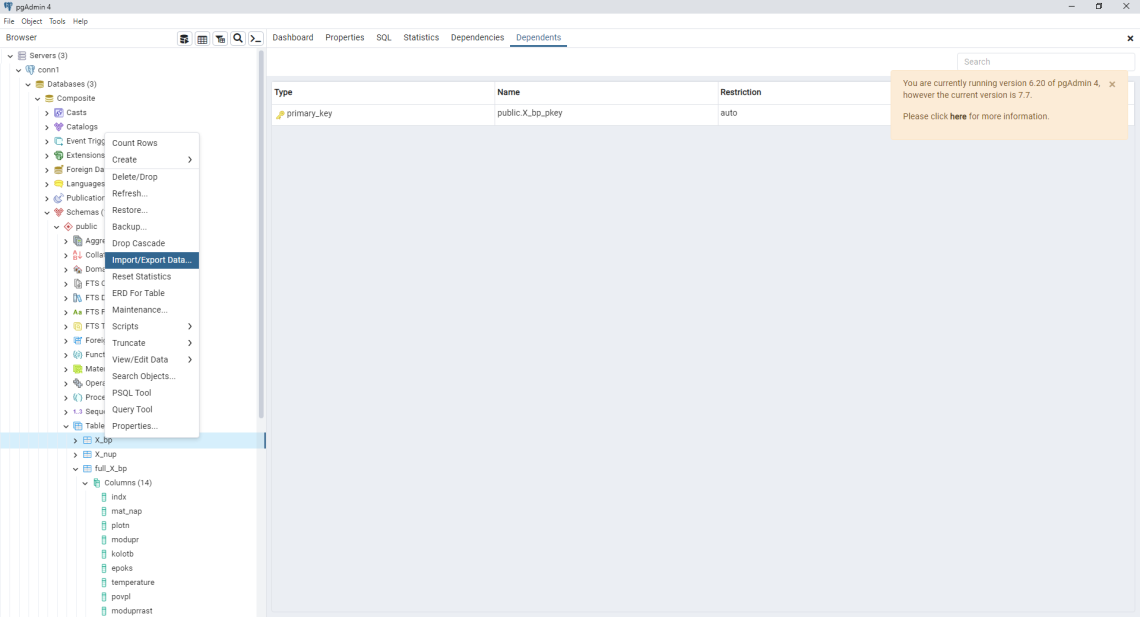


Рисунок 1 – Импорт данных в PostgreSQL

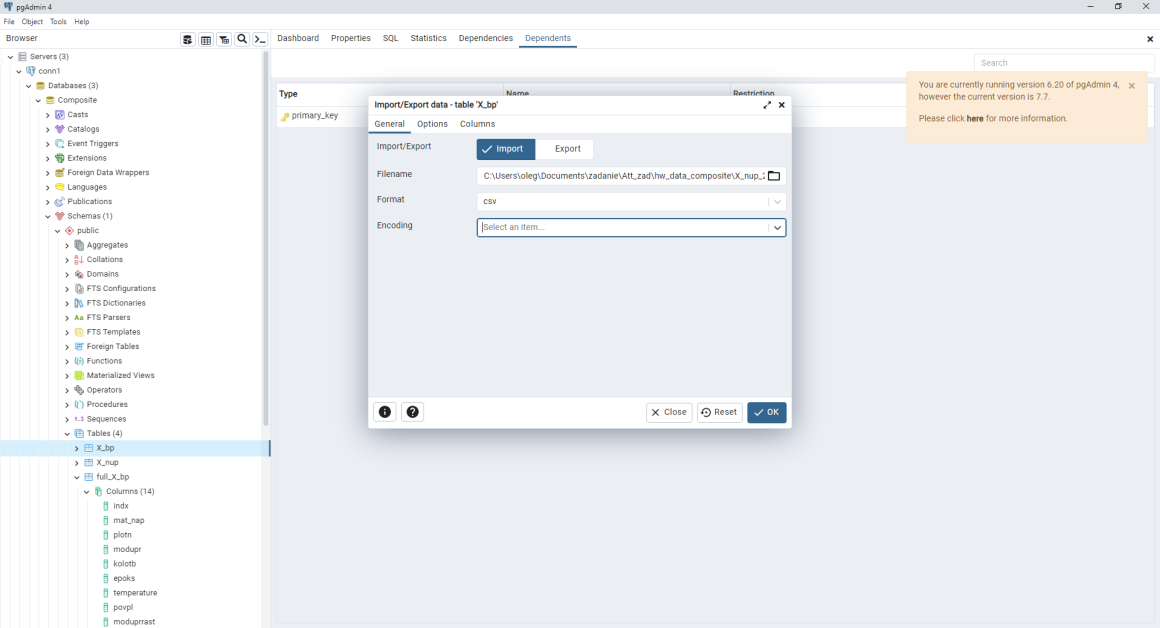


Рисунок 2 – Импорт данных из csv - файла в PostgreSQL

Согласно условию задачи необходимо объединить два датасета по индексу методом INNER JOIN – объединения «в соответствии с рисунком 3» и таким образом получить новый датасет full\_X\_bp с 13 полями - свойствами (признаками), и провести полный анализ его свойств и статистических показателей «в соответствии с рисунком 4».

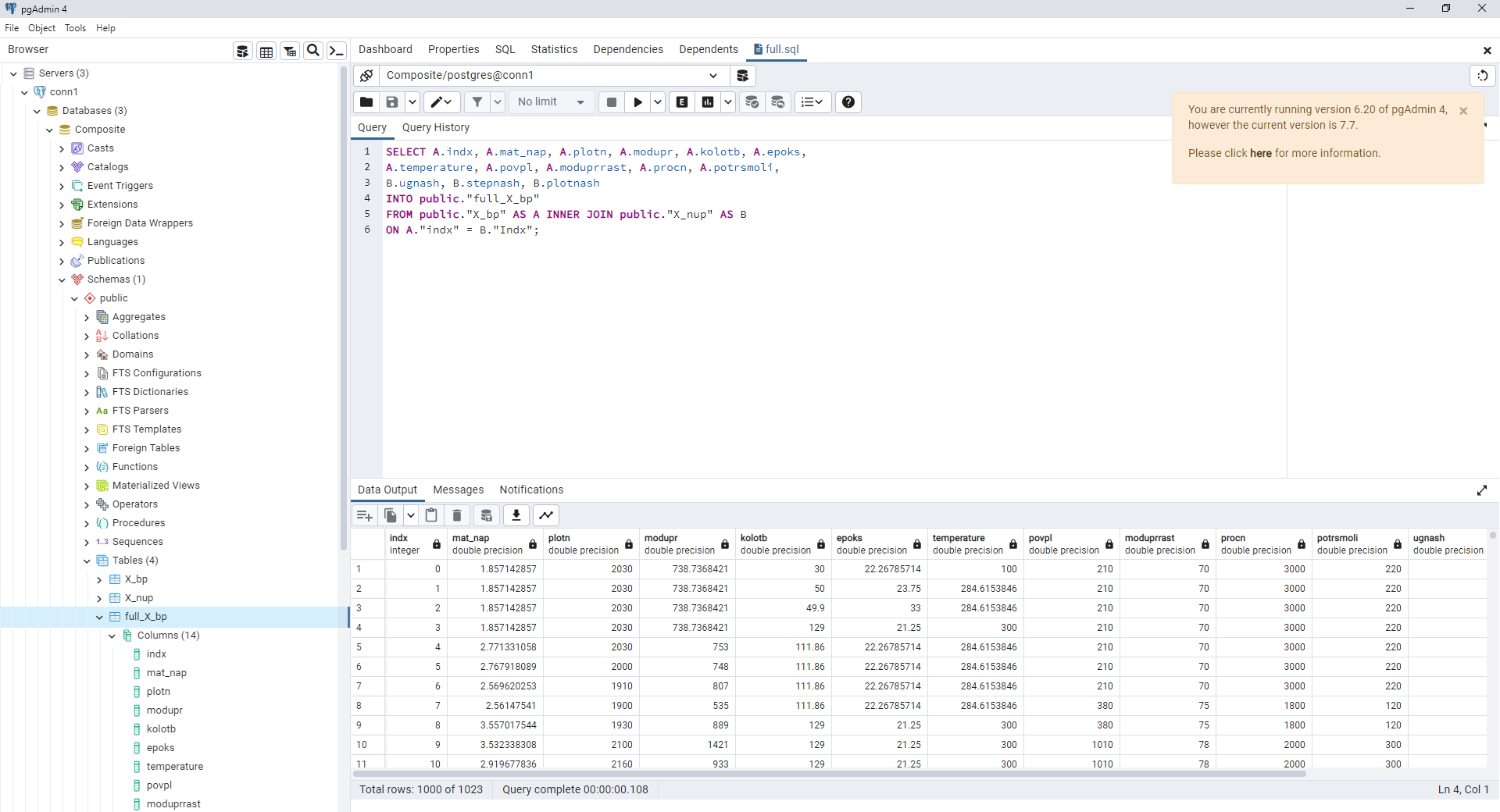


Рисунок 3 – Объединение двух датасет в PostgreSQL

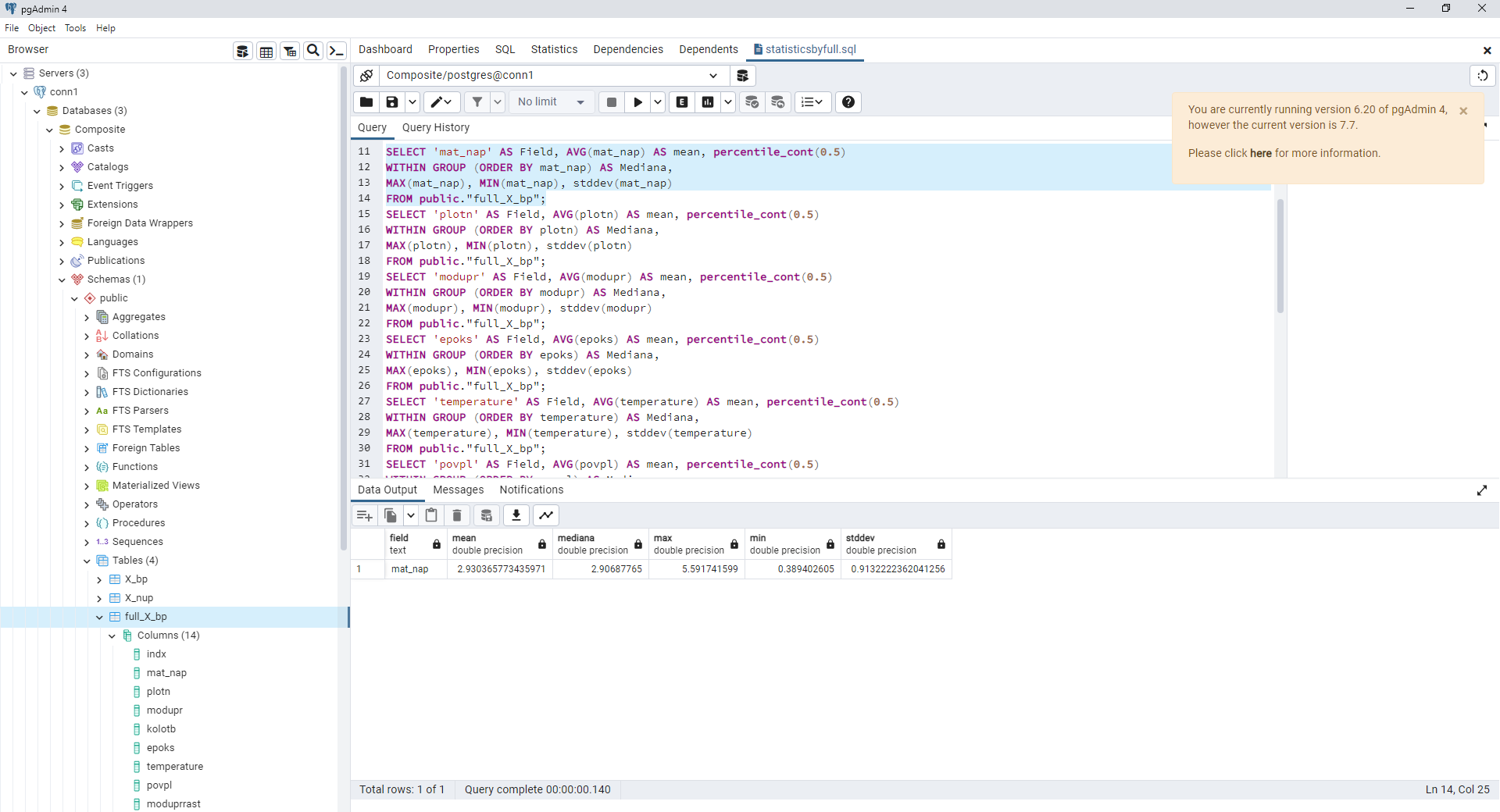


Рисунок 4 – Анализ свойств и статистических показателей в PostgreSQL

Средства Python pandas и seaborn позволяют загрузить данные непосредственно из Excel-файлов без промежуточных преобразований и получить статистические показатели по этим данным, а также объединить коллекции и получить статистические показатели по объединенной коллекции (скрипт 1\_PrimaryAnalizeInputData.py).

Полученные данные заносим в сводные таблицы. Описание признаков объединенного датасета представлены в Таблице 2. Статистические показатели представлены на рисунке 5.

Таблица 1 - Сравнение методов получения показателей описательной статистики средствами PostgreSQL и Python.

|  |  |  |  |
| --- | --- | --- | --- |
| Решаемая задача | Методы и конструкции  PostgreSQL | Методы и конструкции  Python (pandas и seaborn) | |
| 1 | 2 | 3 | |
| Создание  таблиц данных | CREATE TABLE X\_bp (поля, типы)  CREATE TABLE X\_nup (поля, типы) | Отдельно не требуется | |
| Преобразование исходного файла | Штатные средства Excel | Не требуется | |
| Загрузка данных | Штатные средства pgAdmin,  загрузка из csv-файлов | | Непосредственная загрузка данных из Excel  X\_bp\_df = pd.read\_excel(…)  X\_nup\_df = pd.read\_excel(…) | |
| Первичный анализ данных каждой коллекции (таблицы).  Количество строк, описание полей и типы данных | SELECT COUNT(\*) FROM X\_bp  SELECT COUNT(\*) FROM X\_nup  Описание полей определено при создании таблиц.  Получить поля с пустыми значениями получаем запросом вида  SELECT поля FROM таблица  WHERE поле1 IS NULL OR поле2 IS NULL OR ….; | | print(X\_bp\_df.shape)  print(X\_nup\_df.shape)  print(X\_bp\_df.info())  print(X\_nup\_df.info()) | |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Объединение таблиц методом INNER JOIN и получение новой коллекции | SELECT поля…  INTO "full\_X\_bp"  FROM "X\_bp" AS A INNER JOIN "X\_nup" AS B  ON A."indx" = B."Indx"; | X\_bp\_nup\_df = pd.concat([X\_bp\_df, X\_nup\_df], axis=1, join="inner") |
| Получение характеристик и статистических показателей новой (объединенной) коллекции (таблицы)  количество, cреднее, медиана, максимум, минимум, стандартное отклонение. | Используемые агрегатные и статистические функции в конструкции  SELECT … FROM T;   * COUNT(\*) * AVG(поле)   - percentile\_cont(0.5)  WITHIN GROUP (ORDER BY поле)   * MAX(поле) * MIN(поле)   - stddev(поле)  SELECT COUNT(\*) FROM (  SELECT поле FROM T GROUP BY поле) для расчета уникальных значений. | print(X\_bp\_nup\_df.shape)  print(X\_bp\_nup\_df.info())  Данные по каждому полю:  for i in X\_bp\_nup\_df.columns:  print(i) print(X\_bp\_nup\_df[i].describe()) |

Таблица 2 - Описание признаков объединенного датасета (full\_X\_bp)

| Название | Тип данных | Наличие пустых значений | Уникальных значений |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| Соотношение матрица-наполнитель | float64 | нет | 1014 |
| Плотность, кг/м3 | float64 | нет | 1013 |
| модуль упругости, ГПа | float64 | нет | 1020 |
| Количество отвердителя, м.% | float64 | нет | 1005 |
| Содержание эпоксидных групп,%\_2 | float64 | нет | 1004 |
| Температура вспышки, С\_2 | float64 | нет | 1003 |
| Поверхностная плотность, г/м2 | float64 | нет | 1004 |
| Модуль упругости при растяжении, ГПа | float64 | нет | 1004 |
| Прочность при растяжении, МПа | float64 | нет | 1004 |
| Потребление смолы, г/м2 | float64 | нет | 1003 |
| Угол нашивки, град | float64 | нет | 2 |
| Шаг нашивки | float64 | нет | 989 |
| Плотность нашивки | float64 | нет | 988 |

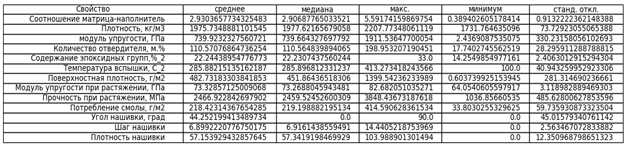


Рисунок 5 - Показатели описательной статистики объединенного датасета

С помощью библиотеки seaborn (sns.displot, sns.rugplot, sns.boxplot, sns.stripplot) произведем оценку плотности и распределения вероятности величин с графической визуализацией в виде гистограмм распределения и диаграмм «ящик с усами» по каждому из параметров (скрипт 2\_BuildDisplotAndBoxplot\_1.py). Полученные гистограммы и диаграммы занесем в таблицу графиков (Таблица 3) и проанализируем.

Таблица 3 - Гистограммы распределения переменных и диаграммы «ящик с усами» по каждому из свойств дататасета.

|  |  |  |
| --- | --- | --- |
| Свойство | Гистограмма распределения | Диаграмма  «ящик с усами» |
| 1 | 2 | 3 |
| Соотношение матрица-наполнитель |  |  |
| Плотность, кг/м3 |  |  |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Модуль упругости, ГПа |  |  |
| Количество отвердителя, м.% |  |  |
| Содержание эпоксидных групп,%\_2 |  |  |
| Температура вспышки, С\_2 |  |  |
| Поверхностная плотность, г/м2 |  |  |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Модуль упругости при растяжении, ГПа |  |  |
| Прочность при растяжении, МПа |  |  |
| Потребление смолы, г/м2 |  |  |
| Угол нашивки, град |  |  |
| Шаг нашивки |  |  |

Продолжение таблицы 3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Плотность нашивки |  |  |

Из гистограмм распределения переменных и диаграмм «ящик с усами» видно, что все признаки, кроме «Угол нашивки», имеют нормальное распределение (график в виде «колокола») и принимают неотрицательные значения.

Все признаки, кроме «Угол нашивки», являются непрерывными, количественными. «Угол нашивки» принимает только два значения (0, 90) и будет рассматриваться как категориальный признак. В дальнейшем при нормализации значение 0 градусов будет обозначено как 0, а 90 градусов как 1. Ранее по результатам анализа данных было выявлено отсутствие пропусков (пустых данных).

С помощью функции seaborn.PairGrid построим попарные графики рассеяния точек представленный на рисунке 6 (скрипт 3\_BuildPearGrid\_1.py).

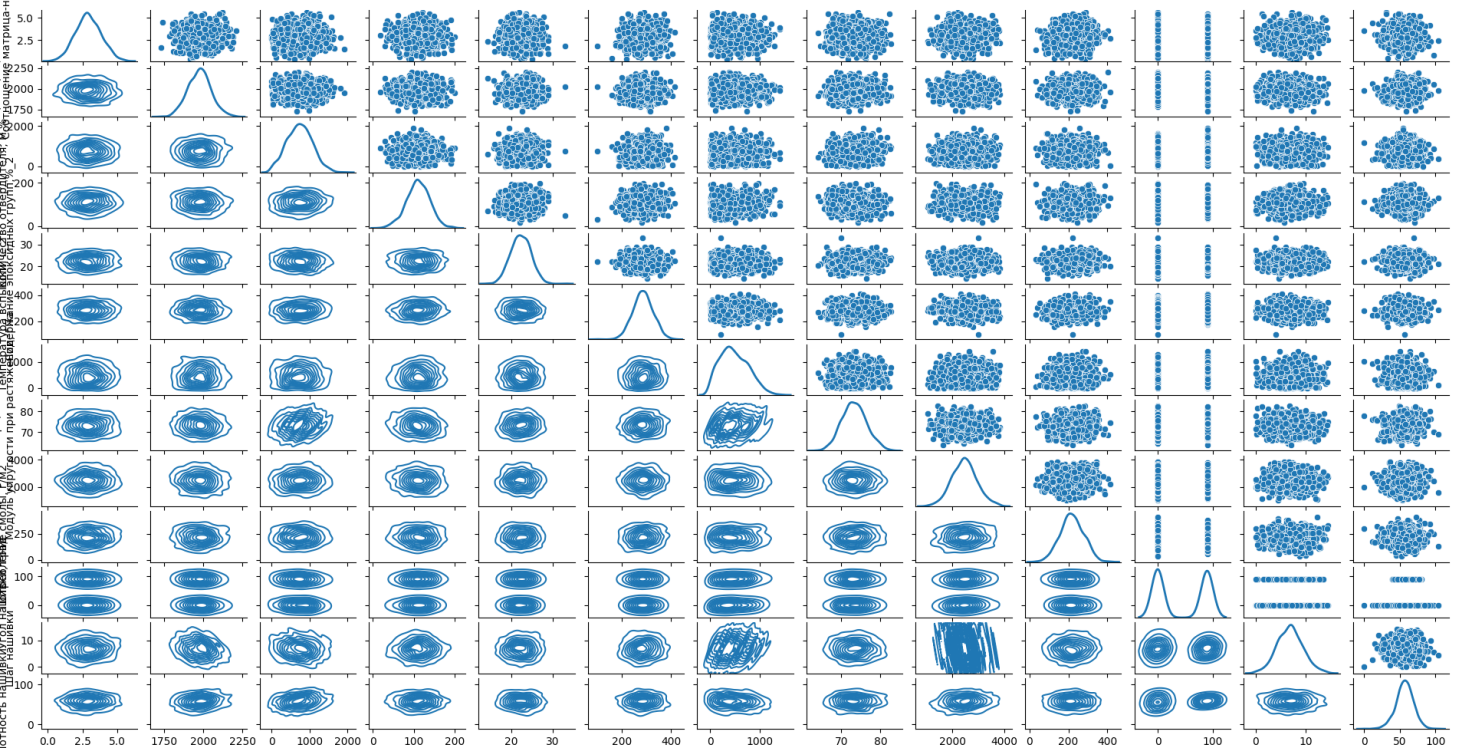


Рисунок 6 — Попарные графики рассеяния

По графикам рассеяния мы видим, что некоторые точки отстоят далеко от общего облака. Так визуально выглядят выбросы — аномальные, некорректные значения данных, выходящие за пределы допустимых значений признака.

Есть следующие методы выявления выбросов для признаков с нормальным распределением:

1. Метод стандартного отклонения (3-х сигм), где верхняя (Limmax) и нижняя (Limmin​) граница выбросов определяются по формуле:

*Limmax* ​= *mean* + *Ns*​∗*S*

*Limmin* ​= *mean* - *Ns*​∗*S ,*

где mean - среднее значение;

S - [стандартное отклонение](https://wiki.loginom.ru/articles/mean-square-deviation.html);

Ns = 3 - заданное число стандартных отклонений (3-х сигм);

1. Метод межквартильных расстояний, где верхняя (Limmax) и нижняя (Limmin​) граница выбросов определяются по формуле:

*Limmax*​=*Q*3​+*Ni*​∗*IQR*

*Limmin*​=*Q*1​−*Ni*​∗*IQR ,*

где Q3 - третий квартиль (значение 75%);

Q1 - первый квартиль (значение 25%);

IQR — интерквартильное расстояние (или [интерквартильный размах](https://wiki.loginom.ru/articles/iqr.html" \t "_blank)), определяемое по формуле IQR= *Q*3− *Q*1;

Ni​ = 1.5 — заданное число интерквартильного размаха.

Как в графическом виде выбросы выглядят, например, для свойства "Плотность, кг/м3" можно получить средствами Python seaborn построив гистограмму распределения и диаграмму «ящик с усами» представленною на рисунке 7:

sns.displot(x="Плотность, кг/м3",kde=True,data=X\_bp\_nup\_df)

sns.boxplot(x=" Плотность нашивки ", data=X\_bp\_nup\_df,

whis=[0, 100], width=.6, palette="vlag") (Рисунок )

|  |  |
| --- | --- |
| plotnas.png | plotnas2.png |

Рисунок 7 — Пример выбросов

Применяем указанные методы к объединенной коллекции средствами PostgreSQL и Python pandas (скипт 4\_OtsevBy3SigmAndIntervals.py) и seaborn отсеиваем данные, содержащие выбросы.

В сравнительную таблицу (Таблица 4) заносим особенности методов и языковых конструкций для реализации этой задачи, и возвращаемые системой значения при выполнении соответствующих скриптов.

Таблица 4 – Сравнительная таблица

|  |  |  |
| --- | --- | --- |
| Решаемая задача | Методы и конструкции  PostgreSQL | Методы и конструкции  Python pandas и seaborn |
| 1 | 2 | 3 |
| Отсев строк, содержащие выбросы методом  3-х сигм. | SELECT поля… FROM public."full\_X\_bp" WHERE mat\_nap > ((SELECT AVG(mat\_nap) FROM public."full\_X\_bp") - 3.0 \* (SELECT stddev(mat\_nap) FROM public."full\_X\_bp" )) AND mat\_nap < ((SELECT AVG(mat\_nap) FROM public."full\_X\_bp") + 3.0 \* (SELECT stddev(mat\_nap) FROM public."full\_X\_bp" )) | filtered\_df = X\_bp\_nup\_df.copy ()  print(filtered\_df)  for i in filtered\_df.columns:  if i != 'Угол нашивки, град':  filtered\_df = filtered\_df[(filtered\_df[i] > filtered\_df[i].mean() - 3 \*filtered\_df[i].std()) & (filtered\_df[i] < filtered\_df[i].mean() + 3 \*filtered\_df[i].std())]  print(filtered\_df) |

Продолжение таблицы 4

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Результат отсева строк, содержащие выбросы методом  3-х сигм. | Из 1023 строк остается 1002 строки  удален 21 выброс  (расхождение может быть связано с особенностями вычислений) | Из 1023 строк остается 1000 строк. Удалено 23 выброса. |
| Отсев строк, содержащие выбросы методом межквартильных расстояний | SELECT mat\_nap FROM public."full\_X\_bp" WHERE mat\_nap > (SELECT percentile\_cont(0.25) WITHIN GROUP (ORDER BY mat\_nap) FROM public."full\_X\_bp") - 1.5 \* ((SELECT percentile\_cont(0.75)  WITHIN GROUP (ORDER BY mat\_nap) FROM public."full\_X\_bp") - (SELECT percentile\_cont(0.25)  WITHIN GROUP (ORDER BY mat\_nap) FROM public."full\_X\_bp")) | filtered\_df = X\_bp\_nup\_df.copy ()  print(filtered\_df)  for i in filtered\_df.columns:  if i != 'Угол нашивки, град':  q3v = q3(filtered\_df[i])  q1v = q1(filtered\_df[i])  NIQR = (q3v - q1v)\*1.5  Limmax = q3v + NIQR  Limmin = q1v - NIQR  filtered\_df = filtered\_df[(filtered\_df[i] > Limmin) & (filtered\_df[i] < Limmax)]  print(filtered\_df) |
| Результат отсева строк, содержащие выбросы методом межкв. рас. | из [1023 rows x 13 columns] остается [936 rows x 13 columns] удалено 87 выбросов. | из [1023 rows x 13 columns] остается [932 rows x 13 columns] удалено 91 выброс. |

Поскольку известно, что датасет получен из реальных производственных задачах Центра НТИ «Цифровое материаловедение: новые материалы и вещества» (структурное подразделение МГТУ им. Н.Э. Баумана), то он очищен явного шума (т.е. свойств материалов не относящихся к реальным композитам). Для очистки от выбросов выберем метод 3-x сигм.

Таким образом, очищенный датасет, который будет использован для обучения и тестирования модели будет содержать 1000 строк и 13 признаков-переменных. Такой очищенный датасет (clearing\_X\_bp\_nup.xlsx) мы получим скиптом 5\_OtsevBy3SigmAndBuildsGrafs.py. Аналогично тому, как мы производили (с помощью библиотеки seaborn) оценку плотности и распределения вероятности величин с графической визуализацией в виде гистограмм распределения и диаграмм «ящик с усами» по каждому из параметров, а также построением попарных графиков рассеяния на объединенной коллекции, выполним тоже самое на очищенной от выбросов коллекции (Таблица 5).

Таблица 5 - Гистограммы распределения переменных и диаграммы «ящик с усами» по каждому из свойств очищенного дататасета.

|  |  |  |
| --- | --- | --- |
| Свойство | Гистограмма распределения | Диаграмма «ящик с усами» |
| 1 | 2 | 3 |
| Соотношение матрица-наполнитель | mat_nap.png | mat_nap2.png |
| Плотность, кг/м3 | plotn.png | plot_2.png |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Модуль упругости, ГПа | mod_upr.png | mod_upr2.png |
| Количество отвердителя, м.% | kol_otv.png | kol_otv2.png |
| Содержание эпоксидных групп,%\_2 | sod_sm.png | sod_sm2.png |
| Температура вспышки, С\_2 | Temperature.png | Temp2.png |
| Поверхностная плотность, г/м2 | pov_pl.png | pov_pl2.png |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Модуль упругости при растяжении, ГПа | mod_upr_rast.png | mod_upr_rast2.png |
| Прочность при растяжении, МПа | procn_rast.png | procn_rast2.png |
| Потребление смолы, г/м2 | potr_sm.png | potr_sm2.png |
| Угол нашивки, град | ugol_nash.png | ugol_nas2.png |
| Шаг нашивки | Step_nas.png | step_nas2.png |

Продолжение таблицы 5

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Плотность нашивки | plot_nas.png | plot_nas2.png |

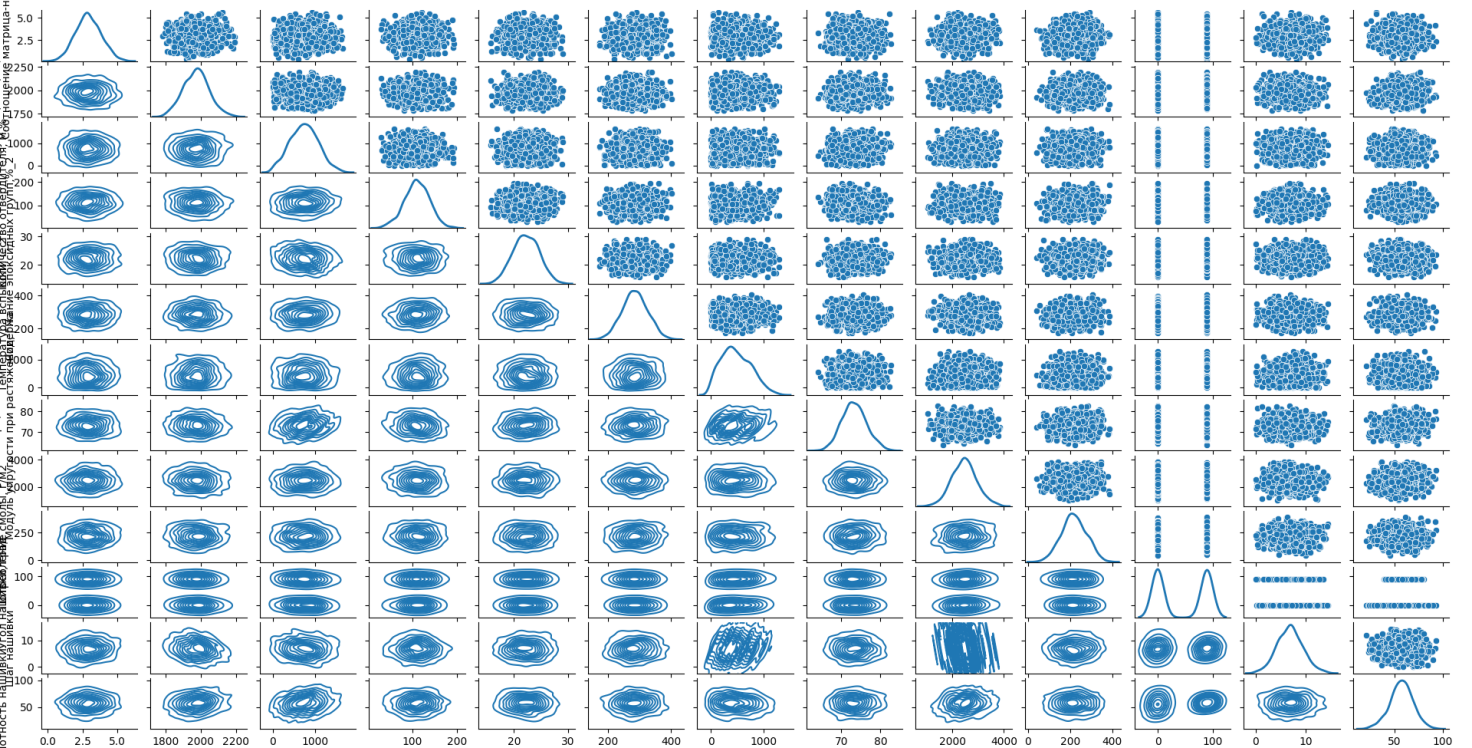


Рисунок 8 - Попарные графики рассеяния

В объединенном и очищенном от выбросов датасете clearing\_X\_bp\_nup.xlsx из 13 колонок, в соответствии с заданием, целевыми переменными, которые необходимо предсказывать, определены:

* модуль упругости при растяжении, Гпа;
* прочность при растяжении, МПа;
* соотношение матрица-наполнитель.
  1. 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, называемые также предикторами или регрессорами. Регрессионный анализ помогает понять, как «типичное» значение зависимой переменной изменяется при изменении независимых переменных.

# В настоящее время разработано много математических методов регрессионного анализа и многие из них реализованы в программные алгоритмы, например, для Python такие алгоритмы собраны в библиотеку scikit-learn, и чтобы активировать оценщик требуемого метода достаточно вызвать соответствующую функцию, даже без параметров и начать пользоваться (обучать и предсказывать). В этом случае гиперпараметры примут значения по умолчанию. Узнать какие гиперпараметры можно настроить, чтобы оптимизировать работу оценщика поможет вызов функции get\_params(). Перечень математических методов регрессивного анализа, соответствующие им оценщики scikit-learn и гиперпараметры по умалчанию сведем в таблицу (Таблица 6).

Таблица 6 – Математические методы регрессивного анализа

|  |  |  |
| --- | --- | --- |
| Математический метод регрессивного анализа | Функция библиотеки scikit-learn | Гиперпараметры  со значениями по умалчанию |
| 1 | 2 | 3 |
| Линейная регрессия | LinearRegression | {'copy\_X': True, 'fit\_intercept': True, 'n\_jobs': None, 'positive': False} |

Продолжение таблицы 6

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Линейная регрессия лассо | Lasso | {'alpha': 1.0, 'copy\_X': True, 'fit\_intercept': True, 'max\_iter': 1000, 'positive': False, 'precompute': False, 'random\_state': None, 'selection': 'cyclic', 'tol': 0.0001, 'warm\_start': False} |
| Гребневая регрессия | Ridge | {'alpha': 1.0, 'copy\_X': True, 'fit\_intercept': True, 'max\_iter': None, 'positive': False, 'random\_state': None, 'solver': 'auto', 'tol': 0.0001} |
| Метод опорных векторов для регрессии | svm.SVR | {'C': 1.0, 'cache\_size': 200, 'coef0': 0.0, 'degree': 3, 'epsilon': 0.1, 'gamma': 'scale', 'kernel': 'rbf', 'max\_iter': -1, 'shrinking': True, 'tol': 0.001, 'verbose': False} |
| Метод k-ближайших соседей | neighbors.KneighborsRegressor | {'algorithm': 'auto', 'leaf\_size': 30, 'metric': 'minkowski', 'metric\_params': None, 'n\_jobs': None, 'n\_neighbors': 5, 'p': 2, 'weights': 'uniform'} |
| Деревья решений | tree.DecisionTreeRegressor | {'ccp\_alpha': 0.0, 'criterion': 'squared\_error', 'max\_depth': None, 'max\_features': None, 'max\_leaf\_nodes': None, 'min\_impurity\_decrease': 0.0, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'min\_weight\_fraction\_leaf': 0.0, 'random\_state': None, 'splitter': 'best'} |

Продолжение таблицы 6

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Случайный лес | ensemble.RandomForestRegressor | {'bootstrap': True, 'ccp\_alpha': 0.0, 'criterion': 'squared\_error', 'max\_depth': 2, 'max\_features': 1.0, 'max\_leaf\_nodes': None, 'max\_samples': None, 'min\_impurity\_decrease': 0.0, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'min\_weight\_fraction\_leaf': 0.0, 'n\_estimators': 100, 'n\_jobs': None, 'oob\_score': False, 'random\_state': 0, 'verbose': 0, 'warm\_start': False} |
| Градиентный бустинг | ensemble.GradientBoostingRegressor | {'alpha': 0.9, 'ccp\_alpha': 0.0, 'criterion': 'friedman\_mse', 'init': None, 'learning\_rate': 0.1, 'loss': 'squared\_error', 'max\_depth': 3, 'max\_features': None, 'max\_leaf\_nodes': None, 'min\_impurity\_decrease': 0.0, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'min\_weight\_fraction\_leaf': 0.0, 'n\_estimators': 100, 'n\_iter\_no\_change': None, 'random\_state': 0, 'subsample': 1.0, 'tol': 0.0001, 'validation\_fraction': 0.1, 'verbose': 0, 'warm\_start': False} |

* 1. 1.3. 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.

Ранее при анализе данных были выявлены выбросы и скриптом (5\_OtsevBy3SigmAndBuildsGrafs.py ) данные очищены от выбросов. Очищенные данные записаны в выходной файл clearing\_X\_bp\_nup.xlsx, и для последующей работы будем использовать именно его.

Помочь выявить связь между признаками может тепловая карта корреляций представленная на рисунке 9, рассчитанная функцией pandas.DataFrame.corr() и построенная функцией seaborn.heatmap(). Необходимо обратить внимание на признаки с высокой корреляцией (близких к 1).

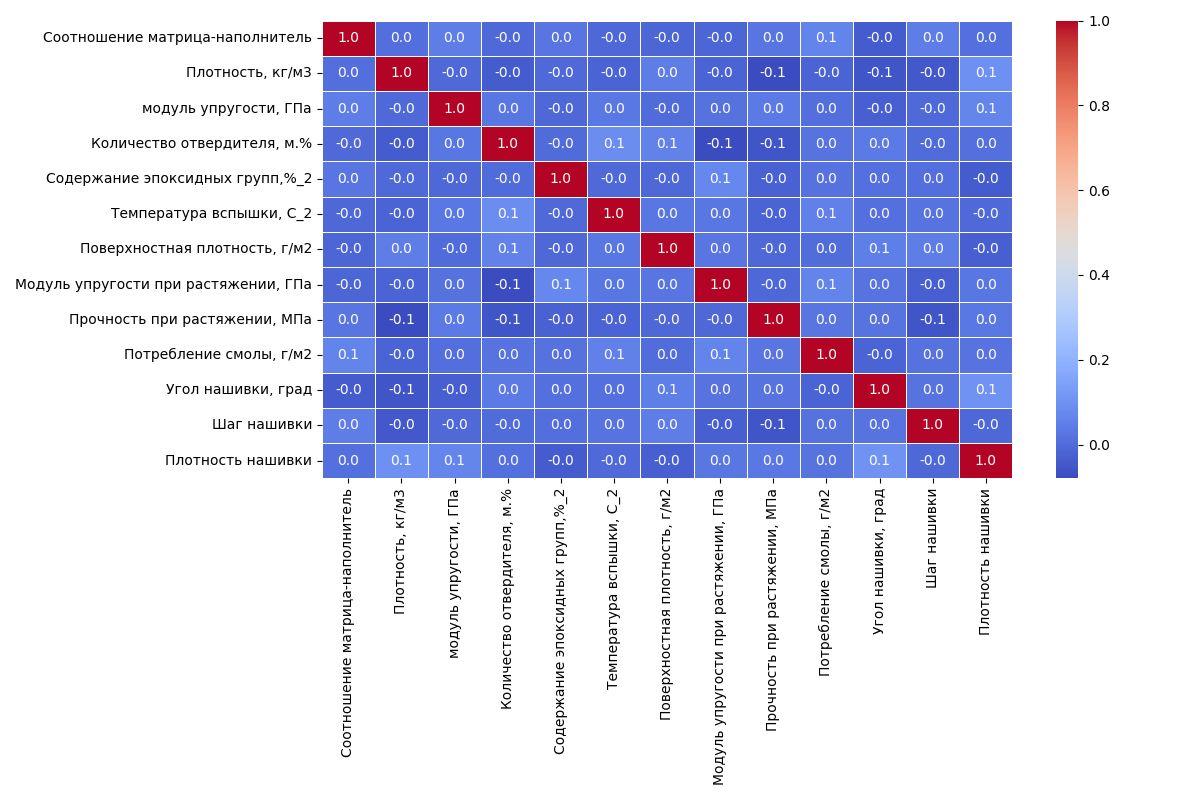


Рисунок 9 – Тепловая карта корреляции

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

* 1. 1.3.1. Выбор признаков

В соответствии с условиями задания выходными признаками являются:

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

Для каждого из целевых признаков будут построены отдельные модели, при этом «модуль упругости при растяжении» и «прочность при растяжении» будет определятся по 11 оставшимся признакам, а «соотношение матрица-наполнитель» по 12.

* 1. 1.3.2. Ход решения задачи

Ход решения каждой из задач и построения оптимальной модели будет следующим:

* проведу нормализацию данных и запишу 3 датасета для каждой из задач, при этом выходной параметр поставлю в последнюю колонку.
* проведу поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой количество блоков равно 10 для всех моделей и методов.
* разделю данные на тренировочную и тестовую выборки в объемах, рекомендованных в задании (70% для обучения, 30% для тестирования).
* проведу тестирование модели всеми рассмотренными ранее методами, с выводом оценочных показатели качества работы модели в виде графиков и таблиц, и остановлюсь на модели, которая будет использована при написании приложения.
* сохраню модели для каждого целевого параметра в виде файла.
* разработаю консольное приложений, которое будет запрашивать у пользователя входные параметры и подгрузив ранее сохраненную модель сделает предсказание целевого параметра.
  1. 1.3.3. Препроцессинг

Цель препроцессинга, или предварительной обработки данных — обеспечить корректную работу моделей.

Для препроцессинга имеются встроенные средства [sklearn.preprocessing](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing).StandardScaler и [sklearn.preprocessing](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing).MinMaxScaler, которые позволяют провести стандартизацию и нормализацию тренеровочных и тестовых данных (fit\_transform() и transform()) и восстановление в исходную коллекцию методом inverse\_transform(), но мы будем использовать «ручной» метод нормализации по формуле (x – min()) / (max() – min()) с сохранением значений дельты и множителя для каждого поля.

Скриптом (6\_normalizedata.py) выполним нормализацию данных. Категориальный признак 'Угол нашивки, град' в результате нормализации получит значение 0 для 0 град и 1 для значения 90 град. В процессе нормализации сохраняю значение множителя и дельты для каждого свойства (колонки данных) в виде словаря, сериализованного в JSON файл, который будет использован для восстановления истинных значений при работе приложения.

Также в ходе нормализации создаю файл normalized\_data.xlsx с нормализованными параметрами.

Далее скриптом (7\_testmodele.py) формирую три датасета moduprrast.csv, modprochrast.csv, mat\_nap.csv, в которых выходные поля (целевые параметры) стоят последними. Это позволить выделять входные параметры конструкцией X = values[:,:-1] # все кроме последнего и выходные параметры конструкцией Y = values[:,-1] # только последний.

* 1. 1.3.4. Метрики качества моделей и методов

В качестве оценочных метрик качества использую метрики sklearn.metrics:

* RMSE (Root Mean Squared Error) - корень из средней квадратичной ошибки) принимает значениях в тех же единицах, что и целевая переменная. Метрика использует возведение в квадрат, поэтому хорошо обнаруживает грубые ошибки, но сильно чувствительна к выбросам;
* MAE (Mean Absolute Error) - средняя абсолютная ошибка так же принимает значениях в тех же единицах, что и целевая переменная;
* MAPE (Mean Absolute Percentage Error) или средняя абсолютная процентная ошибка — безразмерный показатель, представляющий собой взвешенную версию MAE;
* max error или максимальная ошибка данной модели в единицах измерения целевой переменной.

1. 2. Практическая часть
   1. 2.1. Разбиение и предобработка данных

Ранее подготовленные три датасета moduprrast.csv, modprochrast.csv, mat\_nap.csv для прогнозирования модуля упругости при растяжении, прочности при растяжении и соотношения матрица-наполнитель уже имеют нормализованные данные. Также имеются сохраненные значения множителей и дельты для восстановления истинных значений из нормализованных по каждому из полей.

Разбиение на тренировочные и тестовые данные будем осуществлять непосредственно перед их применением методом sklearn.model\_selection. train\_test\_split в объемах, установленных в задании, т.е. 70 и 30 процентов соответственно.

* 1. 2.2. Разработка и обучение модели. Выбор лучшей модели.

Выбор лучшей модели для всех трех прогнозов выполняется аналогичным способом. Сначала в цикле перебираем все модели (для задач регрессии) и функцией get\_params() получаем список всех гиперпараметров (скрипт 9\_getAllgiperparams.py) для них. Далее с помощью GridSearchCV провожу оценку параметров с использование поиска по сетке с перекрестной проверкой с количеством блоков 10. Перекрестная проверка уже встроена в GridSearchCV, а количество блоков указывается в параметрах функции GridSearchCV - cv = 10, а значения гиперпараметров, которые необходимо перебирать, указывается в массиве. Такую операцию провожу для каждой модели и каждого целевого параметра. Результаты представлены в таблицах 7 – 12.

Таблица 7 - Метрики с лучшими параметрами линейной регрессии лассо (Lasso).

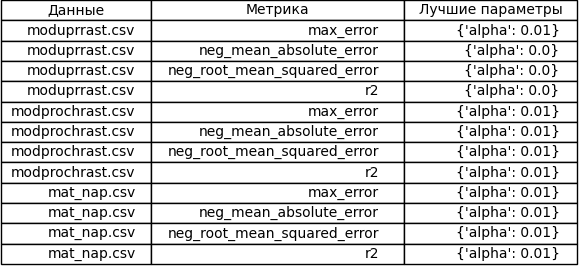


Таблица 8 - Метрики с лучшими параметрами гребневой регрессии (Ridge).

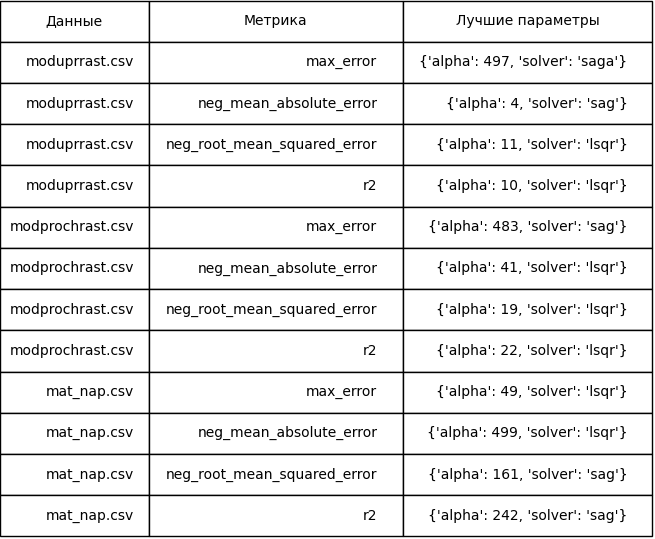


Таблица 9 - Метрики с лучшими параметрами метода опорных векторов для регрессии (SVR).

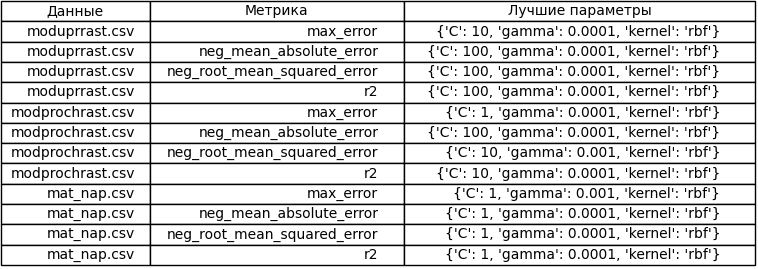


Таблица 10 - Метрики с лучшими параметрами метода k-ближайших соседей (KNeighborsRegressor)

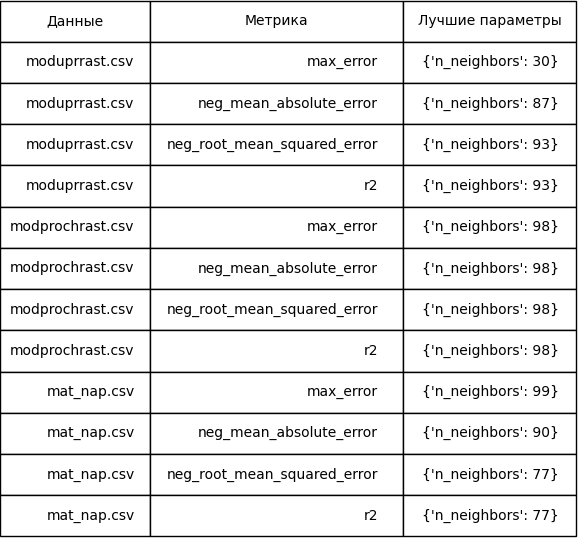


Таблица 11 - Метрики с лучшими параметрами деревья решений

(DecisionTreeRegressor).

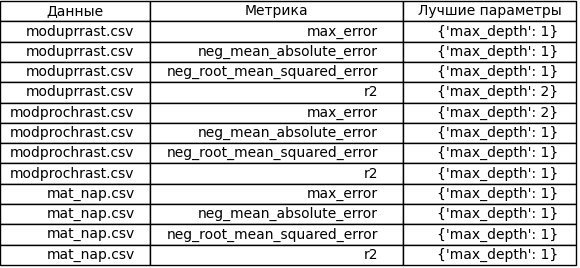
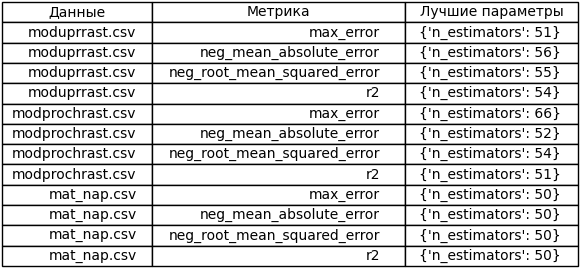


Таблица 12 - Метрики с лучшими параметрами градиентный бустинг

(GradientBoostingRegressor)



Дождаться завершения 1\_podbor\_parameters\_RandomForestRegressor.py не удалось, поэтому для RandomForestRegressor установили гиперпараметры max\_depth=2 и random\_state=0, как указано в примере на сайте https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html.

Скриптами (9\_examine\_mod\_upr.py) в цикле запускаю обучение и тестирование моделей с параметрами по умолчанию, т.е. модель из «коробки», и оптимально настроенными параметрами. Накапливаю оценочные показатели для тестовой и тренировочной выборки, для оптимальных и «дефолтовых» настроек и на основании, которых выстраиваю сравнительные графики (представленные на рисунках 10 - 15) и таблицу оценочных метрик (таблицы 13 - 15). В перечень моделей добавлен DummyRegressor, который делает прогнозы, используя простые правила. Этот регрессор полезен в качестве простой базовой линии для сравнения с другими (реальными) регрессорами.

Разбиение на учебный набор данных (X\_train, Y\_train) и тестовый (X\_test, Y\_test) будет проводиться непосредственно перед применением, поэтому данный скрипт выведет описательную статистику для train и test данных представленых в таблицах 16 - 18.

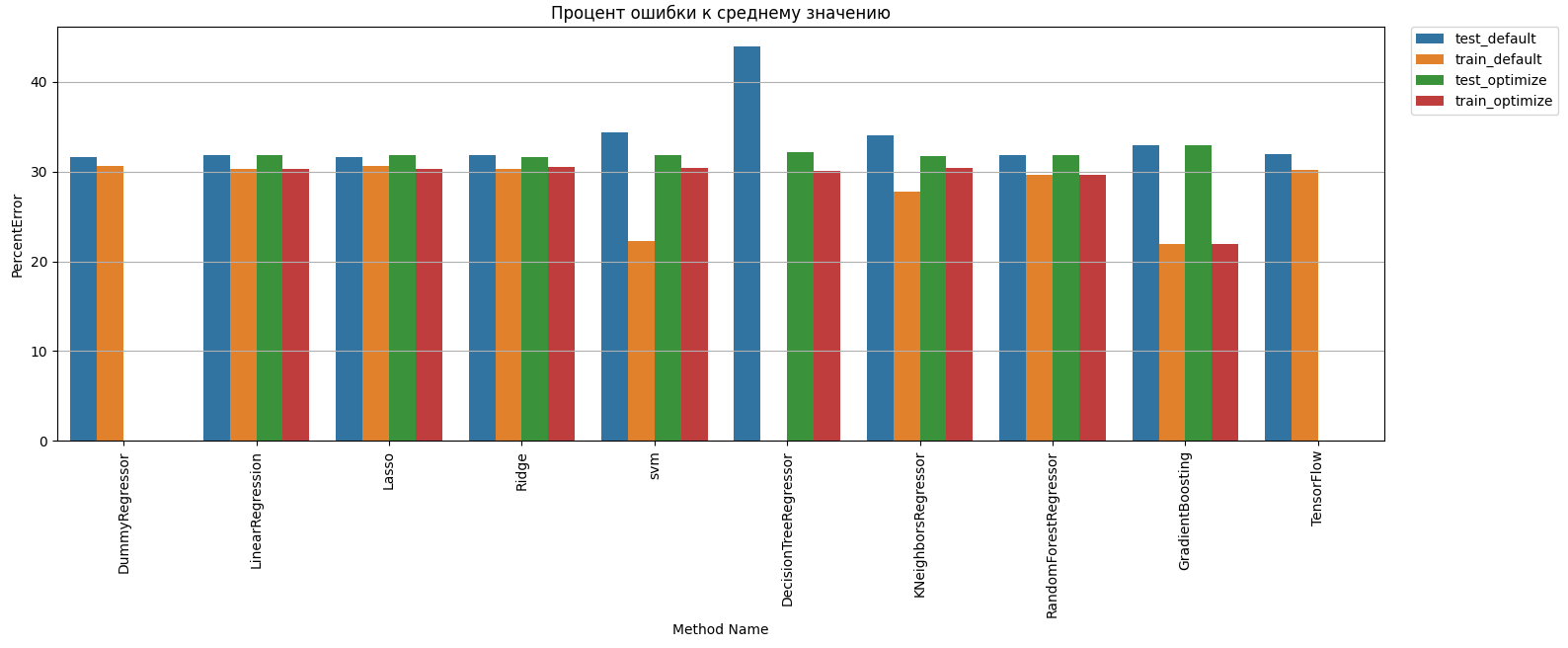


Рисунок 10 – Процент ошибок

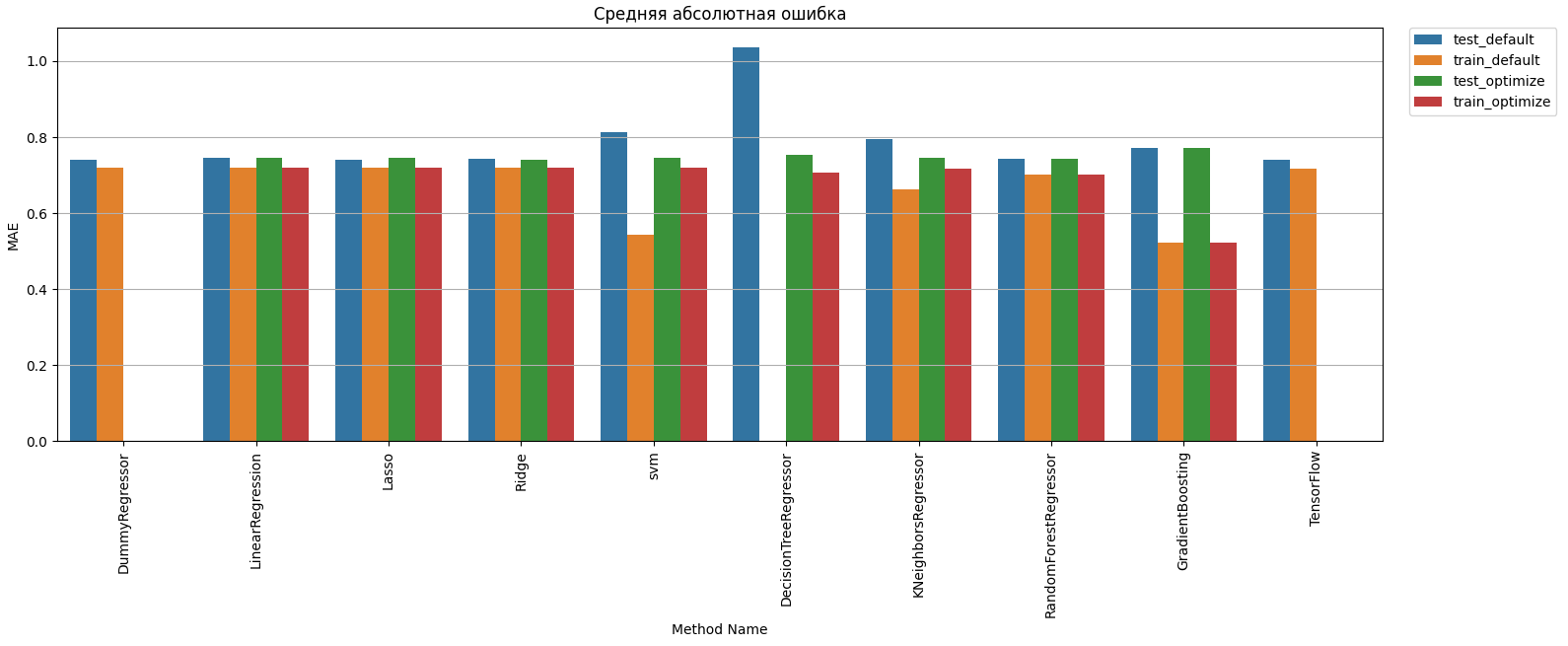


Рисунок 11 – Средняя абсолютная ошибка

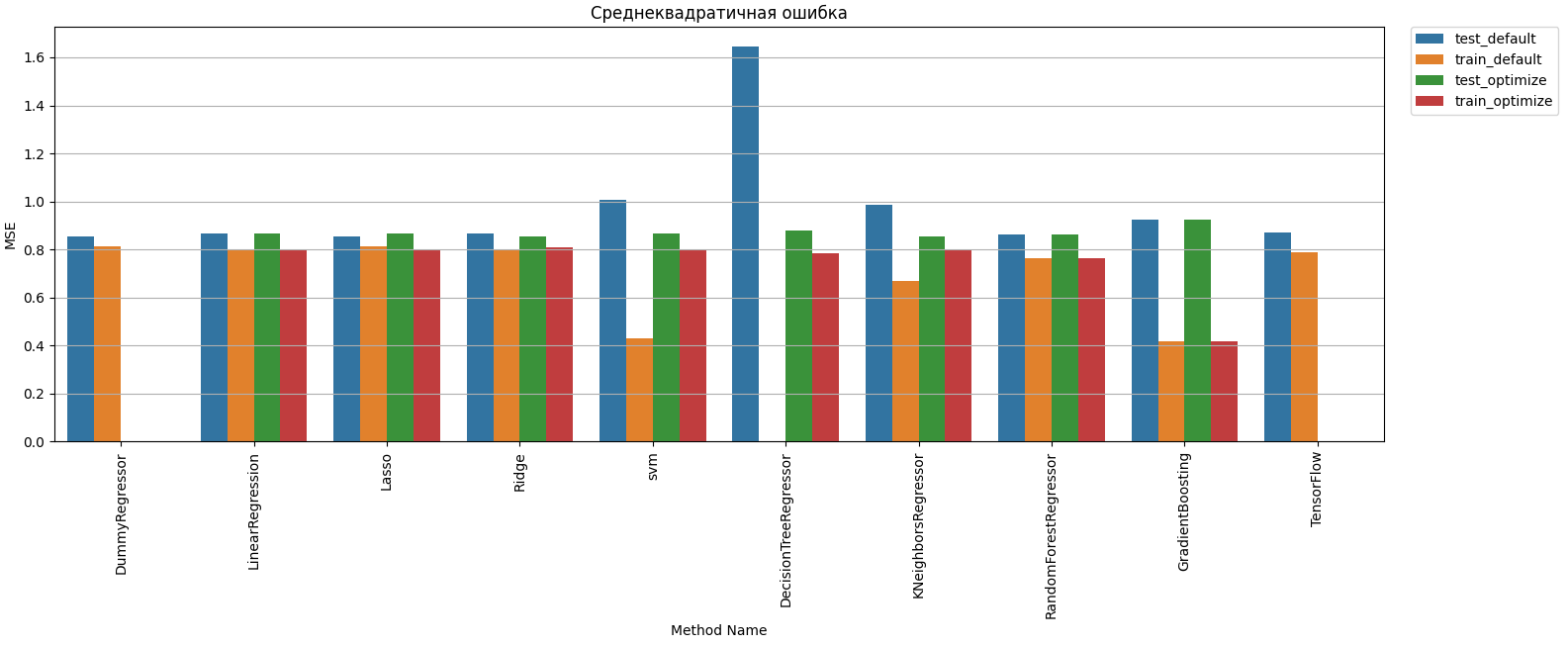


Рисунок 12 – Среднеквадратичная ошибка

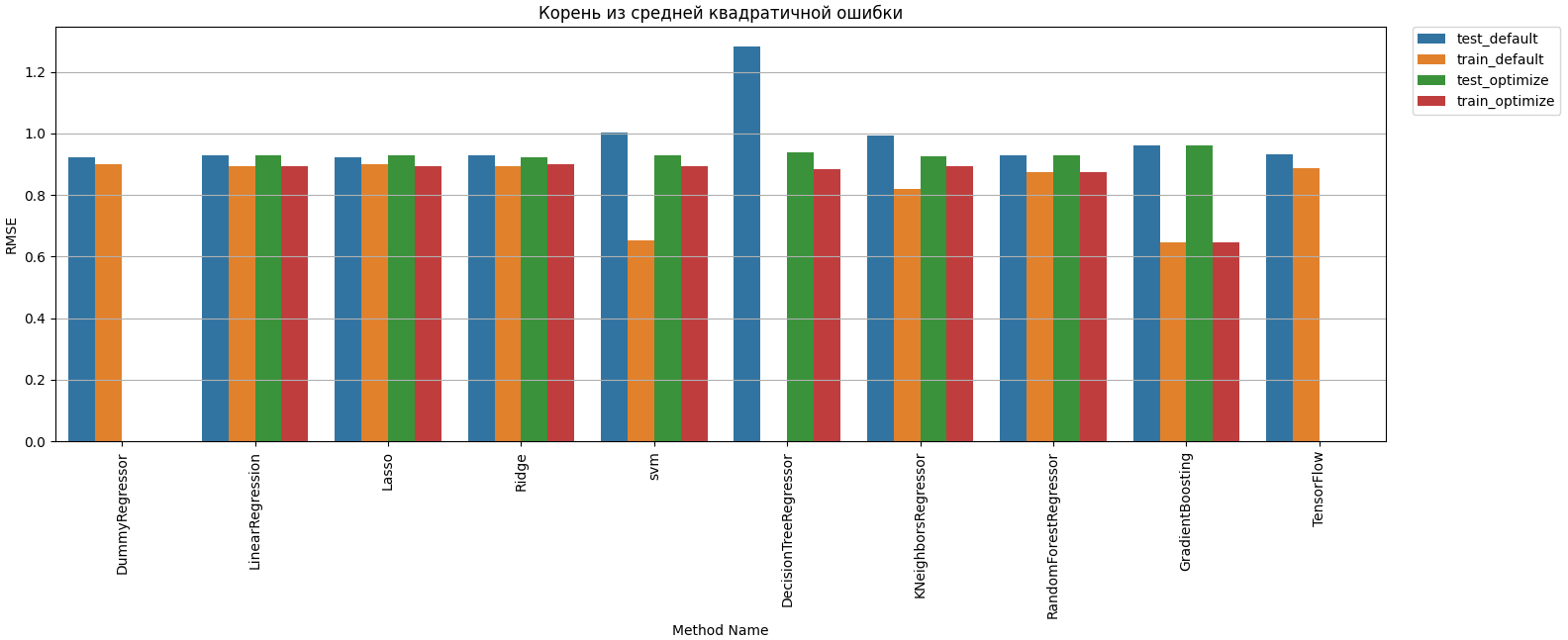


Рисунок 13 – Корень из средней квадратичной ошибки

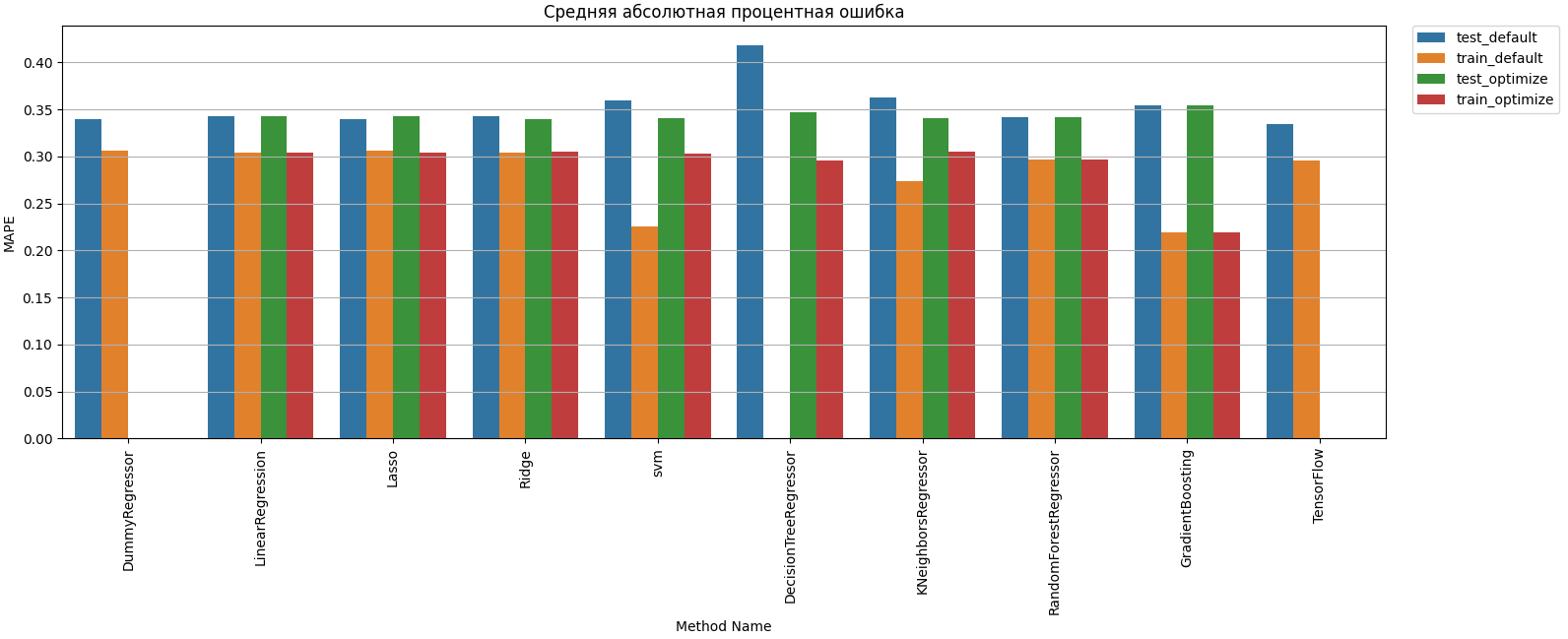


Рисунок 14 – Средняя абсолютная процентная ошибка

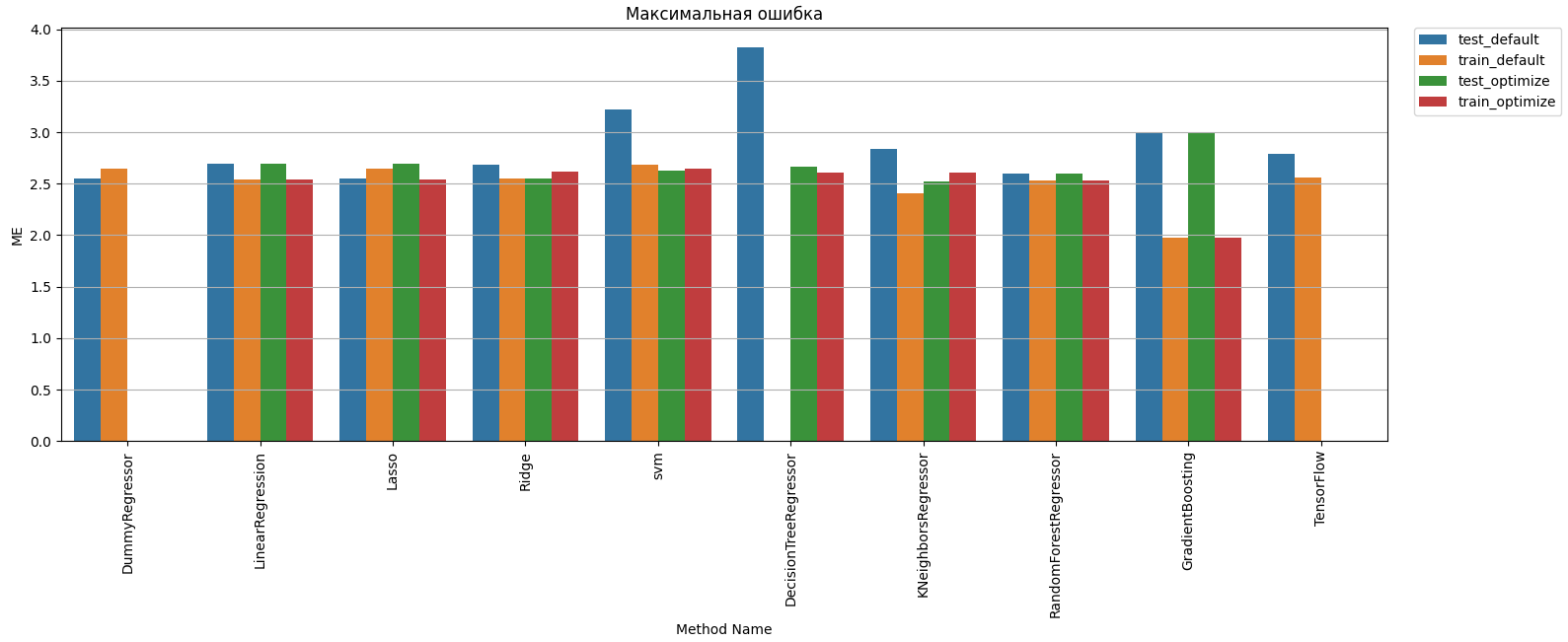


Рисунок 15 – Максимальная ошибка

Таблица 13 – Оценочные характеристики обработки данных материал – наполнитель различными методами

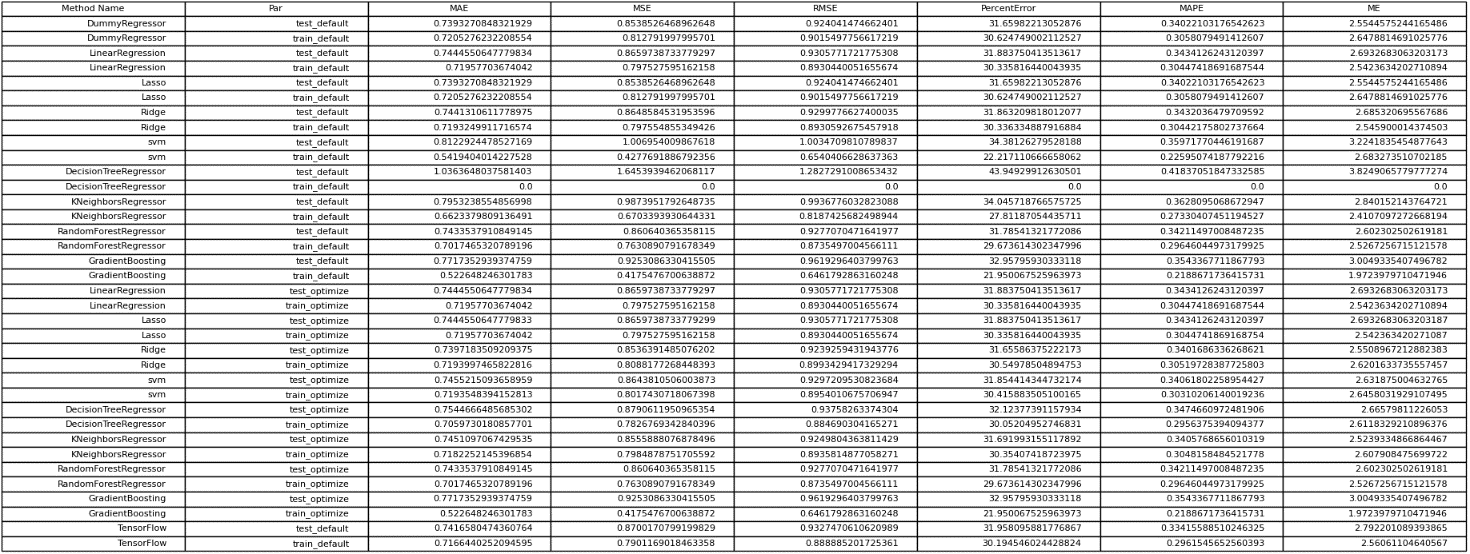


Таблица 14 – Оценочные характеристики обработки данных модуль упругости различными методами

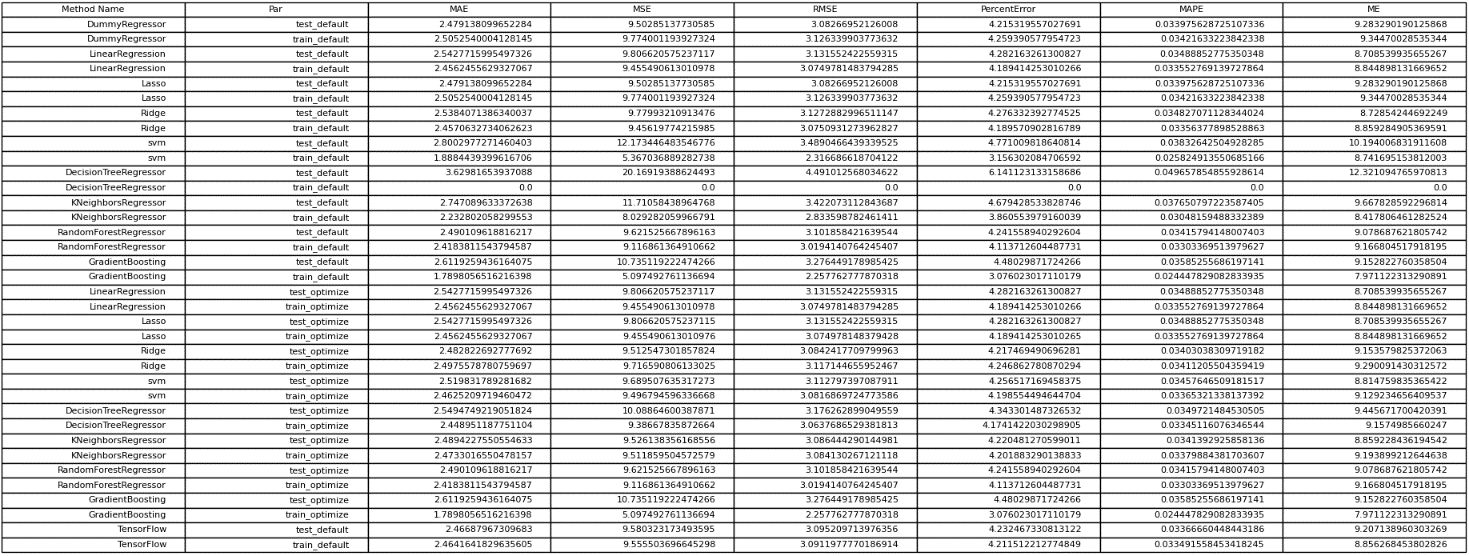


Таблица 15 – Оценочные характеристики обработки данных модуль прочности различными методами

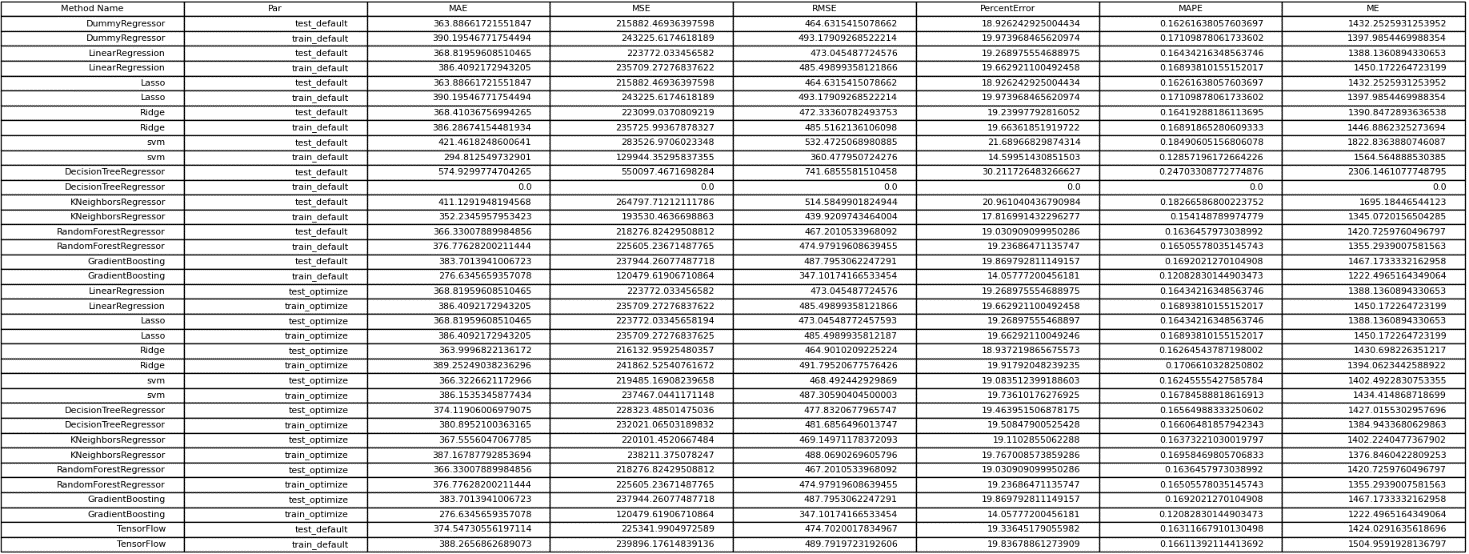


Таблица 16 – Описательная статистика для обучающего и тестового набора данных материал – наполнитель

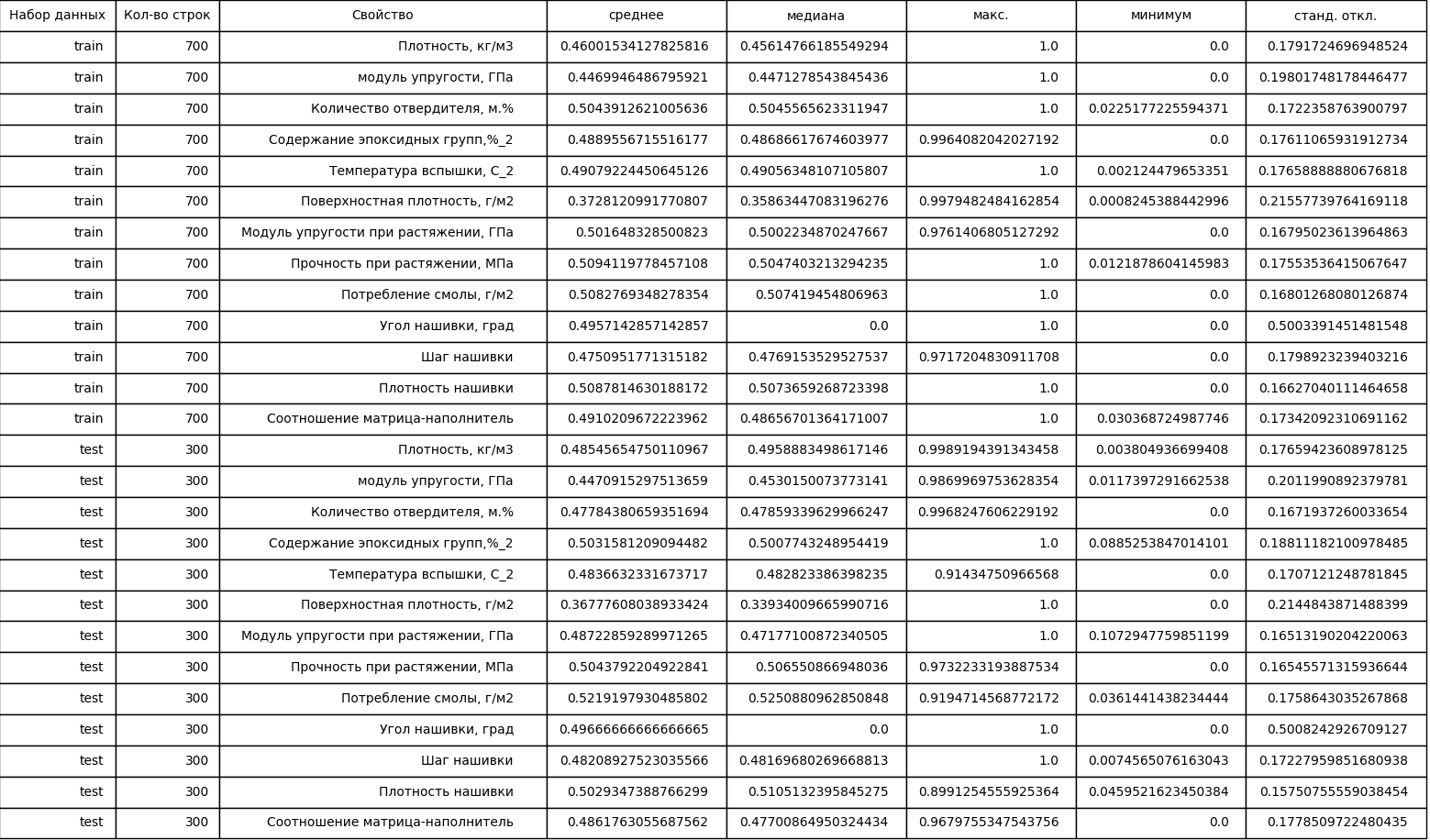


Таблица 17 – Описательная статистика для обучающего и тестового набора данных модуль упругости

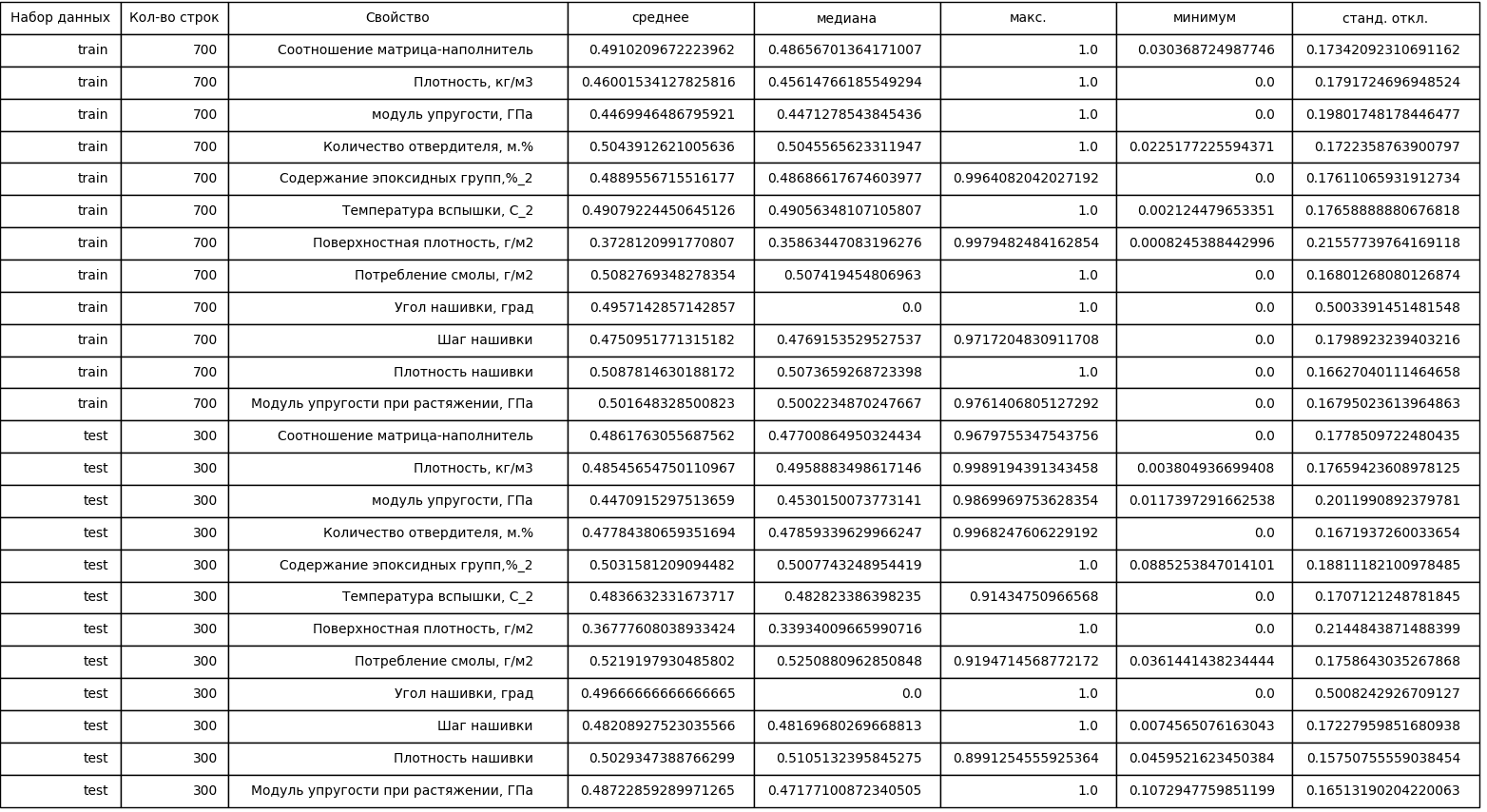
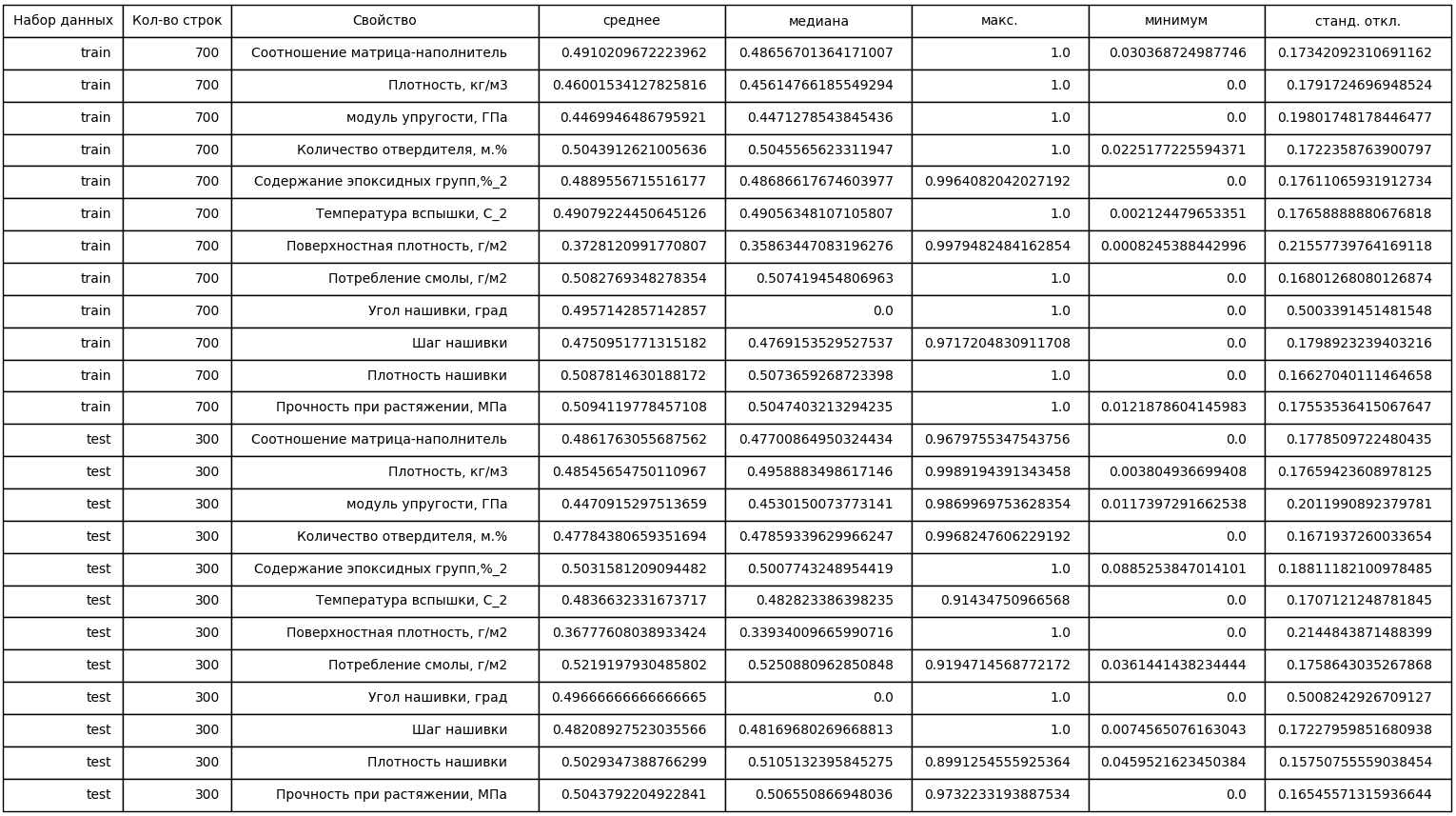


Таблица 18 – Описательная статистика для обучающего и тестового набора данных модуль прочности



На графиках видно, что настройка гиперпараметров улучшает качество работы модели.

В качестве рабочего варианта модели для предсказаний написания приложения выберем tensorflow, которая имеет встроенную функцию сохранение обученной модели, которую можно загрузить из приложения и использовать в работе. Создание модели, обучение, тестирование и отображение хода обучения, а также сохранение обученной модели в файл выполним скриптами 12.1\_keras\_\_modupr.py, 12.2\_keras\_modproch.py, и 12.3\_keras\_\_mat\_nap.py и соответственно получим файлы сохраненных моделей moduprmodel.keras, modprochrastmodel.keras и mat\_napmodel.keras. Ход обучения модели представлен на рисунках 16 – 18. Архитектура нейросети представлена на рисунке 19 и 20.

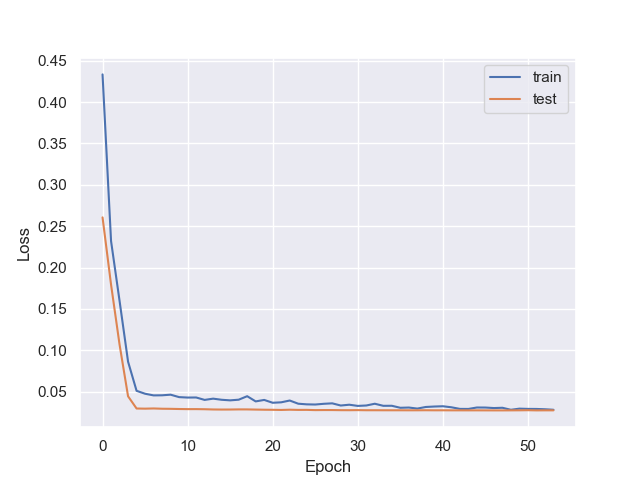
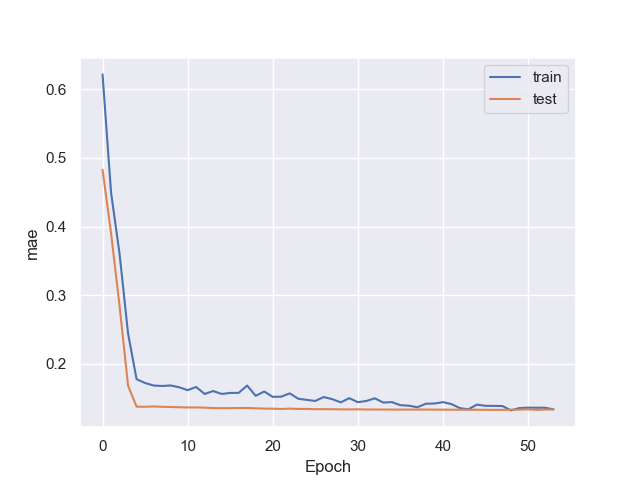


Рисунок 16 – Ход обучения модели для нахождения модуля упругости

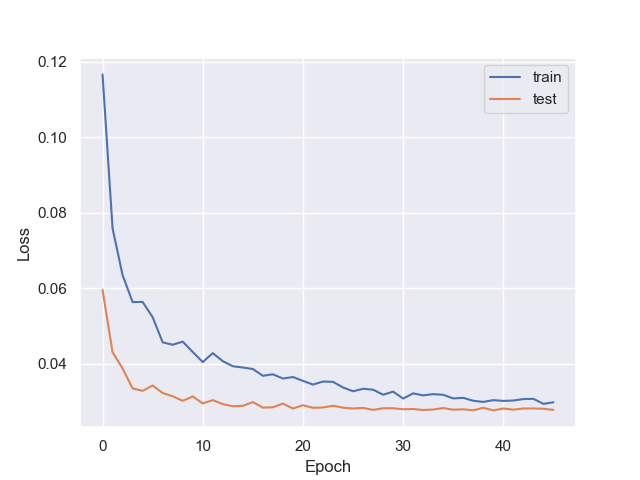
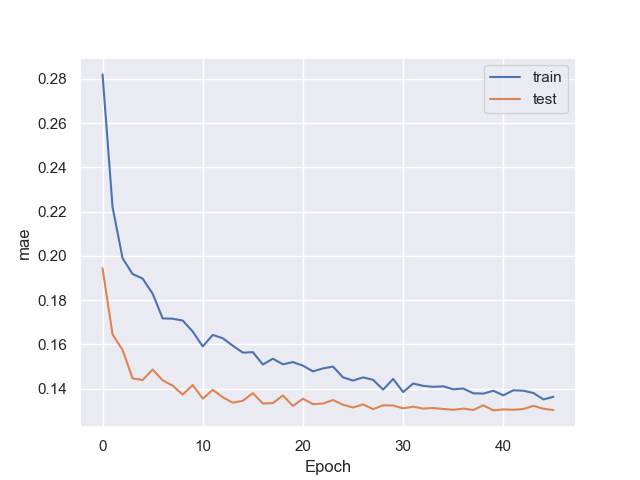


Рисунок 17 – Ход обучения модели для нахождения модуля прочности

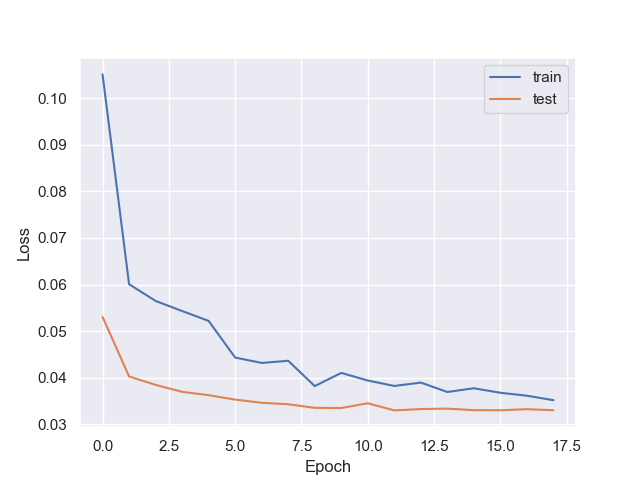
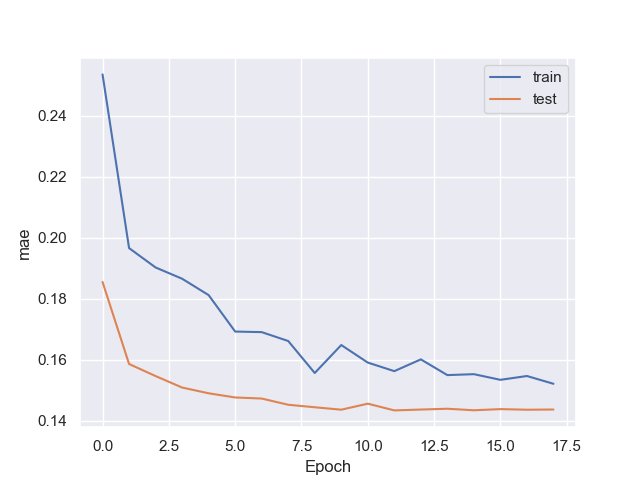


Рисунок 18 – Ход обучения модели для нахождения соотношения

материала – наполнителя

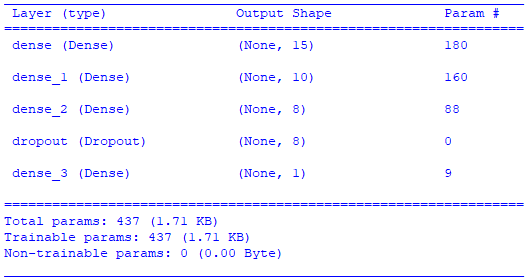


Рисунок 19 — Архитектура нейросети в виде summary

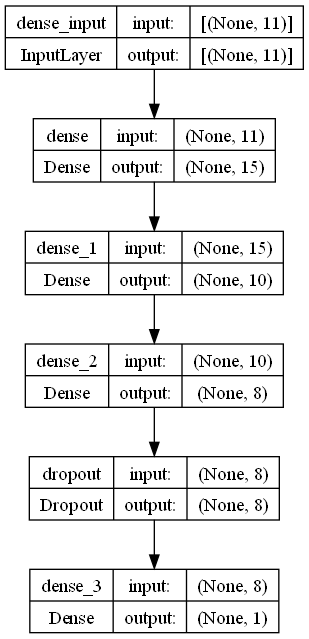


Рисунок 20 — Архитектура нейросети в виде графа

На графиках обучения видно, что с этапами обучения (эпохами) количество ошибок уменьшается, и когда результат практически не улучшается заканчивается обучение функцией ранней остановки.

* 1. 2.3 Разработка приложения

Консольное приложение (13\_consoleprogramms.py) напишем универсальное для всех предсказаний. Пользователь в приглашении командной строки будет выбирать задачу (1 - Определение Модуля упругости при растяжении, ГПа и Прочности при растяжении, МПа; 2 - Соотношения матрицы-наполнителя), а в цикле будет предлагаться ввести данные каждого из входных параметров либо подтвердить значение по умолчанию, также в цикле будут перебираться выходные параметры и модели для их получения. Список входных и выходных данных и моделей для их обработки сохранен в тесте скрипта, а граничные значения каждого из параметров будет подгружаться из json-файла, созданного на этапе первичного анализа данных. Также из json-файла будет подгружаться параметры для нормализации введенных пользователем значений и денормализации выходных данных после отработки оценщиком.

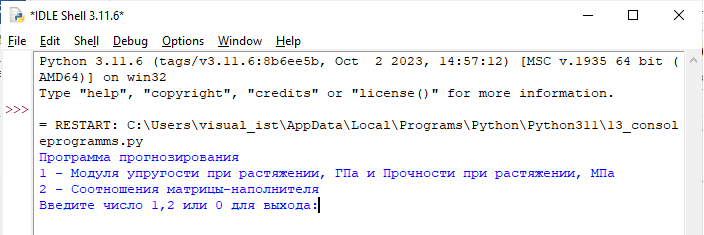


Рисунок 21 - Выбор задачи

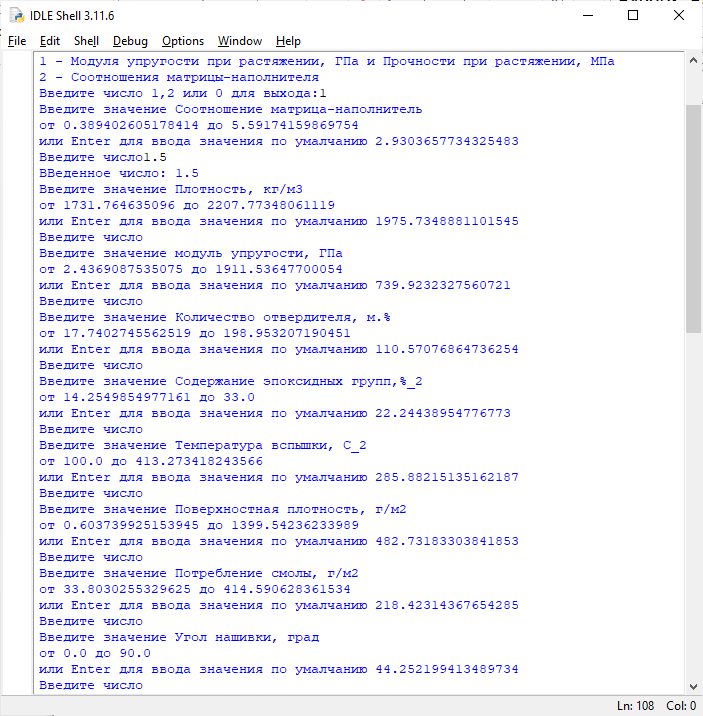


Рисунок 21 – Ввод данных

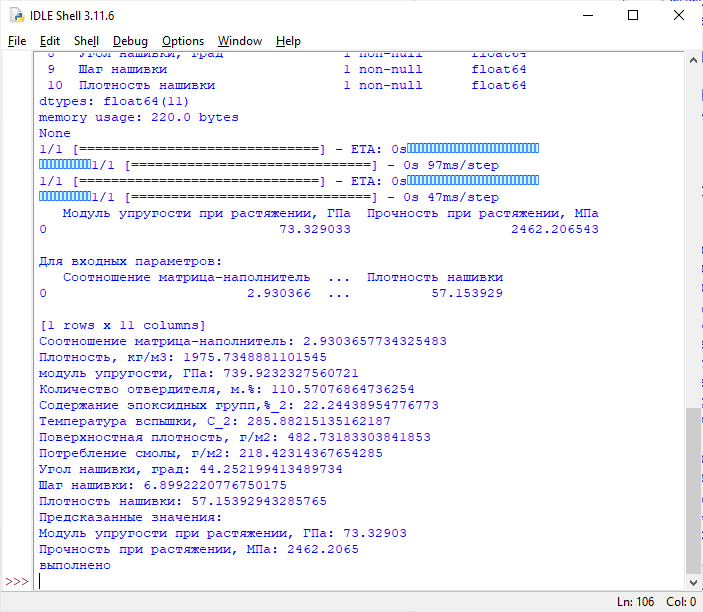


Рисунок 22 – Получение предсказанного значения

* 1. **Состав проекта**

Проект разрабатывался и тестировался в среде PostgreSQL 15.2, pgAdmin 4 ver. 6.2 и IDLE Python 3.11.5 и состоит из отдельных скриптов SQL-запросов и python-скриптов, каждый из которых выполняет отдельную подзадачу. SQL-скрипты размещены в каталоге src\sql, a python-скритпты – в каталоге src\python.

* 1. 2.4.1. Наименование и назначение SQL-скриптов

1. Create\_table\_X\_bp.sql - создает таблицу базы данных для импорта в нее данных файла X\_bp.xlsx;
2. X\_bp.csv – экспортированные данные из X\_bp.xlsx в текстовый формат с разделителем «;» между полями;
3. X\_bp4pg.csv – обработанный файл X\_bp.csv, у которого удален заголовок и заменен разделитель десятичных разрядов с «,» на «.»и пригодный для импортирования в PostreSQL средствами pgAdmin;
4. Create\_table\_X\_nup.sql - создает таблицу базы данных для импорта в нее данных файла X\_nup.xlsx;
5. X\_nup.csv – экспортированные данные из X\_nup.xlsx в текстовый формат с разделителем «;» между полями;
6. X\_nup4pg.csv – обработанный файл X\_nup.csv, у которого удален заголовок и заменен разделитель десятичных разрядов с «,» на «.» и пригодный для импортирования в PostreSQL средствами pgAdmin;
7. CountInX\_bp\_Xnup.sql – подсчитывает количество строк в таблицах X\_bp и X\_nup;
8. INNER\_JOIN.sql – производит INNER JOIN соединение таблиц X\_bp и X\_nup по индексному полю в новую таблицу full\_X\_bp;
9. ProverkaNaNull.sql – производит проверку таблиц на пустые данные;
10. statisticsbyfull.sql - вычисляет параметры описательной статистики (среднее, медиану, максимальное, минимальное и стандарное отклонение) из результирущего множества;
11. OtsevDataBy3Sigma.sql - отсеивает выбросы методом 3 сигм;
12. OtsevDataByInterval.sql – отсеивает выбросы методом межквартильных расстояний.

2.4.2. Наименование и назначение python-скриптов

1. 1\_PrimaryAnalizeInputData.py - загружает файлы X\_bp.xlsx и X\_nup.xlsx в pandas.DataFrame, выводит по ним общие сведения, выполняет соединение таблиц методам INNER JOIN, рассчитывает параметры описательной статистики по объединенной таблице, формирует таблицу результатов. Кроме этого выполняет сохранение минимального, максимального и среднего значения для каждого поля.
2. minmaxmeans.json - сохраненные значения минимум, максимум среднее для каждого поля. Требуется для работы разработанного консольного приложения.
3. 2\_BuildDisplotAndBoxplot\_1.py – построение графиков гистограмм распределения и диаграмм «ящик с усами».
4. 3\_BuildPearGrid\_1.py - построение попарных графиков рассеяния точек.
5. 4\_OtsevBy3SigmAndIntervals.py – отсев выбросов методом 3 сигм и межквартильных расстояний.
6. 5\_OtsevBy3SigmAndBuildsGrafs.py – отсев выбросов методом 3 сигм, построение графиков по отсеянному множеству. Сохранение очищенной коллекции в файл clearing\_X\_bp\_nup.xlsx.
7. clearing\_X\_bp\_nup.xlsx – очищенные от выбросов данные, результат работы скрипта 5\_OtsevBy3SigmAndBuildsGrafs.py.
8. 6\_normalizedata.py – нормализация данных, сохранение нормализаванной коллекции в файл normalized\_data.xlsx. Кроме этого сохраняет коэффициенты и дельты для каждого поля, с которыми проводилась нормализация.
9. mnojanddelta.json – содержит сохраненные коэффициенты и дельты для каждого поля, с которыми проводилась нормализация. Результат работы скрипта 6\_normalizedata.py. Необходим для работы пользовательского приложения.
10. normalized\_data.xlsx – нормализованные данные, результат работы скрипта 6\_normalizedata.py.
11. 7\_CreateDataFor3Models.py – создает 3 датасета для работы моделей для нахождения модуля упругости при растяжении, прочности при растяжении и соотношения матрица-наполнитель.
12. 9\_getAllgiperparams.py – запрашивает все гиперпараметры каждой функции регрессии из библиотеки sklearn.linear, используемой в проекте.
13. 9.1\_podbor\_parameters\_DecisionTreeRegressor.py – подбор гиперпараметров для DecisionTreeRegressor для всех 3-х датасетов.
14. 9.1\_podbor\_parameters\_KNeighborsRegressor.py – подбор гиперпараметров для KNeighborsRegressor для всех 3-х датасетов.
15. 9.1\_podbor\_parameters\_Lasso.py – подбор гиперпараметров для Lasso для всех 3-х датасетов.
16. 9.1\_podbor\_parameters\_svm.py – подбор гиперпараметров для SVR для всех 3-х датасетов.
17. 9.1\_podbor\_parameters\_Ridge.py – подбор гиперпараметров для Ridge для всех 3-х датасетов.
18. 9.1\_podbor\_parameters\_GradientBoosting.py – подбор гиперпараметров для GradientBoosting для всех 3-х датасетов.
19. 9.1\_podbor\_parameters\_RandomForestRegressor.py – подбор гиперпараметров для RandomForestRegressor для всех 3-х датасетов.
20. 9\_examine\_mat\_nap.py – испытание всех моделей на 3-x датасетах, с настройками по умалчанию и оптимальных, на обучающем множестве и тестовом, сбор статистики качества работы модели по разным оценочным метрикам.
21. 12.1\_keras\_\_modupr.py – построение и обучение модели для определения модуля упругости в tensorflow и сохранение обученной модели на диске в виде файла moduprmodel.keras.
22. moduprmodel.keras – сохраненная модель, необходима для работы приложения.
23. 12.2\_keras\_modproch.py – построение и обучение модели для определения модуля прочности в tensorflow и сохранение обученной модели на диске в виде файла modprochrastmodel.keras.
24. modprochrastmodel.keras – сохраненная модель, необходима для работы приложения.
25. 12.3\_keras\_\_mat\_nap.py – построение и обучение модели для определения соотношения матрица - наполнитель в tensorflow и сохранение обученной модели на диске в виде файла mat\_napmodel.keras.
26. mat\_napmodel.keras – сохраненная модель, необходима для работы приложения.
27. 13\_consoleprogramms.py – консольная программа для предсказания модуля упругости при растяжении и прочности при растяжении, рекомендации соотношения матрица-наполнитель.
    1. **Создание удаленного репозитория**

Для данного исследования был создан удаленный репозиторий на GitHub, который находится по адресу <https://github.com/MilaTokProg/tokareva-bmstu-ds-course>. На него были загружены результаты работы: файлы исследования Python и файл приложения.

1. Библиографический список

1. Лекции курса «Data Science» https://sdo.bmstu.ru

2. Документация по библиотеке pandas: - Режим доступа: <https://pandas.pydata.org>.

3. Документация по библиотеке matplotlib: - Режим доступа: <https://matplotlib.org>.

4.Документация по библиотеке seaborn: - Режим доступа: <https://seaborn.pydata.org>.

5. Документация по библиотеке sklearn: - Режим доступа: <https://scikit-learn.org>.