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 Быстренина И.Е.

Москва, 2023

Содержание:

[**Введение** 3](#_Toc147987095)

[**1.Аналитическая часть** 5](#_Toc147987096)

[1.1. Постановка задачи 5](#_Toc147987097)

[1.2. Описание используемых методов 5](#_Toc147987099)

[1.3. Разведочный анализ данных 8](#_Toc147987100)

[1.4. Гистограммы 10](#_Toc147987101)

[1.5. Попарные графики рассеивания 11](#_Toc147987102)

[1.6. Визуализация распределения по целевым переменным, выбросы 12](#_Toc147987103)

[1.7. Матрица корреляции признаков 14](#_Toc147987104)

[1.8. Диаграмма "Ящик с усами" (Boxplot) 15](#_Toc147987105)

[**2 Предобработка данных** 17](#_Toc147987106)

[2.1 Удаление выбросов 17](#_Toc147987107)

[2.2 Нормализация 18](#_Toc147987108)

[2.3 Описательная статистика характеристик после предобработки 19](#_Toc147987109)

[**3 Модели для прогноза целевых переменных** 20](#_Toc147987112)

[3.1  Линейная регрессия 23](#_Toc147987113)

[3.2 Регрессия k-ближайших соседей 26](#_Toc147987114)

[3.3 Метод регрессии опорных векторов SVR 28](#_Toc147987115)

[3.4 Random Forest Regressor 29](#_Toc147987116)

[**4 Рекомендательная нейросеть для соотношения матрица-наполнитель** 32](#_Toc147987117)

[**5 Создание Flask приложения** 34](#_Toc147987118)

[**Заключение** 38](#_Toc147987119)

[**Список использованной литературы** 39](#_Toc147987120)

# **Введение**

Тема ВКР - 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. Современные композиты изготавливаются из разных 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).

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.

**Актуальность исследования** обусловлена тем, что созданные прогнозные модели помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов возможными новыми характеристиками материалов, и цифровыми двойниками новых композитов.

**Целью работы** является прогнозирование конечных свойств новых материалов (композиционных материалов).

Для реализации поставленной цели необходимо решить ряд **задач**:

1. Проведение разведочного анализа и предобработки данных.
2. Разработка, обучение и тестирование модели.
3. Написание нейронной сети, которая будет рекомендовать соотношение матрицы.
4. Разработка приложения для прогнозирования конечных свойств новых материалов (композиционных материалов).

# **1.Аналитическая часть**

## 1.1. Постановка задачи

## Входной информацией для обработки послужили данные о начальных свойствах компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.). Исходная информация была представлена в 2 файлах .xlsx. Файл X\_bp имеет размерность (1023, 11), X\_nup имеет размерность (1040, 4).

В данных отсутствуют пропущенные значения, качественные переменные и выбросы.

## 1.2. Описание используемых методов

В машинном обучение большое множество алгоритмов – некоторые схожи между собой, некоторые не имеют ничего общего. Для решения задач регрессии могут быть использованы следующие алгоритмы:

* деревья решений;
* случайный лес;
* линейная регрессия;
* метод ближайшего соседа;
* градиентный бустинг;
* нейронные сети;
* машина опорных векторов;
* и др.

*Бустинг (AdaBoost)* – это процедура построения алгоритмов, суть которого заключается в построении таких алгоритмов, когда каждый следующий старается компенсировать недостатки предыдущих. Алгоритм AdaBoost строит композицию из алгоритмов обучения для улучшения эффективности. Он строится по тем объектам, которые были плохо классифицированы ранее.

*Случайный лес* – процедура построения алгоритмов, суть которого параллельное обучение алгоритмов, с последующим объединением. Случайный лес – это алгоритм, который объединяет несколько деревьев решений на основе идеи ансамблевого обучения. Объединение деревьев позволяет получить более точные и стабильные прогнозы.

*Дерево решений (CART, C4.5)* – метод применения решающих правил в иерархической структуре. Алгоритм CART позволяет использовать дискретные и непрерывные переменные, при этом в узлах может быть только 2 потомка. Алгоритм C4.5 является улучшенной версией другого алгоритма, и позволяет работать только с дискретной переменной, но при этом может работать с пропущенными данными и нет ограничений на число потомков в узлах.

*Линейная регрессия* – алгоритм, измеряющий связь между некоторыми переменными с применением линейной функции.

*Машина опорных векторов* – алгоритм бинарной классификации, использует линейное разделение пространства. Построение по правилам разделяющей гиперплоскости обеспечивает более точное разделение классов, а также уменьшает ошибку разделения.

*Метод k-ближайшего соседа* – алгоритм, который производит распределение объекта к определенному классу по его ближайшим k соседям.

*Наивный байесовский классификатор* – простой алгоритм классификации. Идеей алгоритма является применение теоремы Байеса – переменные в наборе данных не коррелируют между собой.

*Нейронные сети* – математическая модель, а также воплощение, построенные по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма.

Работа выполнена в среде выполнения Google Colab на языке программирования Python.

Загружаем необходимые библиотеки.

import sklearn – библиотека машинного обучения.

import pandas as pd - для обработки и анализа структурированных данных.

import numpy as np - для математических вычислений.

import seaborn as sns – для создания статистических графиков на Python.

import matplotlib.pyplot as plt - для визуализации данных.

import flask - фреймворк для создания веб-приложений на языке программирования Python.

from sklearn.svm import SVR – метод опорных векторов.

from sklearn.multioutput import MultiOutputRegressor – для многоцелевой регрессии.

from sklearn.preprocessing import RobustScaler – алгоритм нормализации.

from sklearn import preprocessing, metrics Пакет preprocessing для преобразования необработанных векторов признаков в представление, более подходящее для последующих оценок. Пакет metrics включает метрики оценки для подбора моделей.

from sklearn.model\_selection import train\_test\_split, GridSearchCV, cross\_val\_score - метод train\_test\_split разделяет массивы или матрицы на случайные тренировочные и тестовые подмножества. GridSearchCV обеспечивает исчерпывающий поиск по заданным значениям параметров для оценщика.

Функция cross\_val\_score для перекрестной проверки.

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error. Метрики ошибок – среднеквадратичная ошибка регрессии, оценка регрессии по коэффициенту детерминации r2, средняя абсолютная ошибка регрессии.

from sklearn.neighbors import KNeighborsRegressor. Метод регрессии, основанный на k-ближайших соседях.

from sklearn.linear\_model import LinearRegression. Метод линейной регрессии.

from sklearn.ensemble import RandomForestRegressor – алгоритм регрессии случайных лесов.

import tensorflow as tf - библиотека для построения и тренировки нейронных сетей.

from tensorflow import keras - библиотека для глубокого машинного обучения.

from tensorflow.keras import layers - для работы со слоями.

from tensorflow.keras.models import Sequential – модель последовательного обучения ИНС.

from tensorflow.keras.layers import Dense, Dropout, BatchNormalization – слои ИНС.

Версия tensorflow 2.13.0

## 1.3. Разведочный анализ данных

Для загрузки данных импортируем необходимые библиотеки:

import sklearn

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib

import matplotlib.pyplot as plt

import flask

from pandas import DataFrame

from sklearn import preprocessing, svm, metrics

from sklearn.svm import SVR

from sklearn.multioutput import MultiOutputRegressor

from sklearn.preprocessing import RobustScaler

from sklearn.model\_selection import train\_test\_split, GridSearchCV, cross\_val\_score

from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error

from sklearn.ensemble import RandomForestRegressor

from sklearn.linear\_model import LinearRegression

from sklearn.neighbors import KNeighborsRegressor

%matplotlib inline

import warnings

warnings.filterwarnings("ignore")

Загрузим исходные таблицы в виде data frame:

data1 = pd.read\_excel('/content/X\_bp.xlsx')

data2 = pd.read\_excel('/content/X\_nup.xlsx')

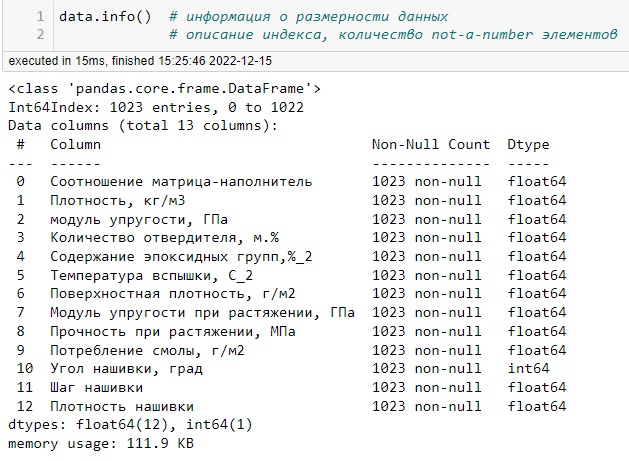
Объединим таблицы в один data frame.

Тип объединения – INNER, объединение по индексам. Означает, что объединяются только те значения, которые можно найти в обеих таблицах.

data = data1.merge(data2, how = 'inner', left\_index=True, right\_index=True)

data.head()

Размерность объединенного датасета: data.shape - 1023 строк, 13 столбцов. Методом data.info() смотрим информацию о размерности данных.



Видим тринадцать столбцов, нулевых значений нет, у всех данных тип float64.

Методом data.describe() смотрим описательную статистику (Рисунок 1). Метод показывает количество значений, среднее значение, стандартное отклонение, минимальное значение, 25-50-75% перцентили и максимальное значение.

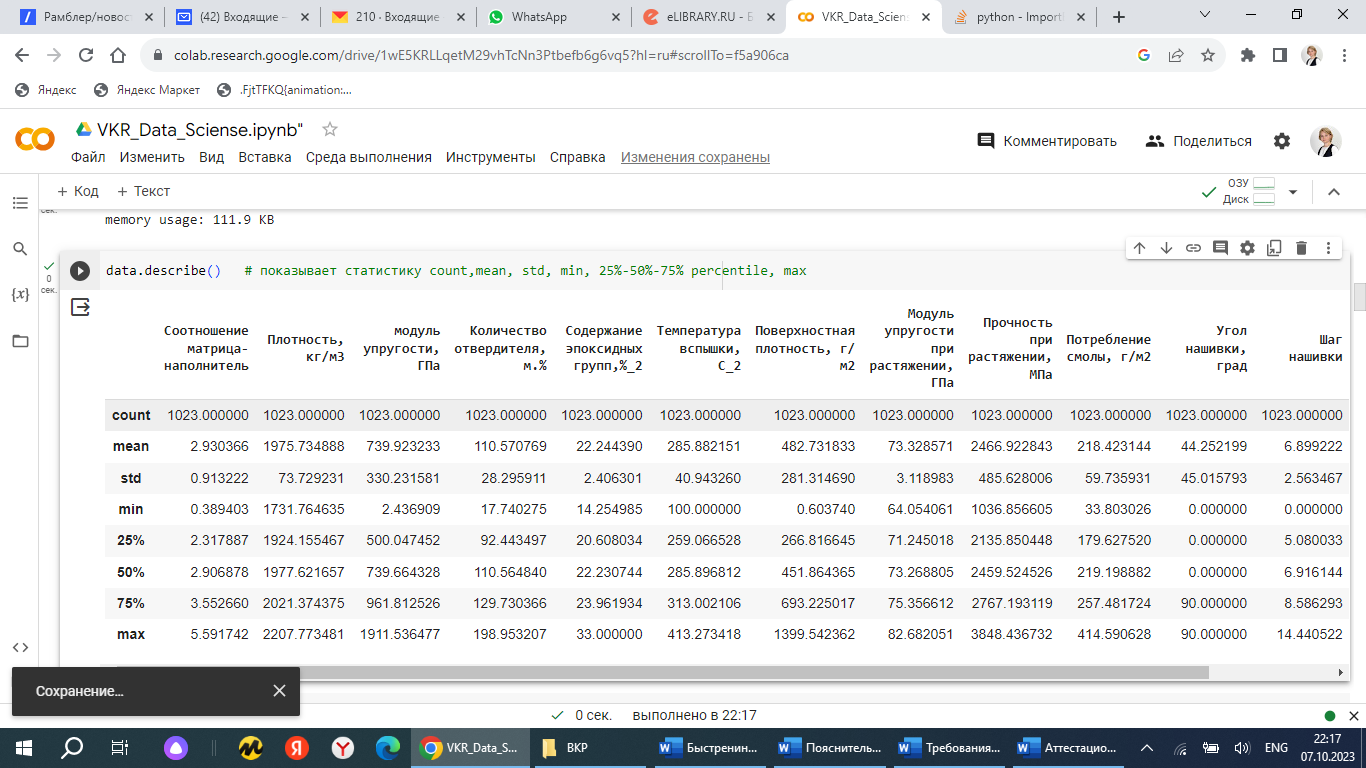


Рисунок 1

По всем признакам есть существенный разброс значений.

Подсчет уникальных значений по столбцам методом data.nunique() показывает, что почти все значения уникальны (Рисунок 2). Кроме признака «Угол нашивки», в нем два значения – 0 и 90 градусов.

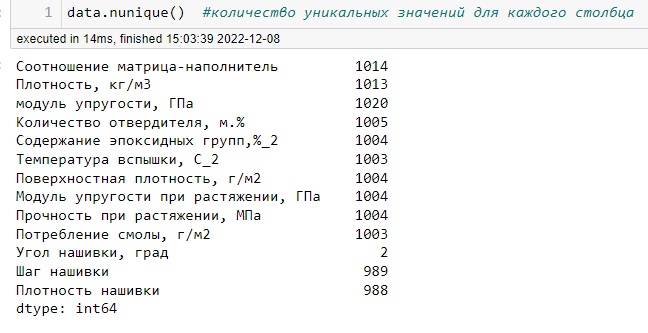


Рисунок 2

Проверяем количество пропусков в сумме по столбцам методом data.isnull().sum(). Пропусков нет (Рисунок 3).

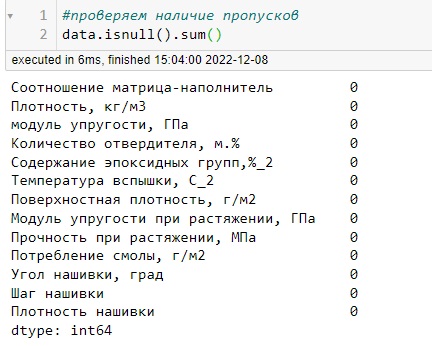


Рисунок 3

## 1.4. Гистограммы

def histogram (data, n\_rows, n\_cols):

    fig=plt.figure(figsize=(20, 15))

    for i, column in enumerate(data.columns):

        ax=fig.add\_subplot(n\_rows,n\_cols,i+1)

        sns.histplot(data=data, x=column, kde=True, bins=35, color = 'grey')

    fig.tight\_layout()

    plt.show()

histogram(data, 4, 4)

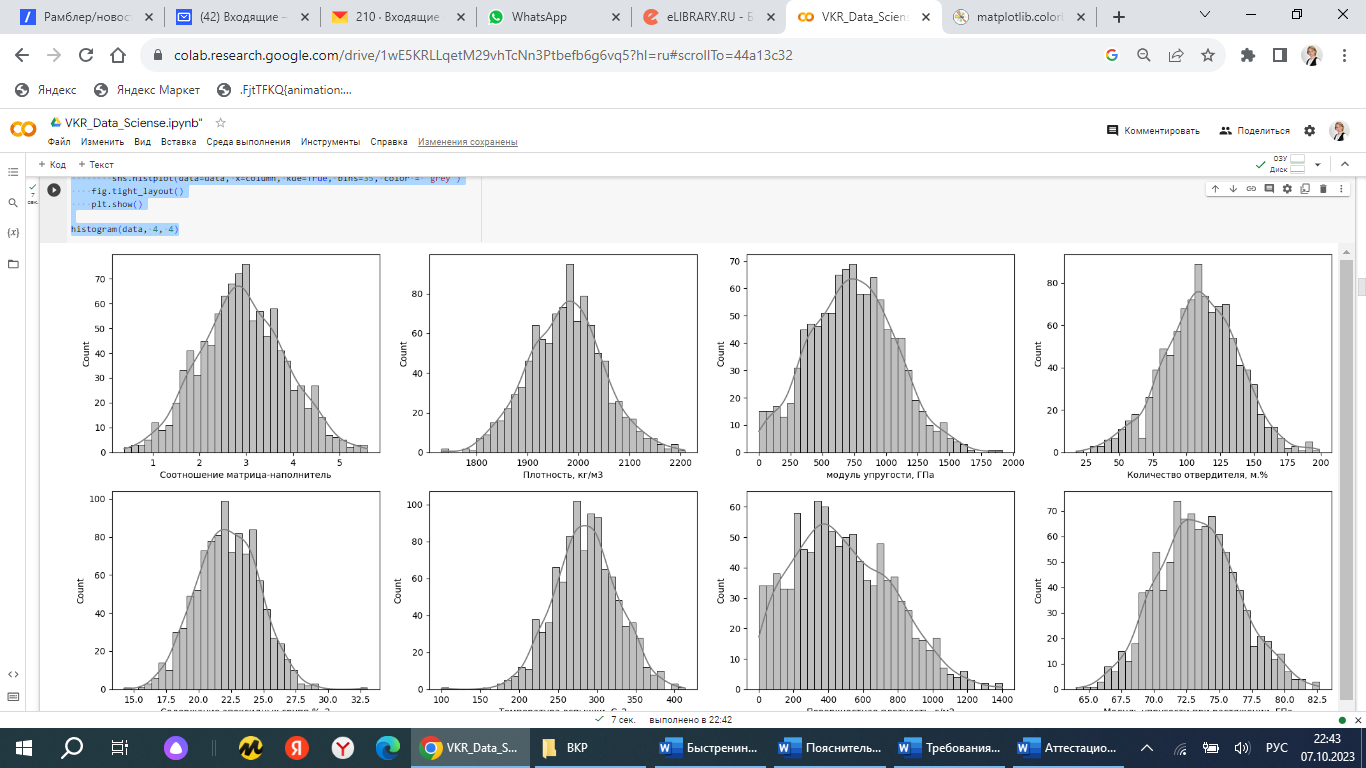




Рисунок 4

По гистограммам (Рисунок 4) видно, что данные имеют нормальное распределение. Есть небольшое смещение.

## 1.5. Попарные графики рассеивания

sns.pairplot(data, height=4, diag\_kind='kde', corner = True)

По графикам попарного рассеивания (Рисунок 5) видим, что данные рассеяны, низко связаны между собой.

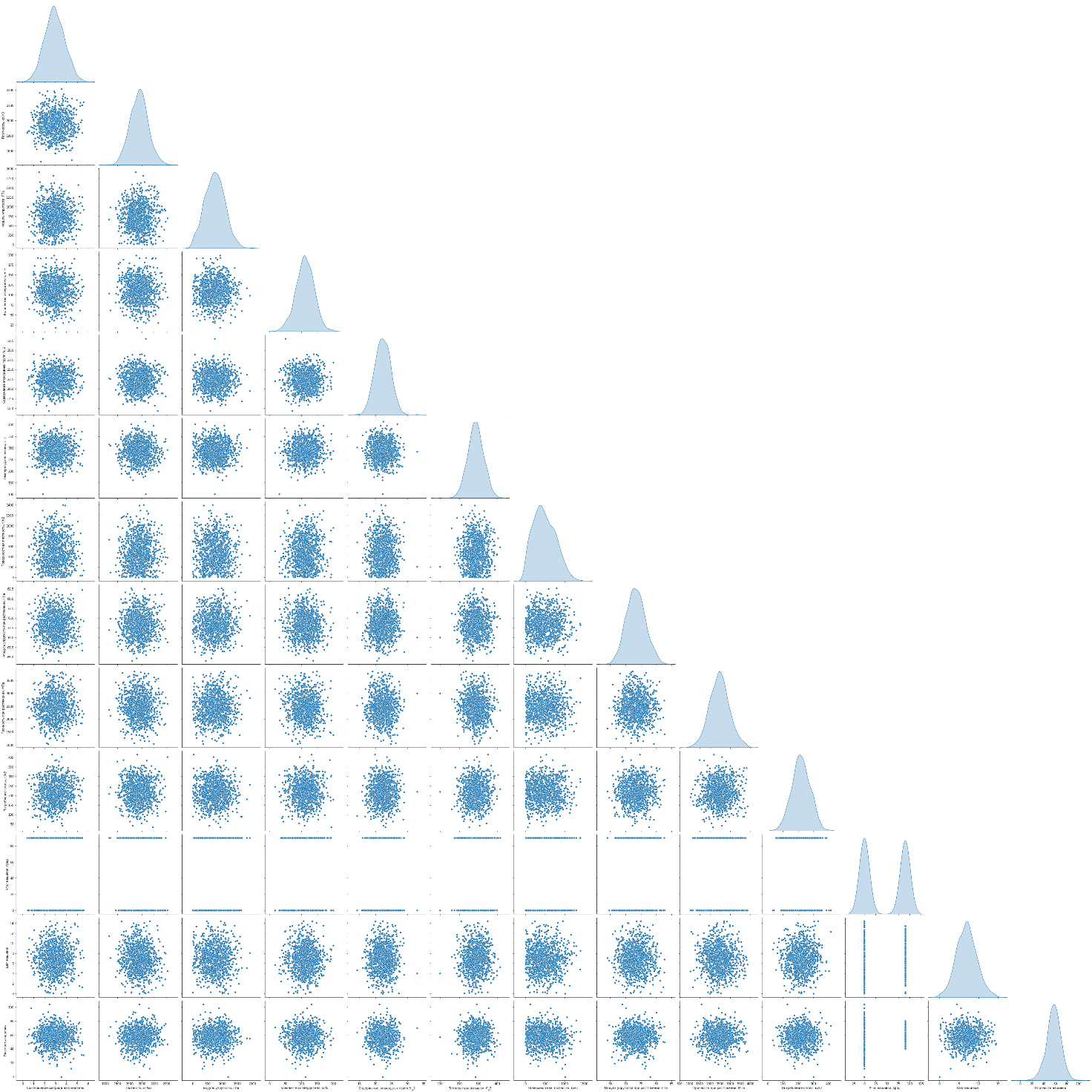


Рисунок 5

## 1.6. Визуализация распределения по целевым переменным, выбросы

По первой переменной (Рисунок 6):

sns.pairplot(data, hue="Модуль упругости при растяжении, ГПа")

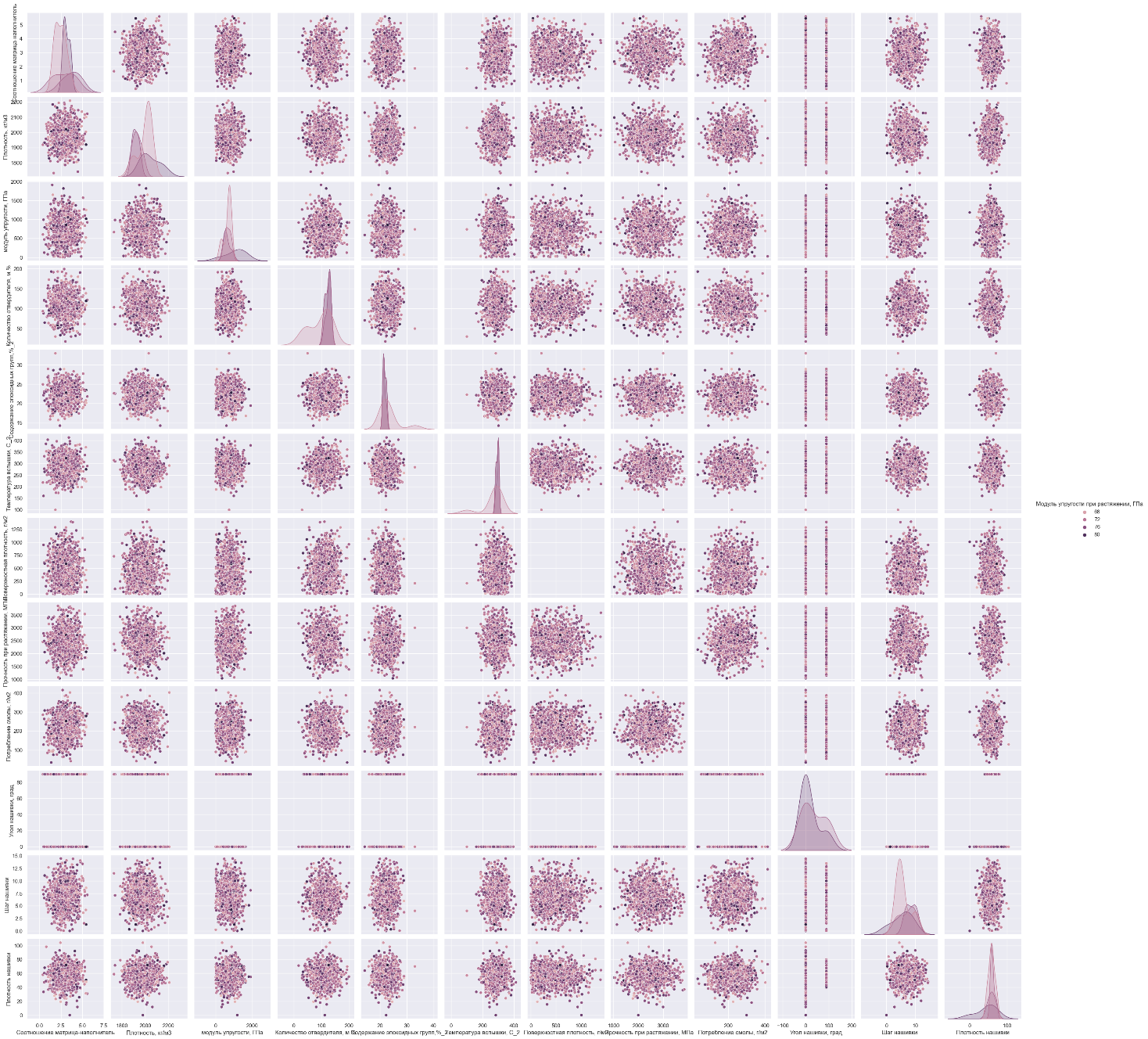


Рисунок 6

По второй переменной (Рисунок 7):

sns.pairplot(data, hue="Прочность при растяжении, МПа")

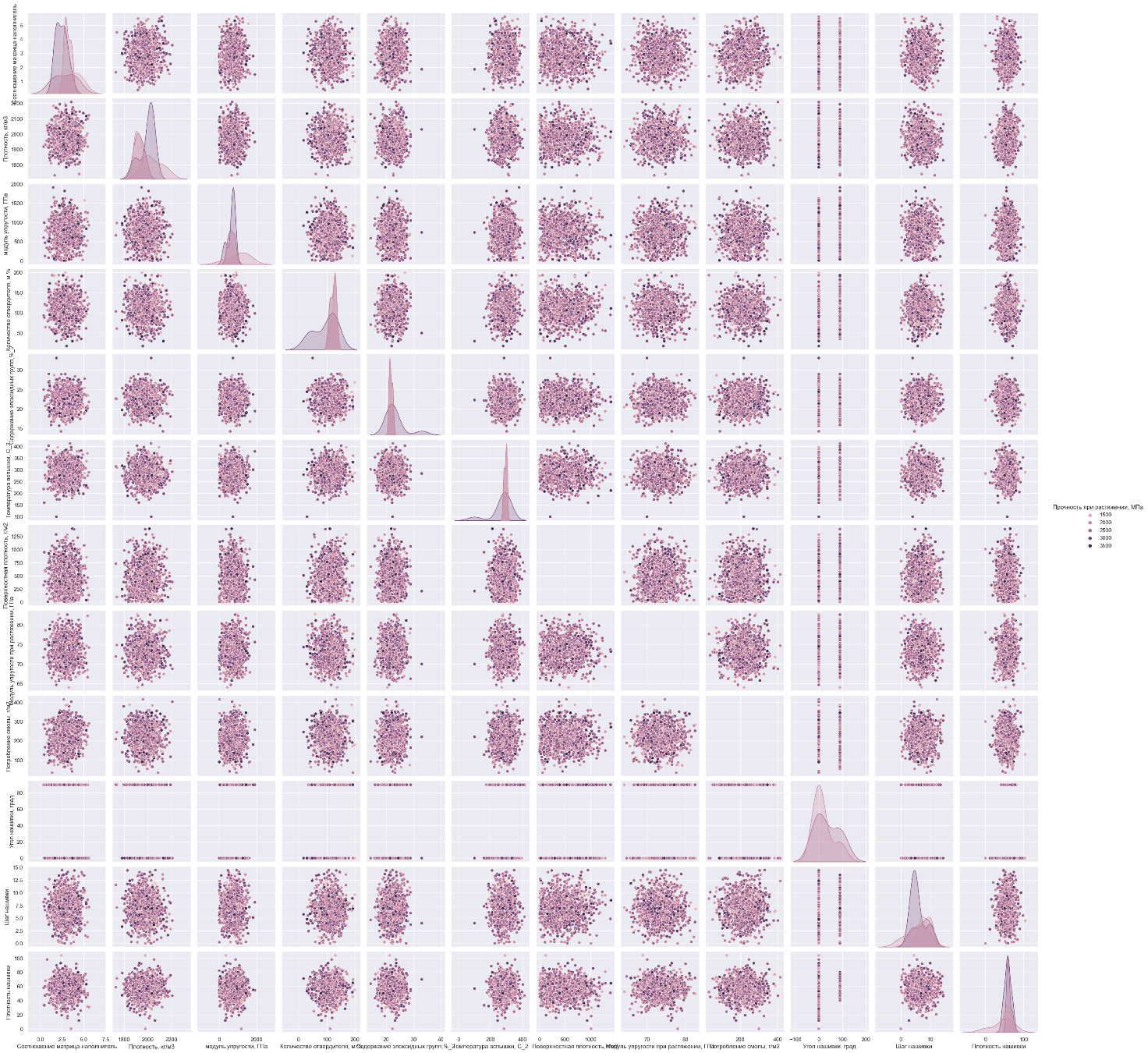


Рисунок 7

## 1.7. Матрица корреляции признаков

corr = data.corr()

fig, ax = plt.subplots(figsize=(16, 11))

sns.heatmap(corr, vmin=-0.5, vmax=0.5, annot=True, fmt='.2f',cmap='PRGn', ax=ax, linewidths = 0.1)

plt.show()

Из матрицы (Рисунок 8) видим, что корреляция низкая. Максимальное значение корреляции 0,11. Что говорит о том, что между признаками нет линейной зависимости.

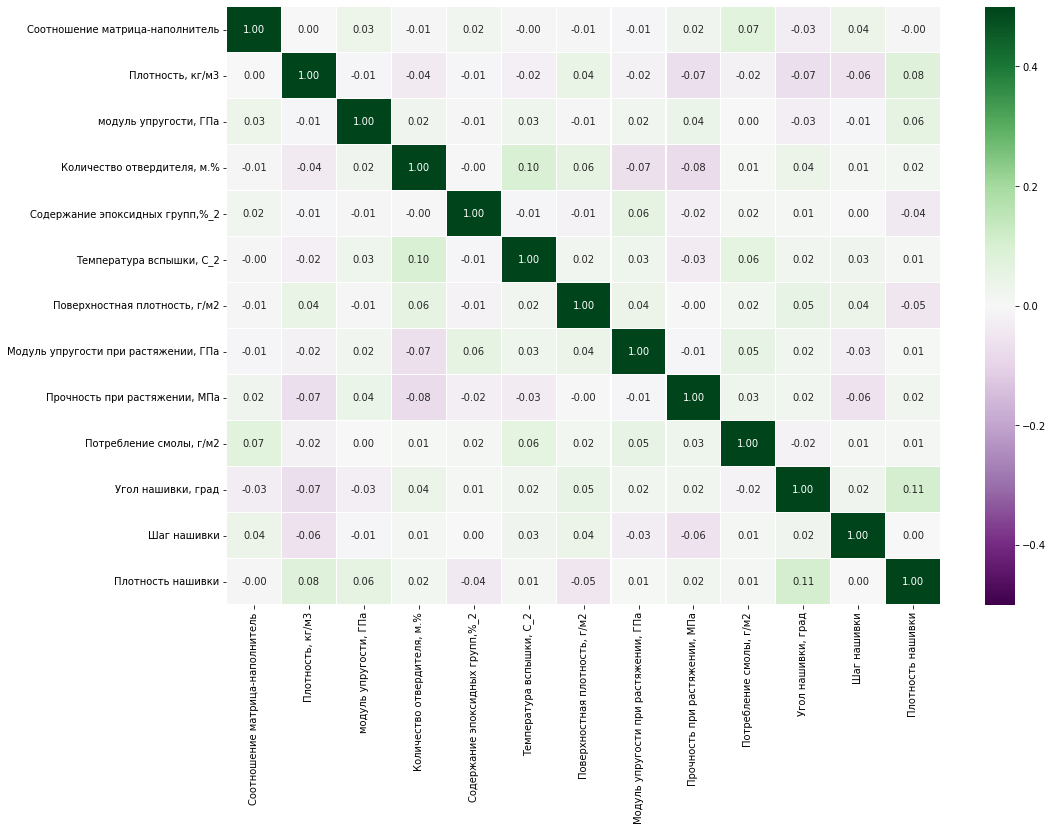


Рисунок 8

## 1.8. Диаграмма "Ящик с усами" (Boxplot)

Чтобы посмотреть наличие выбросов строим диаграмму «Ящик с усами». На «усах» ящика видны выбросы (Рисунок 9).

def box\_pl(data):

  col = 5  # столбцы

  k = 1  # счетчик

  row = 3  # строки

  fig = plt.figure(figsize=(15, 30))

  for i in data.columns:

      plt.subplot(row, col, k)

      plt.xlabel(i)

      sns.boxplot(y = data[i])

      k+=1

  plt.show()

box\_pl(data)

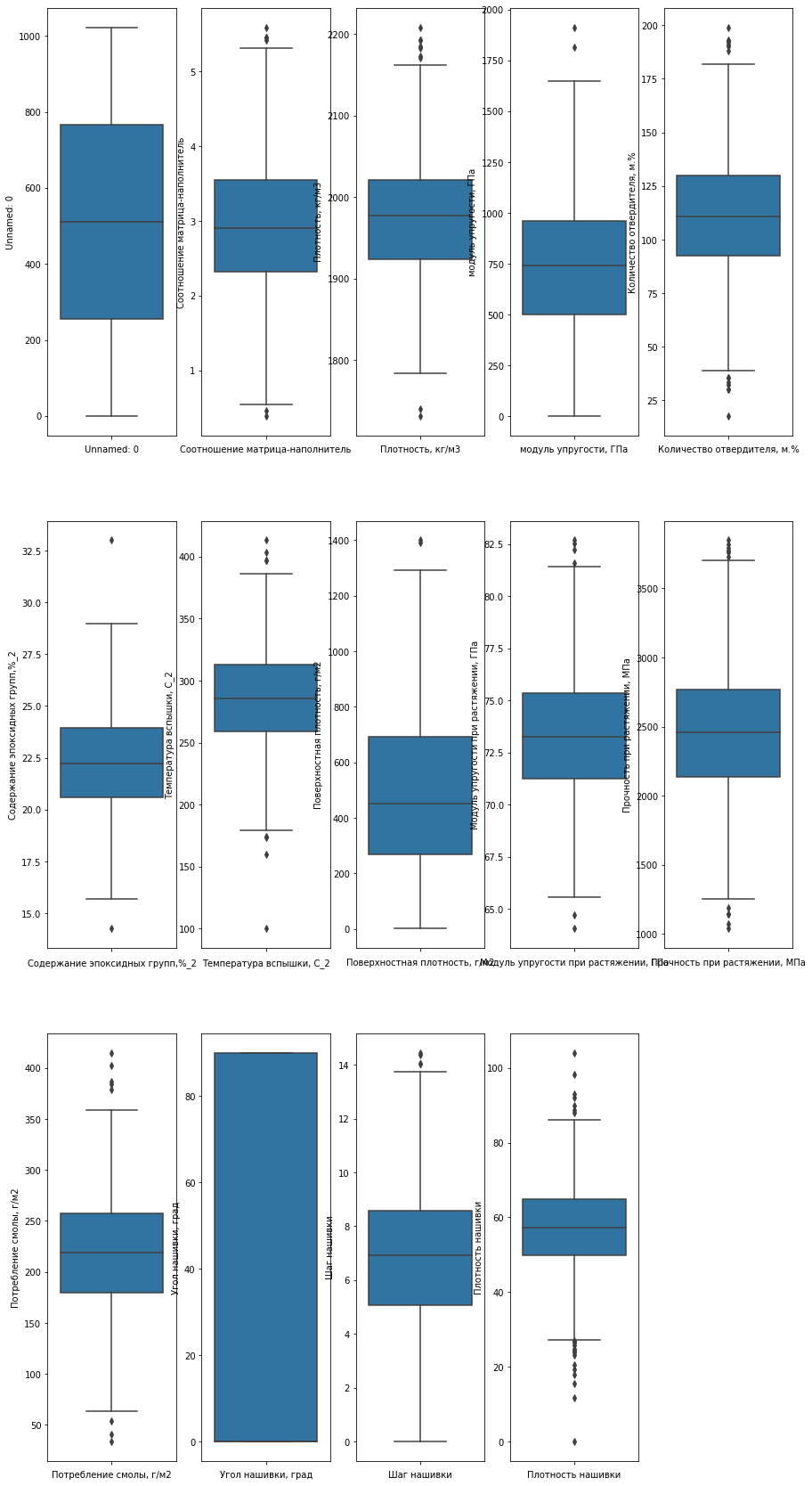


Рисунок 9

# **2 Предобработка данных**

## 2.1 Удаление выбросов

Предобработку данных начнем с удаления выбросов с помощью метода межквартильных расстояний.

for i in data.columns:

    q75, q25 = np.percentile(data.loc[:,i], [75,25])

    intr\_qr = q75 - q25

    max = q75 + (1.5 \* intr\_qr)

    min = q25 - (1.5 \* intr\_qr)

    data.loc[data[i] < min, i] = np.nan

    data.loc[data[i] > max, i] = np.nan

data.dropna(inplace = True)

Функция percentile() вычисляет q-й процентиль (перцентиль) значений элементов массива или элементов вдоль указанной оси.

Результат функции - массив NumPy или число. Если параметр q - это одно число, то результатом так же будет одно число. Если параметр q - это массив из нескольких процентилей, то будет возвращен массив аналогичной длины. Если входной массив имеет несколько осей, указано несколько процентилей и указана ось (или указано несколько осей) в параметре axis, то будет возвращен массив, по первой оси которого расположены значения (массивы значения) для каждого указанного в q процентиля.

Чтобы оценить результат строим "Ящик с усами" после удаления выбросов (Рисунок 10).

box\_pl(data)

sns.boxplot(data = data, orient="h")

plt.show()

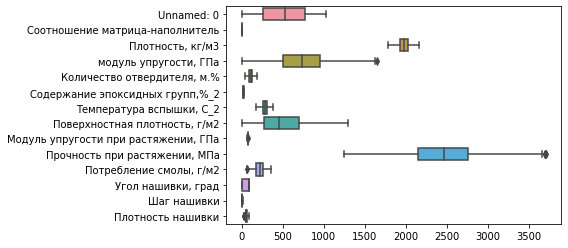


Рисунок 10

Выбросов стало меньше. Посмотрим теперь на размерность датасета методом data.shape(). Осталось 936 строк. То есть удалилось около 10 процентов данных.

## 2.2 Нормализация

Для нормализации данных применим RobustScaler, который, в отличие от MinMaxScaler, StandardScaler менее чувствителен к выбросам.

class sklearn.preprocessing.RobustScaler(\*, with\_centering=True, with\_scaling=True, quantile\_range=25.0, 75.0, copy=True, unit\_variance=False) [[источник]](https://github.com/scikit-learn/scikit-learn/blob/95119c13a/sklearn/preprocessing/_data.py#L1221)

Функция масштабирования с использованием статистики, которая устойчива к отклонениям.

Этот Scaler удаляет медиану и масштабирует данные в соответствии с квантильным диапазоном (по умолчанию IQR: Interquartile Range). IQR-это диапазон между 1-м квартилем (25-й квантиль)и 3-м квартилем (75-й квантиль).

Центрирование и масштабирование происходят независимо для каждой функции путем вычисления соответствующей статистики по выборкам в обучающем наборе. Затем медиана и межквартильный размах сохраняются для использования в последующих данных с помощью метода transform.

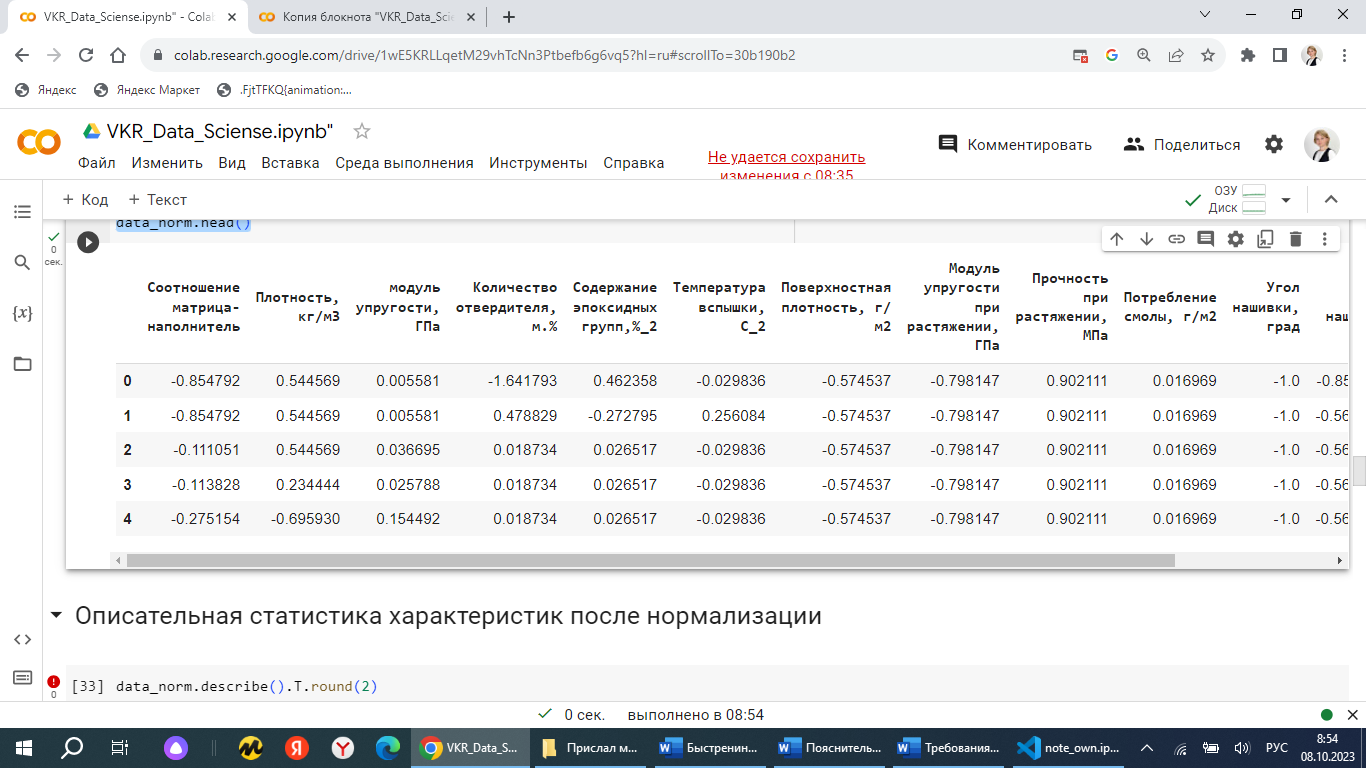
scaler = preprocessing.RobustScaler()

columns = data.columns

data\_norm = scaler.fit\_transform(np.array(data))

data\_norm = pd.DataFrame(data\_norm, columns=columns)

data\_norm.head()



## 2.3 Описательная статистика характеристик после предобработки

Смотрим описательную статистику (Рисунок 11) признаков методом describe().

data\_norm.describe().T.round(2)

Данные нормализованы.

## 

## Рисунок 11

Также посмотрим на «ящик с усами» (Рисунок 12) после нормализации:

sns.set(rc={'figure.figsize':(20,12)})

ax = sns.boxplot(data=data\_norm)

ax.set\_xticklabels(ax.get\_xticklabels(),rotation=90);

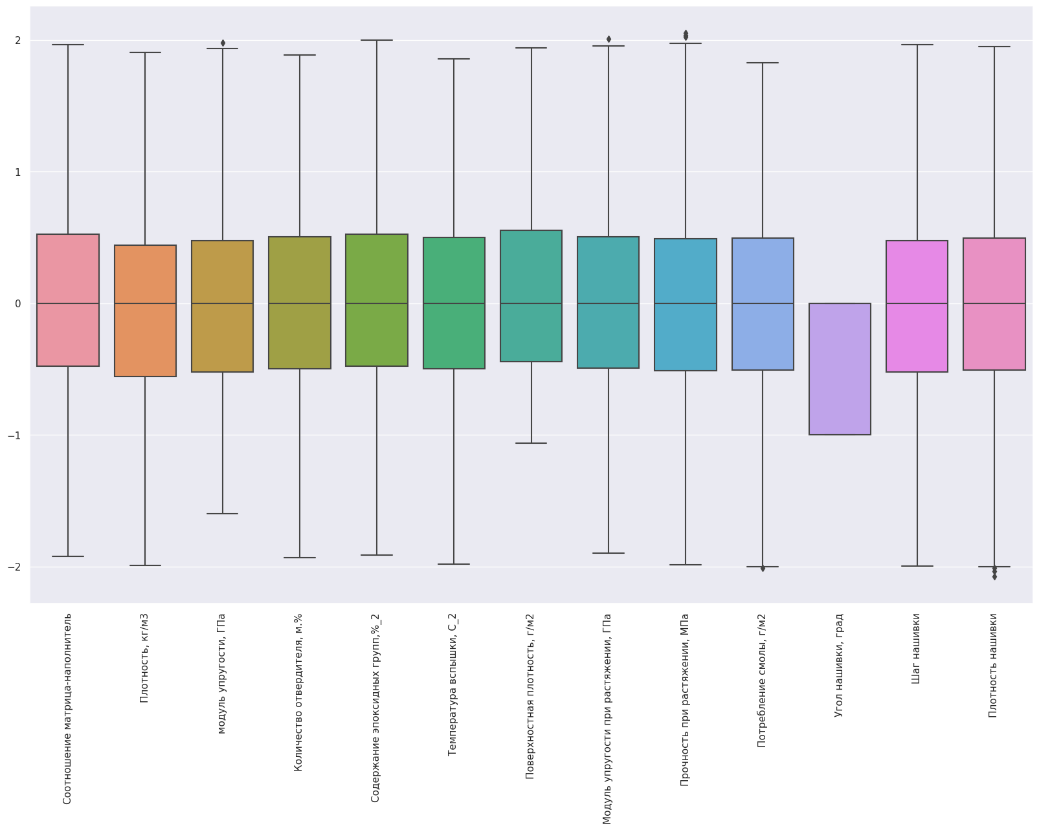


Рисунок 12

Данные приведены к одному диапазону, выбросы остались незначительные. Можно приступать к построению моделей для прогноза целевых переменных.

# **3 Модели для прогноза целевых переменных**

Входы и выходы для моделей.

y\_label = ['Модуль упругости при растяжении, ГПа', 'Прочность при растяжении, МПа']

y = data[y\_label]

X = data\_norm.drop(y\_label, axis = 1)

В переменную y\_label передаем вектор из двух целевых переменных: 'Модуль упругости при растяжении, ГПа' и 'Прочность при растяжении, МПа'.

Обозначаем Х и у. у – это исходный датасет по переданному вектору. Х – нормализованный датасет без вектора целевых переменных.

Разделение на обучающую и тестовую выборки. Разделение датасета произведем методом train\_test\_split:

X\_train, X\_test, y\_train,y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

Метод train\_test\_split разделяет массивы или матрицы на случайные тренировочные и тестовые подмножества. В параметры метода передается Х и У, размер тестовой части, random\_state (управляет перемешиванием данных перед применением разделения).

Зададим переменную loss\_df для построения итоговой таблицы с ошибками моделей.

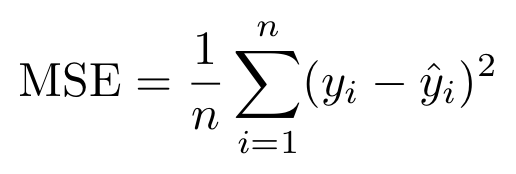
loss\_df = pd.DataFrame(columns=['target', 'model', 'MSE', 'R2'])

Столбцы в таблице:

- target – целевая переменная,

- model – название применяемой модели

- MSE – показатель средне-квадратичной ошибки (Mean Squared Error). Имея правильный ответ y и предсказание ŷ, можно вычислить их разность. Если сложить такие разности для всех ответов, возведённые в квадрат и разделить на количество элементов выборки, получится число, характеризующее качество модели:



- R2 показатель коэффициента детерминации. Коэффициент детерминации  — это доля [дисперсии](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BF%D0%B5%D1%80%D1%81%D0%B8%D1%8F_%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D0%BE%D0%B9_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D1%8B) зависимой переменной, объясняемая рассматриваемой [моделью](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) зависимости, то есть объясняющими переменными. Более точно — это единица минус доля необъяснённой дисперсии (дисперсии случайной ошибки модели, или условной по факторам дисперсии зависимой переменной) в дисперсии зависимой переменной. Его рассматривают как универсальную меру зависимости одной случайной величины от множества других. В частном случае R2 линейной зависимости  является квадратом так называемого [множественного коэффициента корреляции](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D1%8D%D1%84%D1%84%D0%B8%D1%86%D0%B8%D0%B5%D0%BD%D1%82_%D0%BA%D0%BE%D1%80%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%B8) между зависимой переменной и объясняющими переменными. В частности, для модели парной линейной регрессии коэффициент детерминации равен квадрату обычного коэффициента корреляции между y и x.

Истинный коэффициент детерминации модели зависимости случайной величины y от признаков x определяется следующим образом:

R^2 =1-\frac {V(y|x)}{V(y)}=1-\frac {\sigma^2}{\sigma^2_y},

где V(y|x)=\sigma^2 — условная (по признакам x) дисперсия зависимой переменной (дисперсия случайной ошибки модели).

В данном определении используются истинные параметры, характеризующие распределение случайных величин. Если использовать выборочную оценку значений соответствующих дисперсий, то получим формулу для выборочного коэффициента детерминации (который обычно и подразумевается под коэффициентом детерминации):

R^2 =1-\frac {\hat{\sigma}^2}{\hat{\sigma}^2_y}=1-\frac {RSS/n}{TSS/n}=1-\frac {RSS} {TSS},

где

RSS=\sum^n_{t=1}e^2_t=\sum^n_{t=1} (y_t-\hat y_t)^2 — сумма квадратов регрессионных остатков,

TSS=\sum^n_{t=1} (y_t-\bar{y})^2=n \hat \sigma^2_y — общая дисперсия,

y_t,\hat y_t — соответственно, фактические и расчетные значения объясняемой переменной,

\bar{y}=\frac{1}{n}\sum_{i=1}^n y_i  — выборочное вреднее.

В случае [линейной регрессии](http://www.machinelearning.ru/wiki/index.php?title=%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F) с константой TSS=RSS+ESS, где ESS=\sum^n_{t=1} (\hat y_t-\bar{y})^2 — объяснённая сумма квадратов, поэтому получаем более простое определение в этом случае. Коэффициент детерминации — это доля объяснённой дисперсии в общей:

R^2=\frac {ESS} {TSS}.

Необходимо подчеркнуть, что эта формула справедлива только для модели с константой, в общем случае необходимо использовать предыдущую формулу.

Интерпретация:

Коэффициент детерминации для модели с константой принимает значения от 0 до 1. Чем ближе значение коэффициента к 1, тем сильнее зависимость. При оценке регрессионных моделей это интерпретируется как соответствие модели данным. Для приемлемых моделей предполагается, что коэффициент детерминации должен быть хотя бы не меньше 50% (в этом случае коэффициент множественной корреляции превышает по модулю 70%). Модели с коэффициентом детерминации выше 80% можно признать достаточно хорошими (коэффициент корреляции превышает 90%). Равенство коэффициента детерминации единице означает, что объясняемая переменная в точности описывается рассматриваемой моделью.

При отсутствии статистической связи между объясняемой переменной и признаками статистика nR^2 для [линейной регрессии](http://www.machinelearning.ru/wiki/index.php?title=%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F) имеет асимптотическое распределение \chi^2(k-1), где k-1 — число признаков в модели. В случае линейной регрессии с независимыми одинаково распределёнными нормальными случайными ошибками статистика F=\frac {R^2/(k-1)}{(1-R^2)/(n-k)} имеет точное (для выборок любого объёма) [распределение Фишера](http://www.machinelearning.ru/wiki/index.php?title=%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%A4%D0%B8%D1%88%D0%B5%D1%80%D0%B0) F(k-1,n-k). Информация о распределении этих величин позволяет проверить статистическую значимость регрессионной модели исходя из значения коэффициента детерминации. Фактически в этих тестах проверяется гипотеза о равенстве истинного коэффициента детерминации нулю.

## 3.1  Линейная регрессия

Линейная регрессия ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Linear regression) — используемая в [статистике](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0) [регрессионная модель](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) зависимости одной (объясняемой, зависимой) переменной y от другой или нескольких других переменных (факторов, регрессоров, независимых переменных)  x с [линейной функцией](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) зависимости.

Линейная регрессия - это алгоритм машинного обучения с учителем. Иными словами - алгоритму для обучения необходимо указать как входные данные, так и заранее подготовленные выходные данные. Всё месте это называется обучающей выборкой. Линейная регрессия может предсказать только одно значение из бесконечно большого диапазона значений, например температуру, скорость движения спутника, и так далее.

Линейная регрессия может быть двух видов:

- Простая регрессия. Простая линейная регрессия в своей основе содержит простейшую линейную формулу:

y=mx+b

Где m и b это значения, которые необходимо подобрать алгоритму для достижения наилучшей точности в своих предсказаниях. Значение x представляет собой входные данные и y представляет собой выходные данные.

- Множественная регрессия. Более сложная форма линейной регрессии с несколькими переменными. Значения w являются коэффициентами (ещё их называют «весами» - англ. weight), а x1, x2, x3, … входными значениями. В данном случае алгоритм подбирает значения w.

f(x,y,z)=w1x1+w2x2+w3x3

Задаем модель LinearRegression, обучаем модель методом fit() на тренировочных данных. Задаем переменную y\_pred, в которую передаем предсказанные значения методом predict.

Обозначим функцию add\_loss(), с помощью которой передадим в показатели ошибок работы модели в сравнительную таблицу loss\_df.

Построим графики визуализации (Рисунки 13, 14) предсказанных моделью значений с помощью обозначенной ранее функции predicted\_plot().

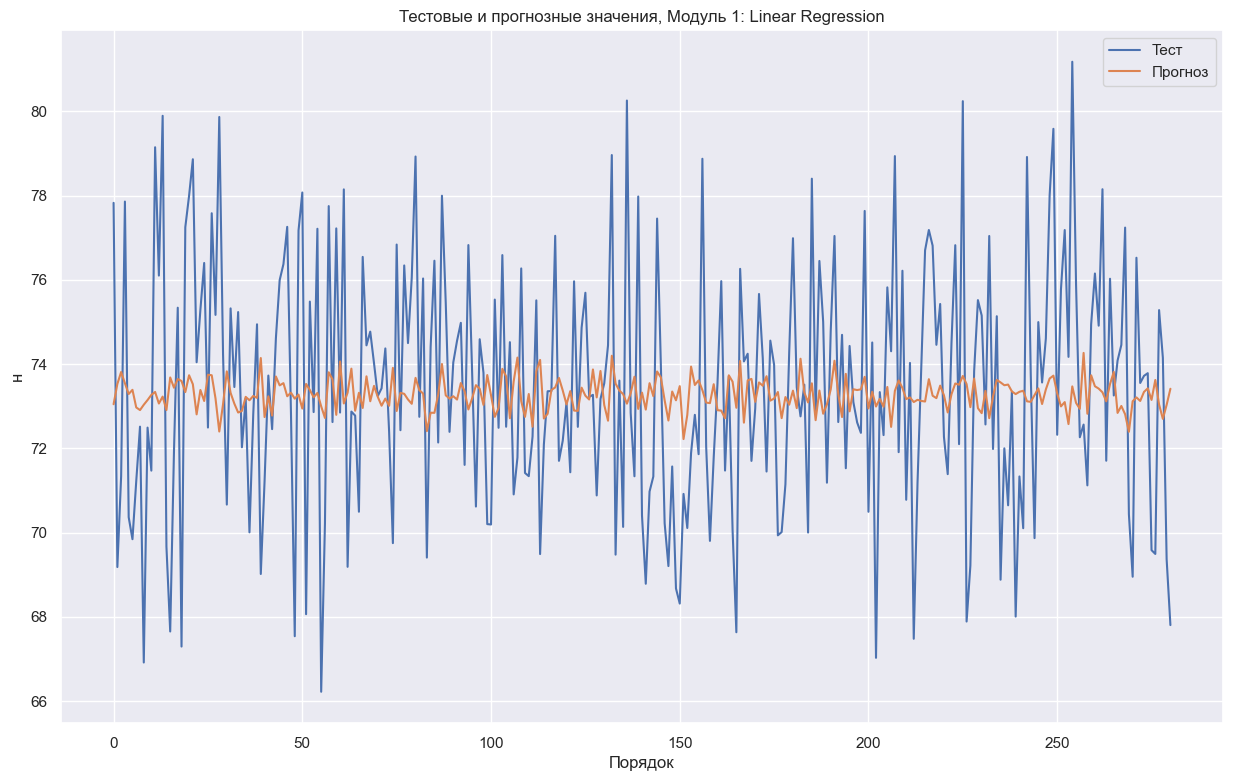


Рисунок 13

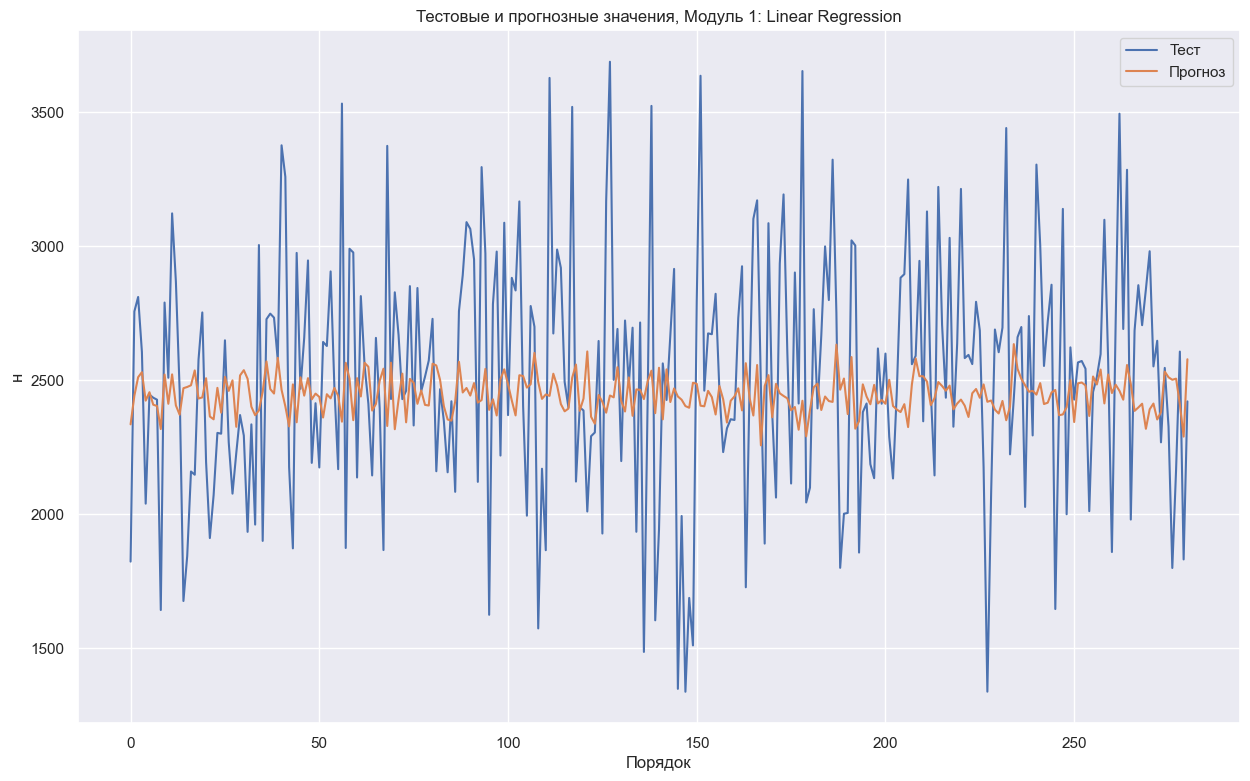


Рисунок 14

Выведем предсказанные значения переменной y\_pred.

По задаче требуется написать приложение, которое будет выдавать прогноз, полученный моделью. Для этого сохраним модель с помощью библиотеки pickle.

Проверяем работу сохраненной модели, данные вывода совпадают.

## 3.2 Регрессия k-ближайших соседей

KNeighborsRegressor – алгоритм регрессии, основанный на методе k-ближайших соседей. Регрессия на основе соседей может использоваться в случаях, когда метки данных являются непрерывными, а не дискретными переменными. Метка, присвоенная точке запроса, вычисляется на основе среднего значения меток ее ближайших соседей.

[KNeighborsRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html#sklearn.neighbors.KNeighborsRegressor) реализует обучение на основе k ближайших соседей каждой точки запроса, где k — целочисленное значение, указанное пользователем.

Базовая регрессия ближайших соседей использует одинаковые веса: то есть каждая точка в локальной окрестности вносит единообразный вклад в классификацию точки запроса. При некоторых обстоятельствах может быть выгодно взвесить точки так, чтобы близлежащие точки вносили больший вклад в регрессию, чем удаленные точки. Это можно реализовать с помощью ключевого слова weights.

Значение по умолчанию weights = ‘uniform’ присваивает всем точкам одинаковые веса. Значение weights = ‘distance’ назначает веса, обратно пропорциональные расстоянию от точки запроса. В качестве альтернативы может быть предоставлена ​​определяемая пользователем функция расстояния, которая будет использоваться для вычисления весов.

Для выполнения задачи передаем в модель [KNeighborsRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html#sklearn.neighbors.KNeighborsRegressor)(). GridSearchCV — функция, входящая в пакет model\_selection библиотеки Scikit-learn. Эта функция помогает перебирать предопределенные гиперпараметры и подгонять оценку (модель) к тренировочному набору. Итак, в итоге мы можем выбрать лучшие параметры из перечисленных гиперпараметров. Поиск по сетке, предоставляемый с помощью [GridSearchCV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html" \l "sklearn.model_selection.GridSearchCV" \t "_blank), исчерпывающе генерирует кандидатов из сетки значений параметров, указанных в  параметре param\_grid.

Чтобы осуществить подбор лучших параметров для модели, зададим переменную grid. В нее помещаем список словарей с именами параметров ( str) в качестве ключей и списков настроек параметров. В этом случае исследуются сетки, охватываемые каждым словарем в списке. Это позволяет выполнять поиск по любой последовательности настроек параметров.

Методом fit() обучим модель. Результаты обучения оцениваем с помощью среднеквадратичной ошибки и коэффициента детерминации. Заданной выше функцией add\_loss передаем результаты в сравнительную таблицу loss\_df.

Визуализируем (Рисунки 15, 16) предсказанные моделью значения с помощью обозначенной ранее функции predicted\_plot().

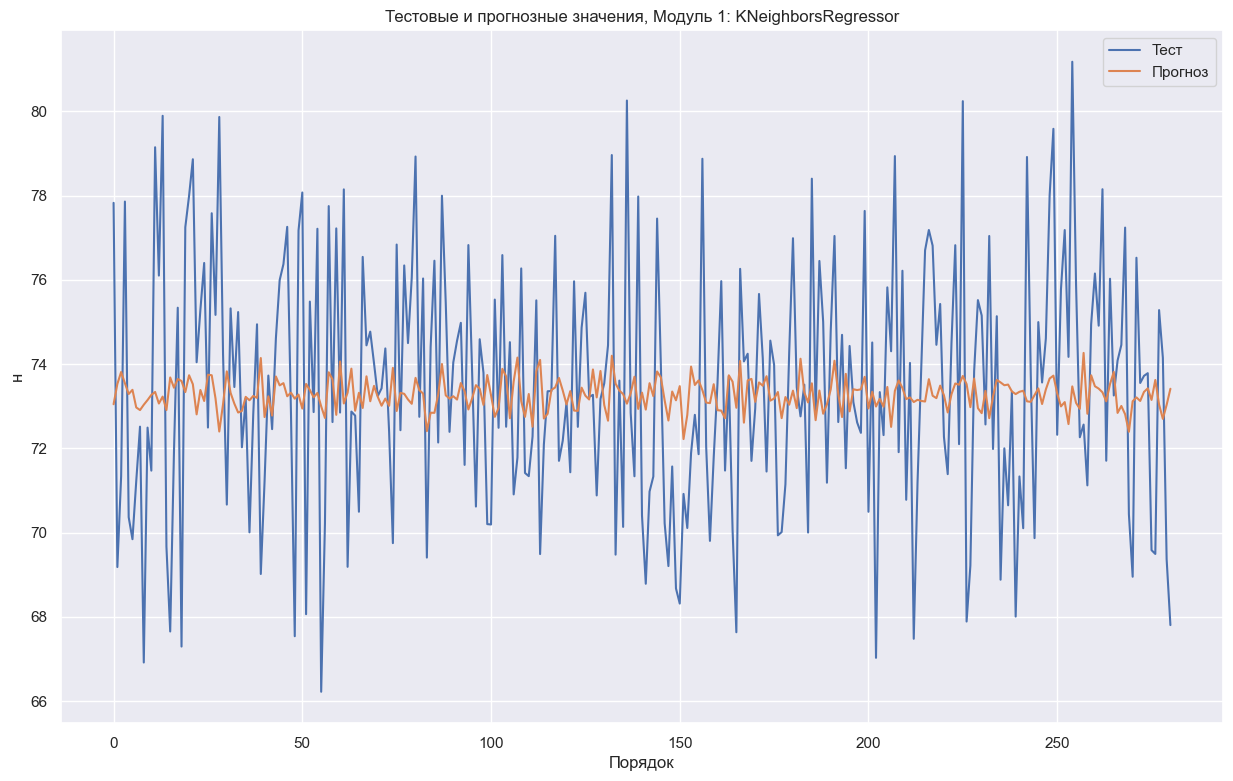


Рисунок 15

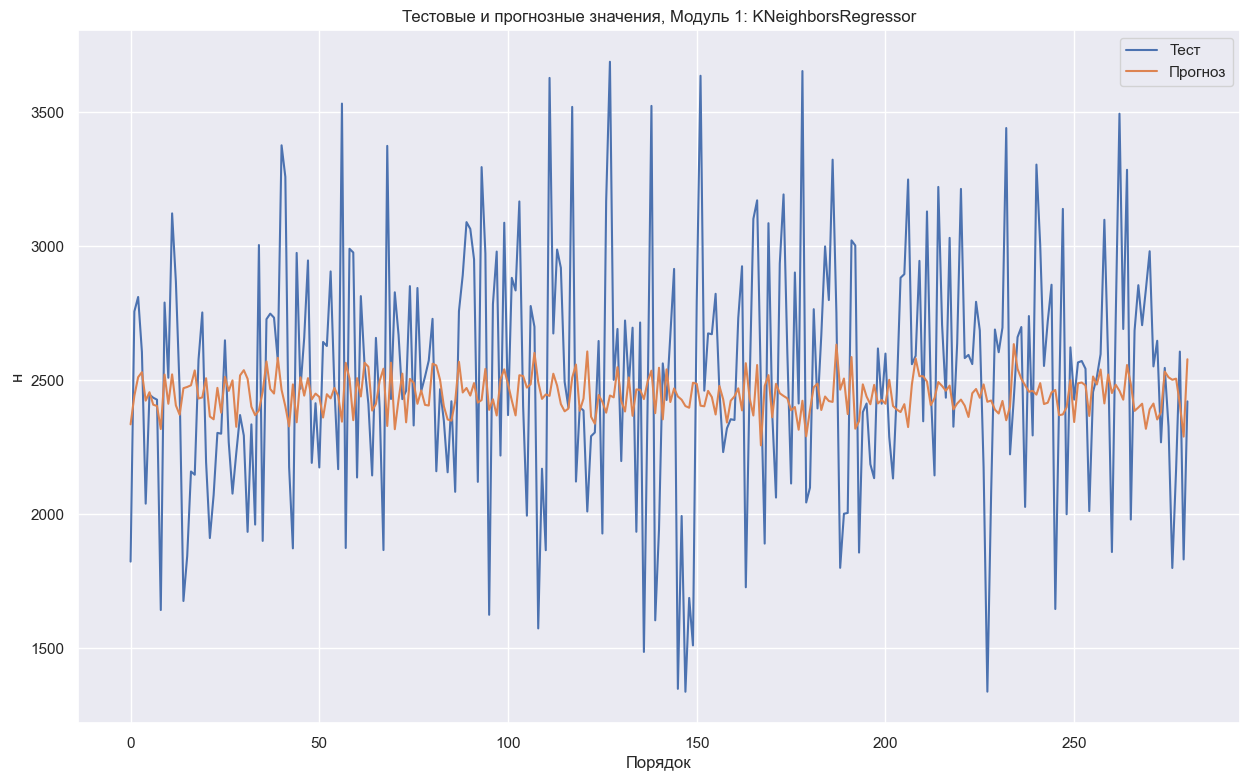


Рисунок 16

Методом gsc.best\_estimator\_ выведем наилучшие параметры, найденные с помощью [GridSearchCV](https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV):

KNeighborsRegressor(n\_neighbors=92)

## 3.3 Метод регрессии опорных векторов SVR

В основе метода регрессии опорных векторов (SVR) лежит поиск гиперплоскости, при которой риск в многомерном пространстве будет минимальным. По сравнению с традиционной регрессионной моделью SVR оценивает коэффициенты путем минимизации квадратичных потерь. Так, если прогнозное значение попадает в область гиперплоскости, то потери равны нулю. В противном случае разности прогнозного и фактического значений.

Метод регрессии опорных векторов SVM не поддерживают мульти-целевую регрессию. Поэтому целесообразно обратиться к модулю MultiOutputRegressor.

Эта стратегия состоит из подбора одного регрессора для каждой целевой переменной. Поскольку каждая цель представлена ​​ровно одним регрессором, можно получить информацию о цели, проверив соответствующий регрессор. Поскольку [MultiOutputRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputRegressor.html" \l "sklearn.multioutput.MultiOutputRegressor" \t "_blank) для каждой цели использует один регрессор, он не может использовать корреляции между целями.

С помощью словаря задаем параметры поиска по сетке GridSearch. Модель – метод регрессии опорных векторов внутри MultiOutputRegressor. Обучаем модель, оцениваем результаты обучения с помощью среднеквадратичной ошибки и коэффициента детерминации. Заданной выше функцией add\_loss передаем результаты в сравнительную таблицу loss\_df.

Визуализируем (Рисунки 17, 18) предсказанные моделью значения с помощью обозначенной ранее функции predicted\_plot().

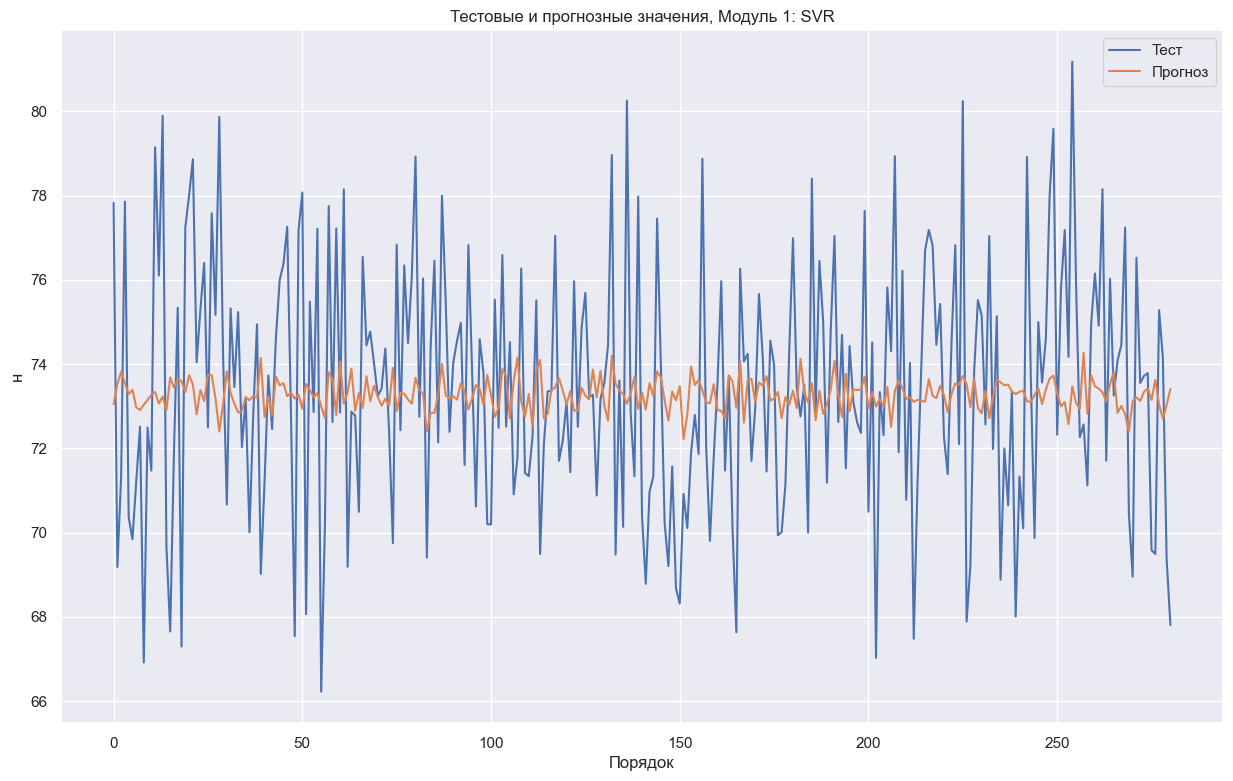


Рисунок 17



Рисунок 18

## 3.4 Random Forest Regressor

Метод случайного леса ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) random forest) — алгоритм [машинного обучения](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), предложенный [Лео Брейманом](https://ru.wikipedia.org/wiki/%D0%91%D1%80%D0%B5%D0%B9%D0%BC%D0%B0%D0%BD,_%D0%9B%D0%B5%D0%BE) и [Адель Катлер](https://en.wikipedia.org/wiki/Adele_Cutler), заключающийся в использовании ансамбля [решающих деревьев](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9). Алгоритм сочетает в себе две основные идеи: метод [бэггинга](https://ru.wikipedia.org/wiki/%D0%91%D1%8D%D0%B3%D0%B3%D0%B8%D0%BD%D0%B3" \o "Бэггинг) Бреймана и [метод случайных подпространств](https://en.wikipedia.org/wiki/Random_subspace_method), предложенный [Тин Кам Хо](https://en.wikipedia.org/wiki/Tin_Kam_Ho). Алгоритм применяется для задач классификации, регрессии и кластеризации. Основная идея заключается в использовании большого [ансамбля](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D1%81%D0%B0%D0%BC%D0%B1%D0%BB%D1%8C_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%B2_(%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BC%D0%B0%D1%88%D0%B8%D0%BD)) [решающих деревьев](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F_%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9), каждое из которых само по себе даёт очень невысокое качество классификации, но за счёт их большого количества результат получается хорошим.

Основными параметрами для настройки при использовании метода являются n\_estimators и max\_features. Первое — это количество деревьев в лесу. Чем больше, тем лучше, но и больше времени потребуется для вычислений. Кроме того, результаты перестанут значительно улучшаться за пределами критического числа деревьев. Последний представляет собой размер случайных подмножеств признаков, которые следует учитывать при разделении узла. Чем ниже, тем больше уменьшение дисперсии, но также и больше увеличение систематической ошибки. Эмпирические хорошие значения по умолчанию равны max\_features=1.0 или эквивалентны max\_features=None(всегда учитываются все функции вместо случайного подмножества) для задач регрессии и max\_features="sqrt"(с использованием случайного подмножества размера sqrt(n\_features)) для задач классификации (где n\_features количество функций в данных). Значение по умолчанию max\_features=1.0 эквивалентно бэггинг деревьям, и большей рандомности можно добиться, установив меньшие значения (например, 0,3 является типичным значением по умолчанию в литературе). Хорошие результаты часто достигаются при настройке max\_depth=None в сочетании с  min\_samples\_split=2 (т.е. при полной разработке деревьев). Однако, эти значения обычно не оптимальны и могут привести к тому, что модели будут потреблять много оперативной памяти.  Кроме того, в случайных лесах по умолчанию используются выборки начальной загрузки ( bootstrap=True), в то время как стратегия по умолчанию для дополнительных деревьев заключается в использовании всего набора данных ( bootstrap=False). При использовании бутстрап-выборки ошибка обобщения может быть оценена на неучтенных или неучтенных выборках. Это можно включить, установив oob\_score=True.

Задаем параметры grid для подстройки модели. Оптимизируем работу регрессора с помощью добавления MultiOutputRegressor, ищем лучшие параметры модели с помощью GridSearch. Обучаем модель, оцениваем результаты обучения с помощью среднеквадратичной ошибки и коэффициента детерминации. Заданной выше функцией add\_loss передаем результаты в сравнительную таблицу loss\_df. Визуализируем (Рисунки 19, 20) предсказанные моделью значения с помощью обозначенной ранее функции predicted\_plot().

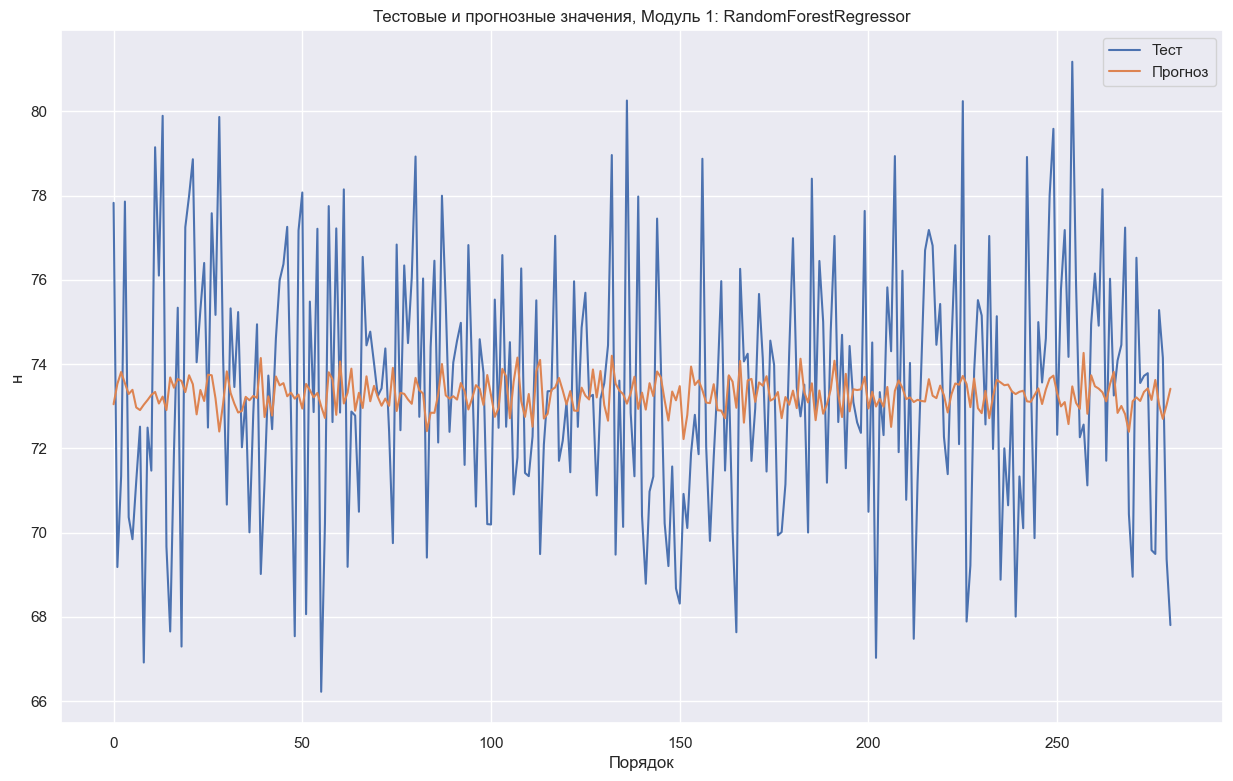


Рисунок 19

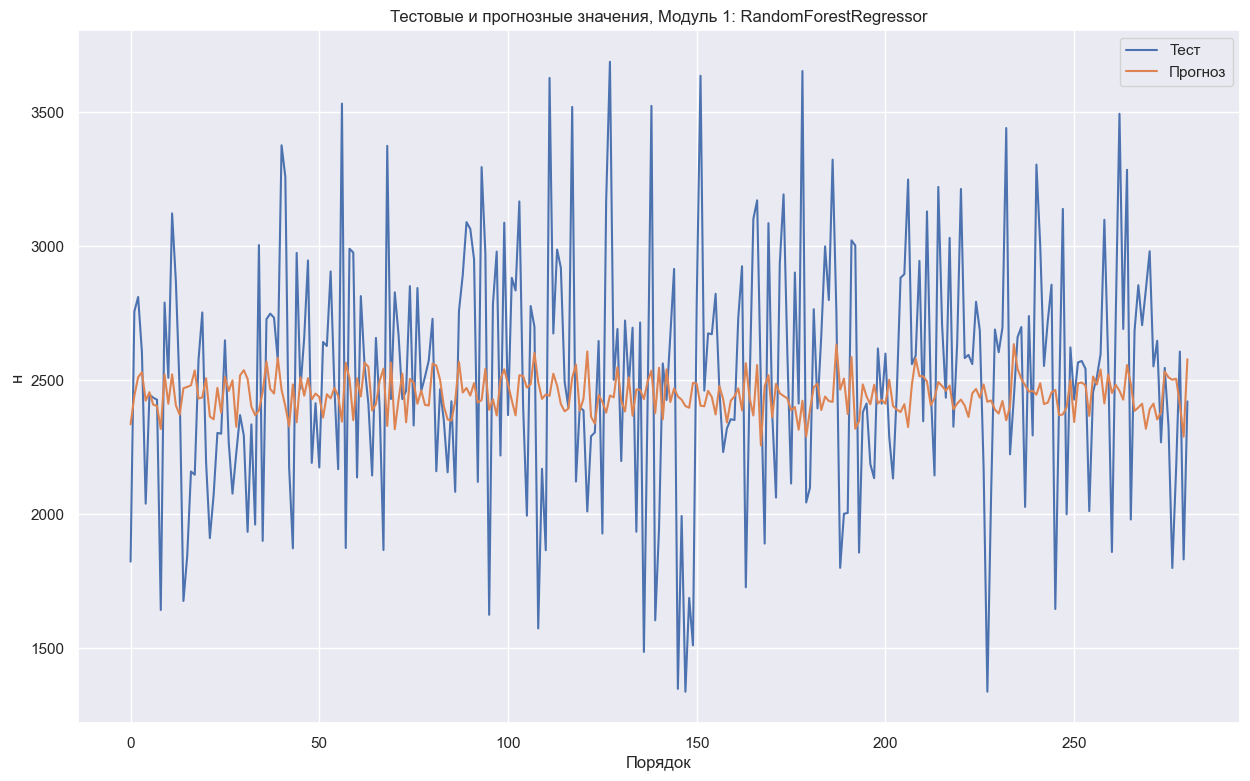


Рисунок 20

Выведем общую таблицу (Рисунок 21) сравнения ошибок моделей loss\_df:

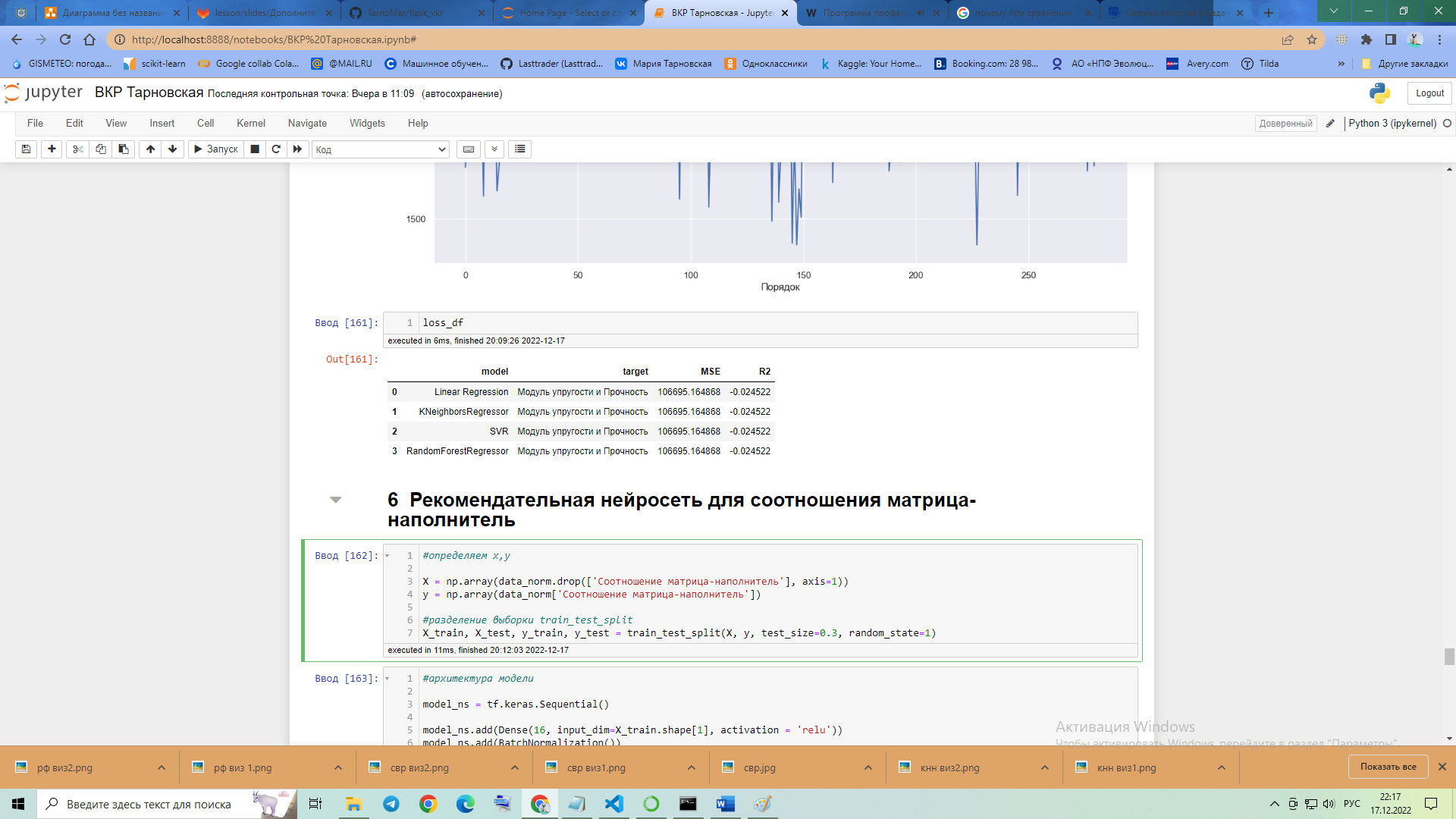


Рисунок 22

Из таблицы ошибок видно, что ни одна модель не дала удовлетворительных результатов.

# **4 Рекомендательная нейросеть для соотношения матрица-наполнитель**

Для построения полносвязной нейросети переопределим Х и у в соответствии с целевой переменной «Соотношение матрица-наполнитель». Разделим выборку на тренировочные и тестовые данные в соотношении 70/30 методом train\_test\_split.

В архитектуре ИНС используется модель Sequential. Она представляет собой линейный стек слоев. Модель состоит из 6 слоев.

На вход используем полносвязный слой Dense, количество нейронов = 16, активационная функция «relu».

Далее следуют четыре скрытых слоя – BatchNormalization, полносвязный Dense, слой Dropout (как метод регуляризации ИНС, предназначен для уменьшения переобучения сети за счет предотвращения сложных коадаптаций отдельных нейронов на тренировочных данных во время обучения) и еще один полносвязный слой Dense.

На выходе полносвязный слой Dense с одним нейроном, активационная функция «sigmoid».

Компилируем модель. Используем оптимизатор Adam. Оптимизатор Адам — это метод стохастического градиентного спуска, основанный на адаптивной оценке моментов первого и второго порядка.

Согласно [Kingma et al., 2014](http://arxiv.org/abs/1412.6980) , этот метод « эффективен в вычислительном отношении, требует мало памяти, инвариантен к диагональному масштабированию градиентов и хорошо подходит для задач, которые являются большими с точки зрения данных/параметров ».

Функция потерь – MAE, средняя абсолютная ошибка. Метрика – также, МАЕ.

Запускаем обучение модели (Рисунок 23). Путем эмпирического подбора параметров, наилучший результат показало использование 20 эпох.

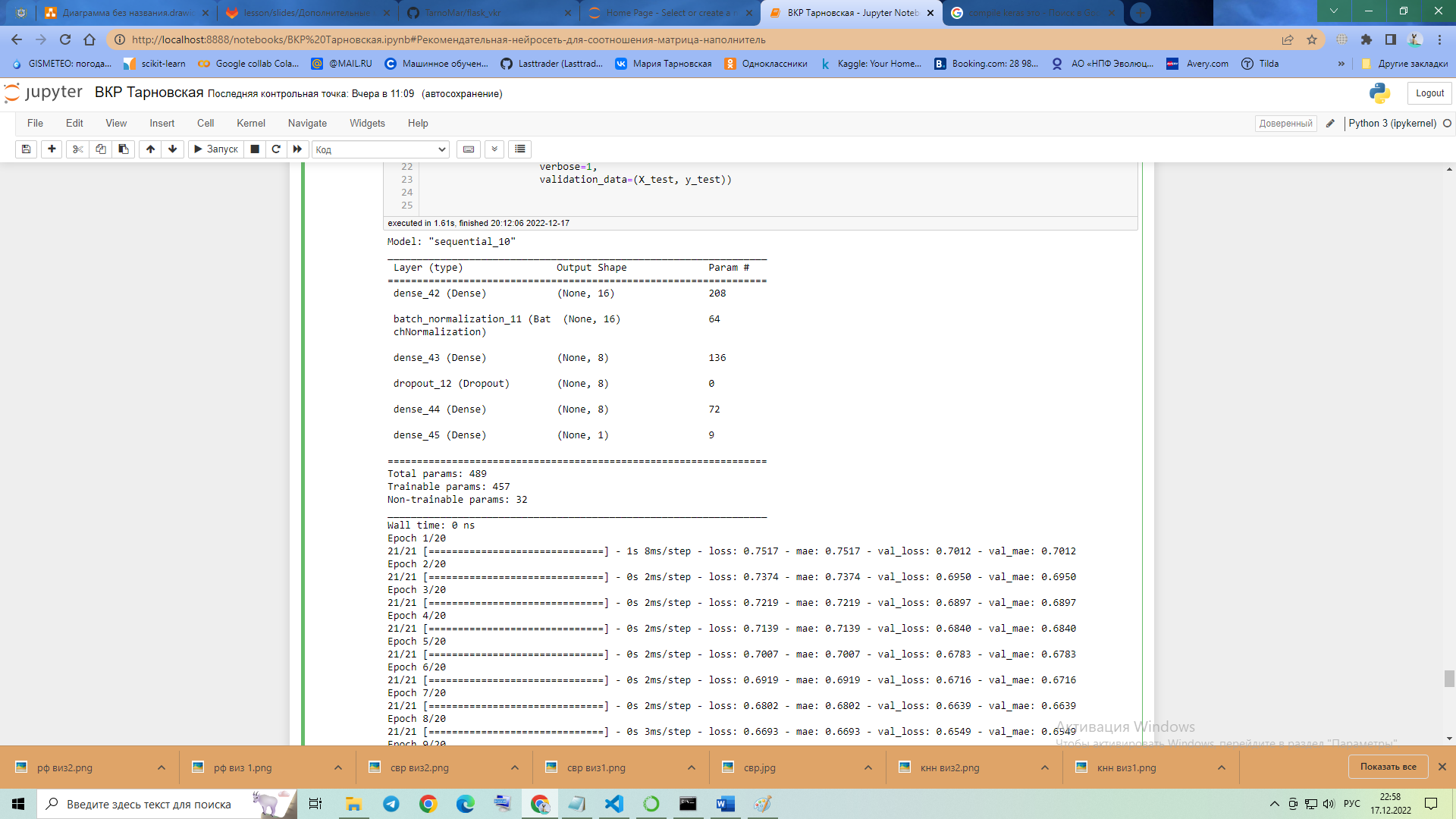


Рисунок 23

Визуализируем процесс обучения нейросети (Рисунок 24).

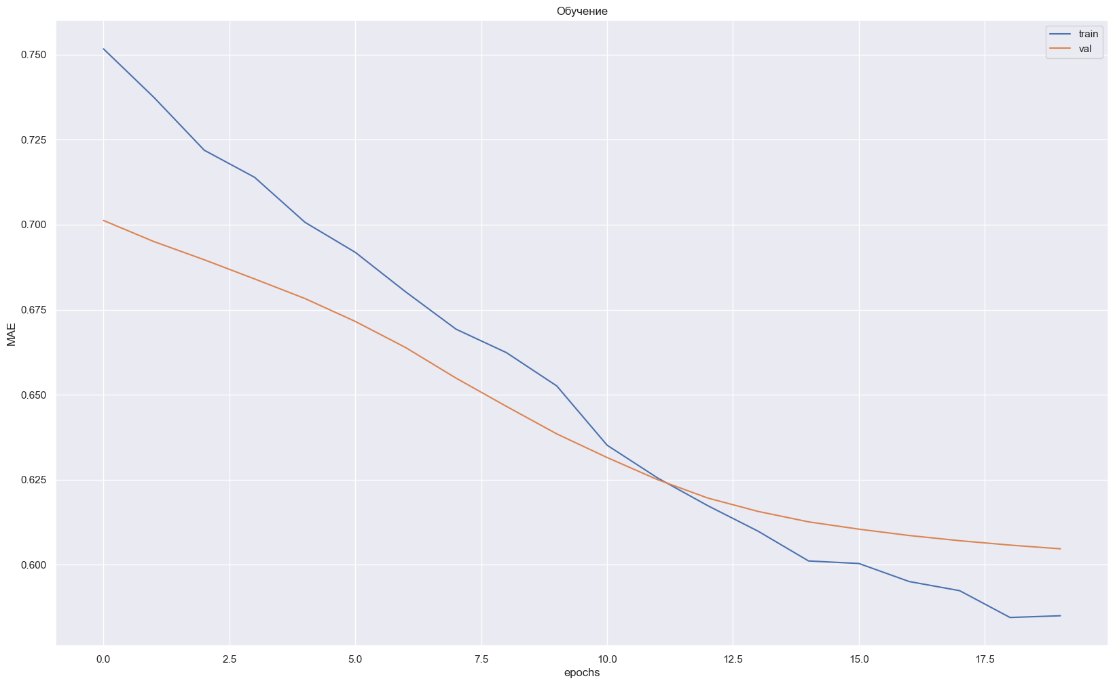


Рисунок 24

Выводим метрики ошибок MSE и R2, добавляем с помощью функции add\_loss() в сравнительную таблицу (Рисунок 25).

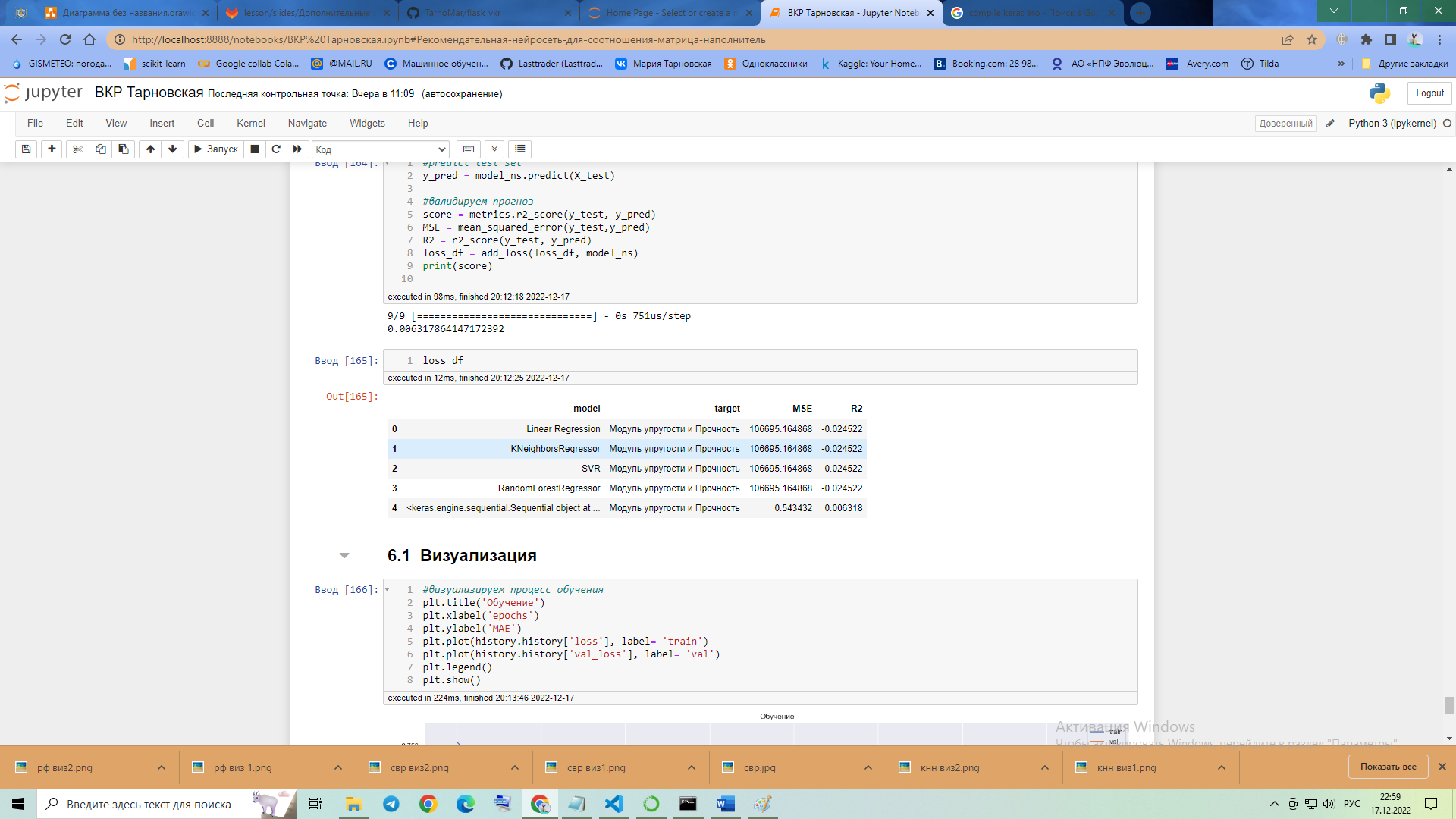


Рисунок 25

Наилучший результат из использованных моделей показала нейросеть.

# **5 Создание Flask приложения**

Создание приложение осуществлялось в среде разработки Visual Studio Code. В терминале активируем виртуальную среду conda activate vkr.

C:\Users\all>python -m venv fproject\venv

C:\Users\all>cd fproject

C:\Users\all\fproject>venv\scripts\activate

(venv) C:\Users\all\fproject>pip install flask

pip install scikit-learn

Заполняем структуру модели FLASK-VKR, добавляем файл сохраненной модели lr-model.pkl.

Далее создаем файл app.py. Импортируем необходимые библиотеки. Обозначаем, что приложение Flask, прописываем путь с методами Post и Get, задаем функцию main, которая описывает методы и открывает модель для прогноза.

Создаем файл гипертекстовой разметки страницы index.html (Рисунок 26).

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Расчет по линейной модели</title>

</head>

<body>

{% if message %}

<p>**{{** message **}}**</p>

{% endif %}

<form action="" method="post">

<p>

<label for="x1">Соотношение матрица-наполнитель</label>

<input type="text" name="x1">

</p>

<p>

<label for="x2">Плотность, кг/м3</label>

<input type="text" name="x2">

</p>

<p>

<label for="x3">модуль упругости, ГПа</label>

<input type="text" name="x3">

</p>

<p>

<label for="x4">Количество отвердителя, м.%</label>

<input type="text" name="x3">

</p>

<p>

<label for="x5">Содержание эпоксидных групп,%\_2</label>

<input type="text" name="x3">

</p>

<p>

<label for="x6">Температура вспышки, С\_2</label>

<input type="text" name="x3">

</p>

<p>

<label for="x7">Поверхностная плотность, г/м2</label>

<input type="text" name="x3">

</p>

<p>

<label for="x8">Потребление смолы, г/м2</label>

<input type="text" name="x3">

</p>

<p>

<label for="x9">Угол нашивки, град</label>

<input type="text" name="x3">

</p>

<p>

<label for="x10">Шаг нашивки</label>

<input type="text" name="x3">

</p>

<p>

<label for="x11">Плотность нашивки</label>

<input type="text" name="x3">

</p>

<p>

<input type="submit">

</p>

{% if message2 %}

<p>**{{** message2 **}}**</p>

{% endif %}

</form>

</body>

</html>

Рисунок 26

В котором указываются параметры текста, поле ввода, размер шрифта, пробелы.

В терминале с помощью команды python app.py запускаем приложение. Получаем адрес <http://127.0.0.1:5000/>, по которому можно зайти в приложение локально (Рисунок 27).

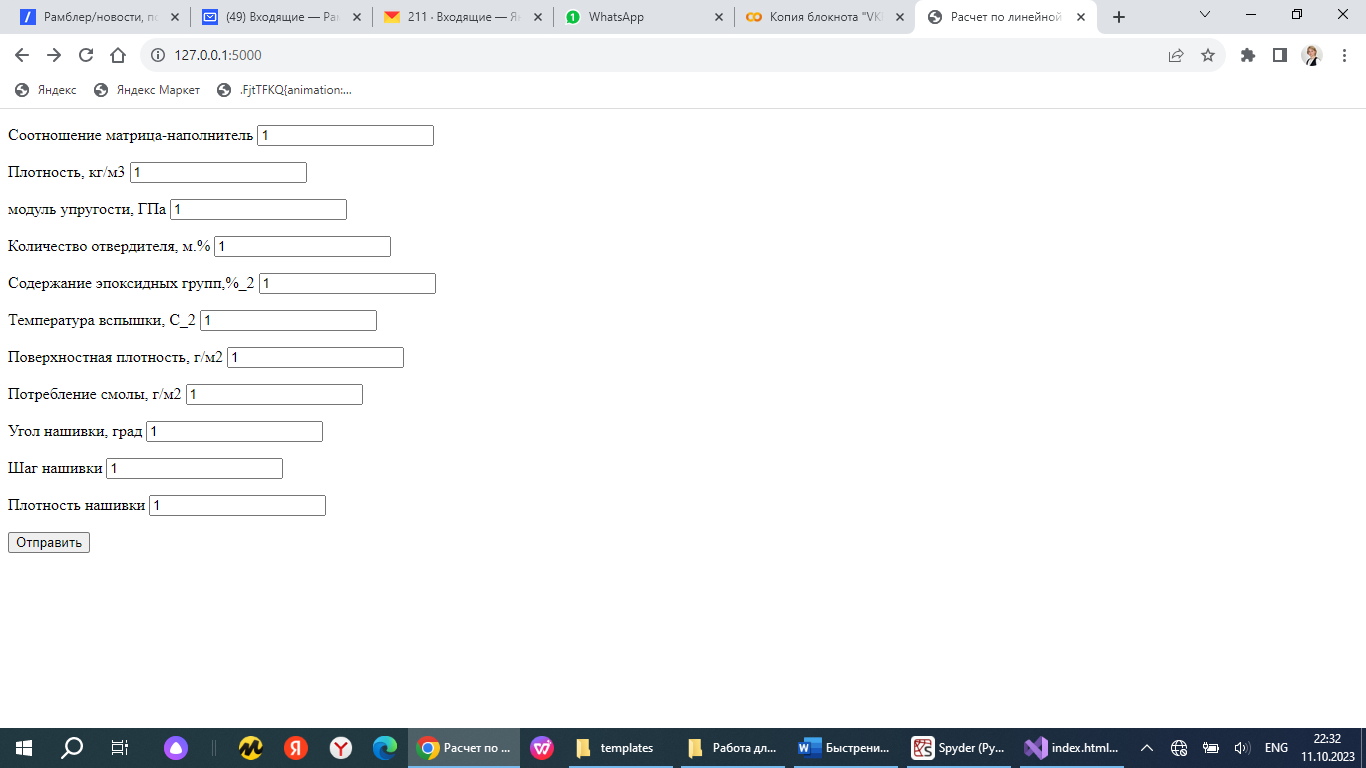


Рисунок 27

# **Заключение**

В результате анализа датасета с помощью моделей машинного обучения и нейросети удовлетворительного результата получено не было. У моделей слишком высокие показатели ошибок.

Набор данных мал. Возможно, есть смысл получить больший датасет. Большее количество данных позволит моделям лучше обучиться.

Другой вариант решения поставленной задачи – взаимодействие с представителями и экспертами индустрии. Для получения консультаций по зависимостям признаков, коррелирующих с целевыми переменными.

# **Список использованной литературы**

1. Deep Learning with Keras via Artificial Neural Network [Электронный ресурс]- Режим доступа: https://rpubs.com/A\_Rodionoff/Regression-Keras(Дата обращения: 20.09.2023).
2. Jackson L Разница между классификацией и регрессией: [Электронный ресурс] – Режим доступа: https://ru.strephonsays.com/classification-and-vs-regression-13803 (дата обращения: 10.09.2023).
3. Алексеев, Д. С. Технологии интеллектуального анализа данных : учебник для вузов / Д. С. Алексеев, О. В. Щекочихин. — Санкт-Петербург : Лань, 2022. — 176 с. — ISBN 978-5-8114-8299-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/187559
4. Бринк Х. Машинное обучение / Х. Бринк, Дж. Ричардс, М. Феверолф. — пер. с англ. Рузмайкина И. — Санкт-Петербург: Питер, 2017. — 336 с.
5. Демидова, Л. А. Интеллектуальный анализ данных на языке Python : учебно-методическое пособие / Л. А. Демидова. — Москва : РТУ МИРЭА, 2021. — 92 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/218693
6. Землянский, А.А. Теория электронной обработки информации: монография / А.А. Землянский. – М.: РГАУ-МСХА им. К. А. Тимирязева, 2012. - 151 с.
7. Кудрявцев, Н. Г. Практика применения компьютерного зрения и элементов машинного обучения в учебных проектах : учебное пособие / Н. Г. Кудрявцев, И. Н. Фролов. — Горно-Алтайск : ГАГУ, 2022. — 180 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/271100
8. Макшанов, А. В. Технологии интеллектуального анализа данных : учебное пособие / А. В. Макшанов, А. Е. Журавлев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2022. — 212 с. — ISBN 978-5-8114-4493-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/206711
9. Машинное обучение на практике с Python и Keras [Электронный ресурс]. – Режим доступа: https://pythonru.com/primery/mashinnoe-obucheniena-praktike-s-python-i-keras (дата обращения 08.10.2023).
10. Остроух, А. В. Системы искусственного интеллекта : монография / А. В. Остроух, Н. Е. Суркова. — 2-е изд., стер. — Санкт-Петербург : Лань, 2021. — 228 с. — ISBN 978-5-8114-8519-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/176662>
11. П. Брюс, Э. Брюс. 1. Разведочный анализ данных // Практическая статистика для специалистов Data Science. — СПб.: БХВ-Петербург, 2018. — С. 19—58. — 304 с.
12. Программные системы статистического анализа. Обнаружение закономерностей в данных с использованием системы R и языка Python : учебное пособие / В. М. Волкова, М. А. Семёнова, Е. С. Четвертакова, С. С. Вожов. — Новосибирск : НГТУ, 2017. — 74 с. — ISBN 978-5-7782-3183-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/118287 (дата обращения: 08.10.2022). — Режим доступа: для авториз. Пользователей.,
13. Федоров, Д. Ю. Программирование на языке высокого уровня Python : учебное пособие для вузов / Д. Ю. Федоров. — 4-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2022. — 214 с. — (Высшее образование). — ISBN 978-5-534-15733-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/509562