МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э.
Баумана
(национальный исследовательский университет)»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по курсу

«Data Science»

Слушатель: Дряев Феликс Аланович

Содержание

ВВЕДЕНИЕ

Композиционные материалы, представляют собой металлические или неметаллические матрицы (основы) с заданным распределением в них упрочнителей (волокон дисперсных частиц и др.); при этом эффективно используются индивидуальные свойства составляющих композиции. По характеру структуры композиционные материалы подразделяются на волокнистые, упрочненные непрерывными волокнами и нитевидными кристаллами, дисперстно упрочненнные материалы, полученные путем введения в металлическую матрицу дисперсных частиц упрочнителей, слоистые материалы, созданные путем прессования или прокатки разнородных материалов. К композиционным материалам также относятся сплавы с направленной кристаллизацией эвтектических структур. Комбинируя объемное содержание, можно, в зависимости от назначения, получать материалы с требуемыми значениями прочности, жаропрочности, модуля упругости, абразивной стойкости, а также создавать композиции с необходимыми магнитными, диэлектрическими, радиопоглощающими и другими специальными свойствами.

Волокнистые композиционные материалы, армированные нитевидными кристаллами и непрерывными волокнами тугоплавких соединений и элементов (SiC, Al2O3, бор, углерод и др.), являются новым классом материалов. Однако принципы армирования для упрочнения известны в технике с глубокой древности. Еще в Вавилоне использовали тростник для армирования глины при постройке жилищ, а в Древней Греции железными прутьями укрепляли мраморные колонны при постройке дворцов и храмов. В 1555-1560 гг. при постройке храма Василия Блаженного в Москве русские зодчие Барма и Постник использовали армированные железными полосами каменные плиты. Прообразом композиционные материалы являются широко известный железобетон, представляющий собой сочетание бетона,

работающего на сжатие и стальной арматуры, работающей на растяжение, а также полученные в XIX веке прокаткой слоистые материалы.

Успешному развитию современным КМ содействовали: разработка и применение в конструкциях волокнистых стеклопластиков, обладающих высокой удельной прочностью (1940-1950 гг.). Открытие весьма высокой прочности, приближающейся к теоретической, нитевидных кристаллов и доказательства возможности использования их для упрочнения металлических и неметаллических материалов (1950-1960 гг.), разработка новых армирующих материалов – высокопрочных и высокомодульных непрерывных волокон бора, углерода, Al2O3, SiC и волокон других неорганических тугоплавких соединений, а также упрочнителей на основе металлов (1960-1970 гг.).

Важнейшими технологическими методами изготовления композиционных материалов являются: пропитка армирующих волокон матричным материалом; формирование в пресс-форме лент упрочнителя и матрицы, получаемых намоткой; холодное прессование обоих компонентов с последующим спеканием; электрохимическое нанесение покрытий на волокна с последующим прессованием; осаждение матрицы плазменным напылением на упрочнитель с последующим обжатием; пакетная диффузионная сварка монослойных лент компонентов; совместная прокатка армирующих элементов с матрицей и др. Весьма перспективны композиционные материалы, армированные нитевидными кристаллами (усами) керамических, полимерных и др. материалов. Размеры усов обычно составляют от долей до нескольких мкм. По диаметру и примерно 10-15мм. по длине.

1. Аналитическая часть

1.1 Постановка задачи

Для дипломной работы были даны 2 файла:

1. X_bp.xlsx (состоящий из 1022 строки и 11 столбцов) и содержащий следующие данные: соотношение матрица-наполнитель, плотность (кг/м3), модуль упругости (Гпа), количество отвердителя (м.%), содержание эпоксидных групп (%_2), температура вспышки (С_2, поверхностная плотность (г/м2), модуль упругости при растяжении (Гпа), прочность при растяжении (МПа), потребление смолы (г/м2);

2. X_nup.xlsx (состоящий из 1039 строки и 4 столбцов) и содержащий следующие данные: угол нашивки (град), шаг нашивки, плотность нашивки.

Шаг 1:

Импортируем нам необходимые библиотеки (рисунок 1):

In [1]:
```
#Импортируем нам необходимые библиотеки!
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
import seaborn as sns
import plotly.express as px
import tensorflow as tf
import sklearn
from sklearn import linear_model
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression, LogisticRegression, SGDRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_percentage_error, mean_absolute_error
from sklearn.model_selection import train_test_split, GridSearchCV, KFold, cross_val_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn import preprocessing
from sklearn.preprocessing import Normalizer, LabelEncoder, MinMaxScaler, StandardScaler
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from tensorflow import keras as keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten, Dropout, BatchNormalization, Activation
from pandas import read_excel, DataFrame, Series
from keras.wrappers.scikit_learn import KerasClassifier, KerasRegressor
from tensorflow.keras.models import Sequential
from numpy.random import seed
from scipy import stats
import warnings
warnings.filterwarnings("ignore")
```

Рисунок 1 – необходимые для работы библиотеки

Шаг 2: Загружаем два файла эксель и удаляем ненужные столбцы (рисунок 2, 3):

```
In [2]: #Загружаем первый файл эксель
        df_bp = pd.read_excel("C:/Users/fdrya/anaconda3/X_bp.xlsx")
        df_bp.shape
Out[2]: (1023, 11)

In [3]: #Удаляем первый столбец из первого файла
        df_bp.drop(['Unnamed: 0'], axis=1, inplace=True)
        #Посмотрим на первые 5 строк первого файла
        df_bp.head()
```

Out[3]:

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

```
In [4]: # Размерность первого файла
        df_bp.shape
Out[4]: (1023, 10)
```

Рисунок 2 – загрузка и первые действия с файлом X_bp

```
In [5]: # Загружаем второй файл
        df_nup = pd.read_excel("C:/Users/fdrya/anaconda3/X_nup.xlsx")
        df_nup.shape
Out[5]: (1040, 4)

In [6]: #Удаляем первый столбец
        df_nup.drop(['Unnamed: 0'], axis=1, inplace=True)
        #Посмотрим на первые 5 строк второго файла
        df_nup.head()
```

Out[6]:

	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	0	4.0	57.0
1	0	4.0	60.0
2	0	4.0	70.0
3	0	5.0	47.0
4	0	5.0	57.0

```
In [7]: # Проверим размерность второго файла
        df_nup.shape
Out[7]: (1040, 3)
```

Рисунок 3 – загрузка и первые действия с файлом X_nup

Шаг 3: Собираем файлы (датасеты) в один набор данных (рисунок 4)

```
In [8]: # Два датасета имеют разный объем строк.
        # Надо собрать исходные данные файлы в единый набор данных.
        # Объединяем их по типу INNER.
        df = df_bp.merge(df_nup, left_index = True, right_index = True, how = 'inner')
        df.head().T
```

Out[8]:

	0	1	2	3	4
Соотношение матрица-наполнитель	1.857143	1.857143	1.857143	1.857143	2.771331
Плотность, кг/м3	2030.000000	2030.000000	2030.000000	2030.000000	2030.000000
модуль упругости, ГПа	738.736842	738.736842	738.736842	738.736842	753.000000
Количество отвердителя, м.%	30.000000	50.000000	49.900000	129.000000	111.860000
Содержание эпоксидных групп,%_2	22.267857	23.750000	33.000000	21.250000	22.267857
Температура вспышки, С_2	100.000000	284.615385	284.615385	300.000000	284.615385
Поверхностная плотность, г/м2	210.000000	210.000000	210.000000	210.000000	210.000000
Модуль упругости при растяжении, ГПа	70.000000	70.000000	70.000000	70.000000	70.000000
Прочность при растяжении, МПа	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
Потребление смолы, г/м2	220.000000	220.000000	220.000000	220.000000	220.000000
Угол нашивки, град	0.000000	0.000000	0.000000	0.000000	0.000000
Шаг нашивки	4.000000	4.000000	4.000000	5.000000	5.000000
Плотность нашивки	57.000000	60.000000	70.000000	47.000000	57.000000

Рисунок 4 – объединение двух датасетов в один

Шаг 4: Так как ак как кол-во уникальных значений в колонке Угол нашивки равно 2, можем привести данные в этой колонке к значениям 0 и 1 (рисунок 5)

```
In [12]: # Поработаем со столбцом "Угол нашивки"

In [13]: df['Угол нашивки, град'].nunique()
Out[13]: 2

In [14]: #Проверим кол-во элементов где равен 0 угол
         df['Угол нашивки, град'][df['Угол нашивки, град'] == 0.0].count()
Out[14]: 520

In [15]: # Приведем Угол нашивки к 0 и 1 и integer
         df = df.replace({'Угол нашивки, град': {0.0 : 0, 90.0 : 1}})
         df['Угол нашивки, град'] = df['Угол нашивки, град'].astype(int)

In [16]: #Переименуем столбец
         df = df.rename(columns={'Угол нашивки, град' : 'Угол нашивки'})
         df
```

Out[16]:

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки
0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000	210.000000	70.000000	3000.000000	220.000000	0
1	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385	210.000000	70.000000	3000.000000	220.000000	0
2	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385	210.000000	70.000000	3000.000000	220.000000	0

Рисунок 5 – объединение двух датасетов в один

Шаг 5: Проверим где пропущенные данные (рисунок 6)

```
In [26]: # Проверим где пропущенные данные
         df.isnull().sum()
         # Пропущенных данных нет и нулевых значений нет
         # Очистку проводить не будем
```

```
Out[26]: Соотношение матрица-наполнитель          0
         Плотность, кг/м3                         0
         модуль упругости, ГПа                    0
         Количество отвердителя, м.%              0
         Содержание эпоксидных групп,%_2          0
         Температура вспышки, С_2                 0
         Поверхностная плотность, г/м2            0
         Модуль упругости при растяжении, ГПа     0
         Прочность при растяжении, МПа            0
         Потребление смолы, г/м2                  0
         Угол нашивки                             0
         Шаг нашивки                              0
         Плотность нашивки                        0
         dtype: int64
```

Рисунок 6 – Пропущенные данные

Шаг 6: Построить первый график с выбросами по-нашему датасету (рисунок 7,8).

```
# "Диаграмы" - первый вариант
scaler = MinMaxScaler()
scaler.fit(df)
plt.figure(figsize = (15, 15))
plt.suptitle('Диаграмма', y = 0.9 ,
             fontsize = 30)
plt.boxplot(pd.DataFrame(scaler.transform(df)),
            labels = df.columns,patch_artist = True,
            meanline = True, vert = False,
            boxprops = dict(facecolor = 'r', color = 'b'),
            medianprops = dict(color = 'lime'),
            whiskerprops = dict(color="g"),
            capprops = dict(color = "black"),
            flierprops = dict(color = "y", markeredgecolor = "maroon"))
plt.show()
```

Рисунок 7 – Код для создания диаграммы

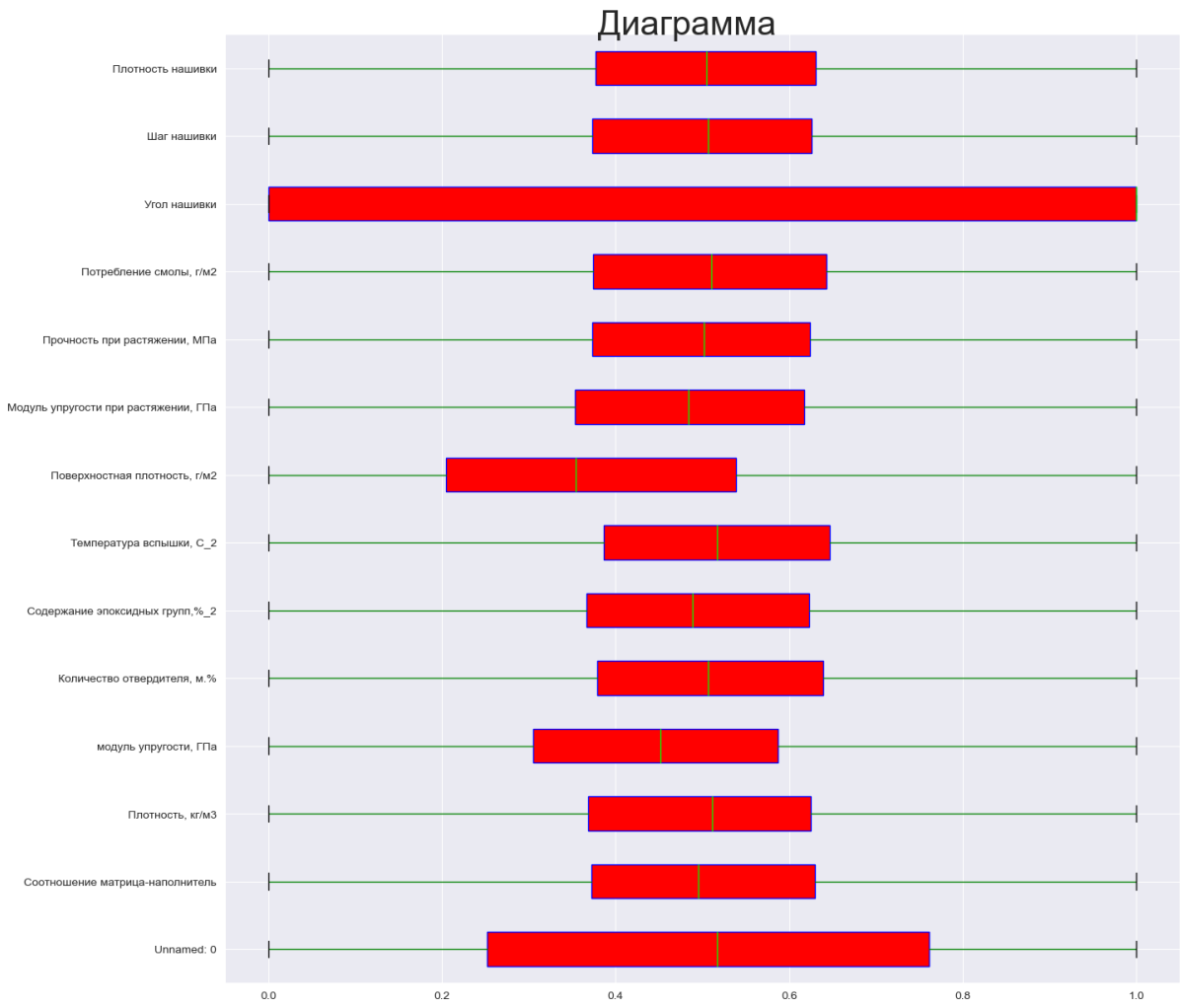


Рисунок 8 –Диаграмма с выбросами

1.2 Описание используемых методов

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

- метод опорных векторов;
- случайный лес;
- линейная регрессия;
- градиентный бустинг;
- К-ближайших соседей;

- дерево решений;
- стохастический градиентный спуск;
- многослойный перцептрон;
- Лассо;

Метод опорных векторов (Support Vector Regression) (рисунок 9) – этот бинарный линейный классификатор был выбран, потому что он хорошо работает на небольших датасетах. Данный алгоритм – это алгоритм обучения с учителем, использующихся для задач классификации и регрессионного анализа, это контролируемое обучение моделей с использование схожих алгоритмов для анализа данных и распознавания шаблонов. Учитывая обучающую выборку, где алгоритм помечает каждый объект, как принадлежащий к одной из двух категорий, строит модель, которая определяет новые наблюдения в одну из категорий.

Модель метода опорных векторов – отображение данных точками в пространстве, так что между наблюдениями отдельных категорий имеется разрыв, и он максимален.

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

Достоинства метода: для классификации достаточно небольшого набора данных. При правильной работе модели, построенной на тестовом множестве, вполне возможно применение данного метода на реальных данных. Эффективен при большом количестве гиперпараметров. Способен обрабатывать случаи, когда гиперпараметров больше, чем количество наблюдений. Существует возможность гибко настраивать разделяющую функцию. Алгоритм максимизирует разделяющую полосу, которая, как подушка безопасности, позволяет уменьшить количество ошибок классификации.

Недостатки метода: неустойчивость к шуму, поэтому в работе была проведена тщательнейшая работа с выбросами, иначе в обучающих данных шумы становятся опорными объектами-нарушителями и напрямую влияют на построение разделяющей гиперплоскости; для больших наборов данных требуется долгое время обучения; достаточно сложно подбирать полезные преобразования данных; параметры модели сложно интерпретировать, поэтому были рассмотрены и другие методы.

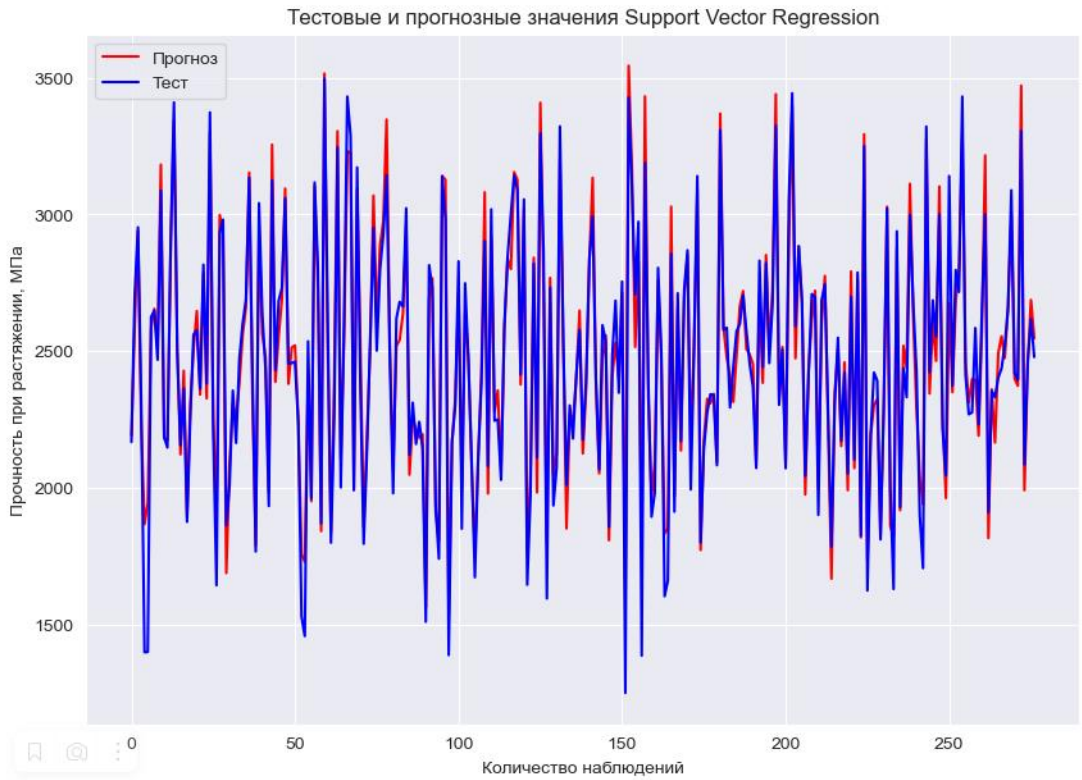


Рисунок 9 - график метода опорных векторов для прочности при растяжении, МПа

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

Достоинства метода: не переобучается; не требует предобработки входных данных; эффективно обрабатывает пропущенные данные, данные

с большим числом классов и признаков; имеет высокую точность предсказания и внутреннюю оценку обобщающей способности модели, а также высокую параллелизуемость и масштабируемость (рисунок 10).

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

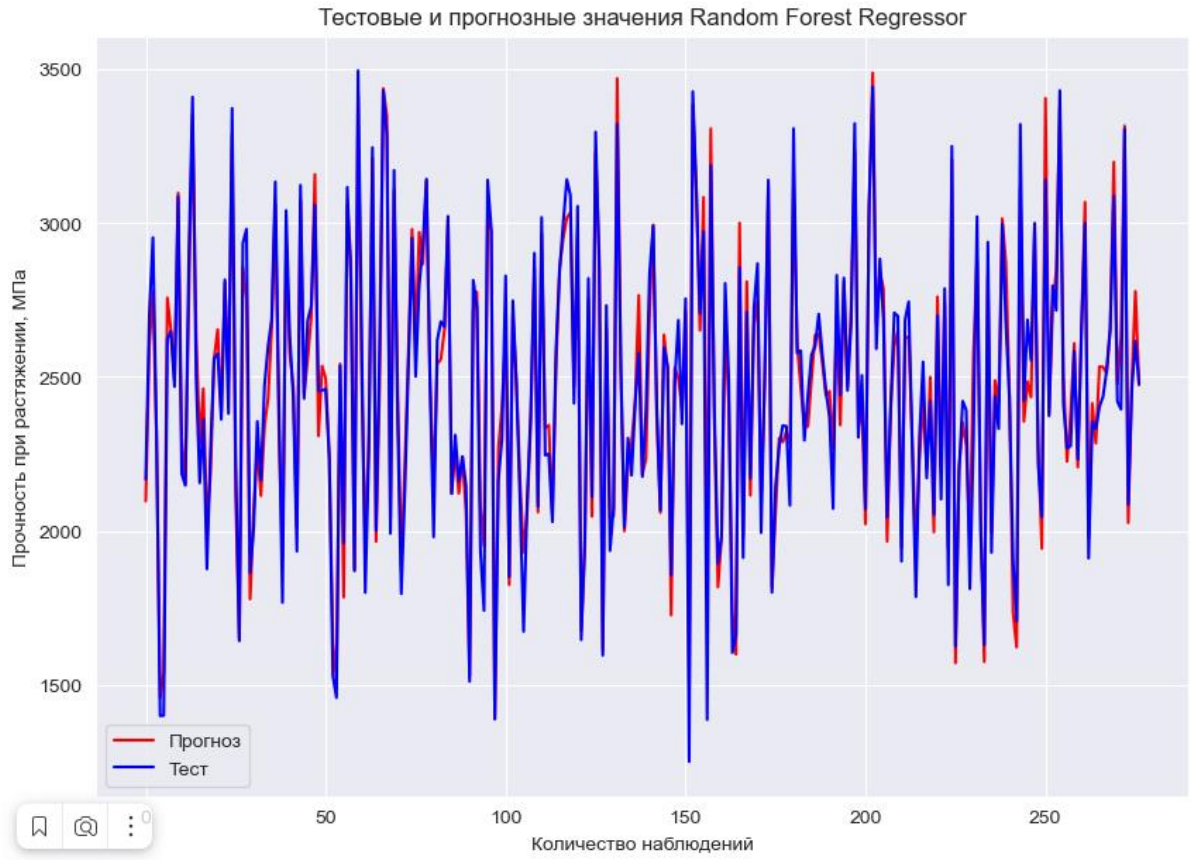


Рисунок 10 - график "случайного леса" для прочности при растяжении, МПа

Линейная регрессия (Linear regression) (рисунок 11) — это алгоритм машинного обучения, основанный на контролируемом обучении, рассматривающий зависимость между одной входной и выходными переменными. Это один из самых простых и эффективных инструментов статистического моделирования. Она определяет зависимость переменных

с помощью линии наилучшего соответствия. Модель регрессии создаёт несколько метрик. R^2 , или коэффициент детерминации, позволяет измерить, насколько модель может объяснить дисперсию данных. Если R-квадрат равен 1, это значит, что модель описывает все данные. Если же R-квадрат равен 0,5, модель объясняет лишь 50 процентов дисперсии данных. Оставшиеся отклонения не имеют объяснения. Чем ближе R^2 к единице, тем лучше.

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

Недостатки метода: моделирует только прямые линейные зависимости; требует прямую связь между зависимыми и независимыми переменными; выбросы оказывают огромное влияние, а границы линейны.

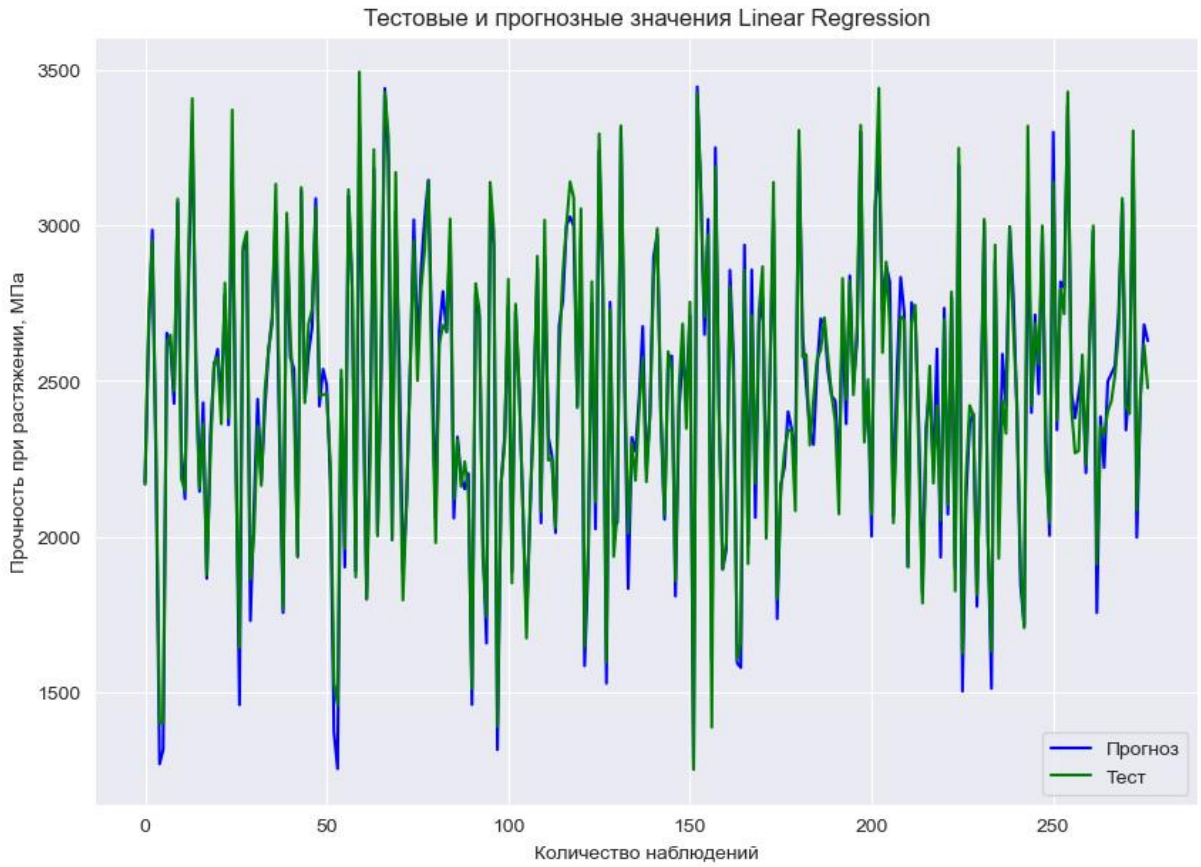


Рисунок 11 - график линейной регрессии для прочности при растяжении, МПа

Градиентный бустинг (Gradient Boosting) (рисунок 12) — это ансамбль деревьев решений, обученный с использованием градиентного бустинга. В основе данного алгоритма лежит итеративное обучение деревьев решений с целью минимизировать функцию потерь. Основная идея градиентного бустинга: строятся последовательно несколько базовых классификаторов, каждый из которых как можно лучше компенсирует недостатки предыдущих. Финальный классификатор является линейной композицией этих базовых классификаторов.

Достоинства метода: новые алгоритмы учатся на ошибках предыдущих; требуется меньше итераций, чтобы приблизиться к фактическим прогнозам; наблюдения выбираются на основе ошибки; прост в настройке темпа обучения и применения; легко интерпретируем.

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

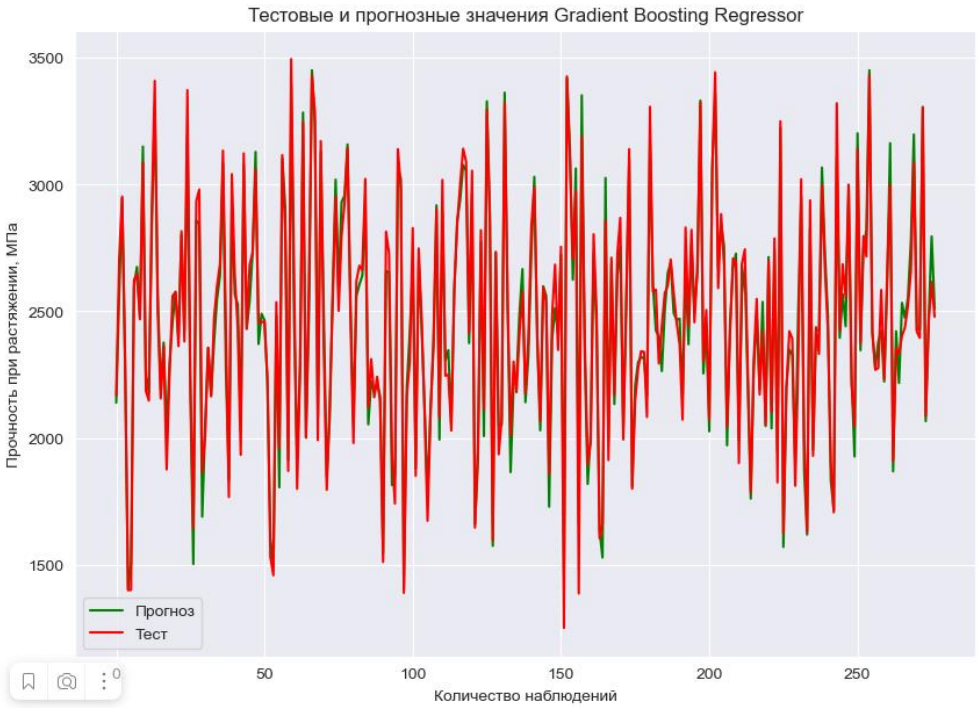


Рисунок 12 - график градиентного бустинга для прочности при растяжении, МПа

Метод ближайших соседей - К-ближайших соседей (kNN - k Nearest Neighbours) (рисунок 13) ищет ближайшие объекты с известными значения целевой переменной и основывается на хранении данных в памяти для сравнения с новыми элементами. Алгоритм находит расстояния между запросом и всеми примерами в данных, выбирая определенное количество примеров (k), наиболее близких к запросу, затем голосует за наиболее часто встречающуюся метку (в случае задачи классификации) или усредняет метки (в случае задачи регрессии).

Достоинства метода: прост в реализации и понимании полученных результатов; имеет низкую чувствительность к выбросам; не требует построения модели; допускает настройку нескольких параметров; позволяет делать дополнительные допущения; универсален; находит лучшее решение из возможных; решает задачи небольшой размерности.

Недостатки метода: замедляется с ростом объёма данных; не создаёт правил; не обобщает предыдущий опыт; основывается на всем массиве

доступных исторических данных; невозможно сказать, на каком основании строятся ответы; сложно выбрать близость метрики; имеет высокую зависимость результатов классификации от выбранной метрики; полностью перебирает всю обучающую выборку при распознавании; имеет вычислительную трудоёмкость.

Рисунок 13 - график К-ближайших соседей для прочности при растяжении, Мпа

Дерево принятия решений (DecisionTreeRegressor) (рисунок 14) – метод автоматического анализа больших массивов данных. Это инструмент принятия решений, в котором используется древовидная структура, подобная блок-схеме, или модель решений и всех их возможных результатов, включая результаты, затраты и полезность. Дерево принятия решений - эффективный инструмент интеллектуального анализа данных и предсказательной аналитики. Алгоритм дерева решений подпадает под категорию контролируемых алгоритмов обучения. Он работает как для непрерывных, так и для категориальных выходных переменных. Правила генерируются за счёт обобщения множества отдельных наблюдений (обучающих примеров), описывающих предметную область. Регрессия дерева решений отслеживает особенности объекта и обучает модель в структуре дерева прогнозированию данных в будущем для получения значимого непрерывного вывода. Дерево решений один из вариантов

решения регрессионной задачи, в случае если зависимость в данных не имеет очевидной корреляции.

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

Недостатки метода: ошибаются при классификации с большим количеством классов и небольшой обучающей выборкой; имеют нестабильный процесс (изменение в одном узле может привести к построению совсем другого дерева); имеет затратные вычисления; необходимо обращать внимание на размер; ограниченное число вариантов решения проблемы.

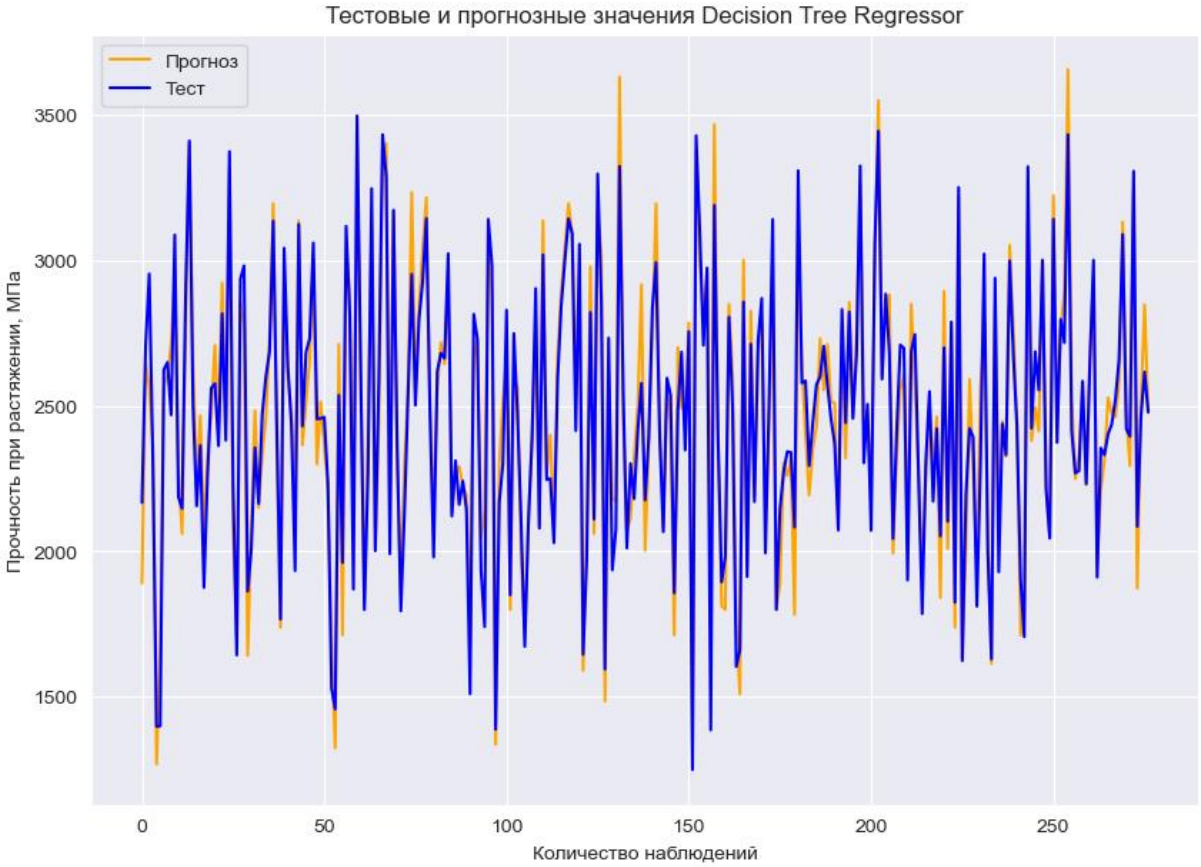


Рисунок 14 - график дерева принятия решений для прочности при растяжении

Стохастический градиентный спуск (SGDRegressor) (рисунок 15) — это простой, но очень эффективный подход к подгонке линейных классификаторов и регрессоров под выпуклые функции потерь. Этот подход подразумевает корректировку весов нейронной сети, используя аппроксимацию градиента функционала, вычисленную только на одном случайном обучающем примере из выборки.

Достоинства метода: эффективен; прост в реализации; имеет множество возможностей для настройки кода; способен обучаться на избыточно больших выборках.

Недостатки метода: требует ряд гиперпараметров; чувствителен к масштабированию функций; может не сходиться или сходиться слишком медленно; функционал многоэкстремален; процесс может "застрять" в одном из локальных минимумов; возможно переобучение.

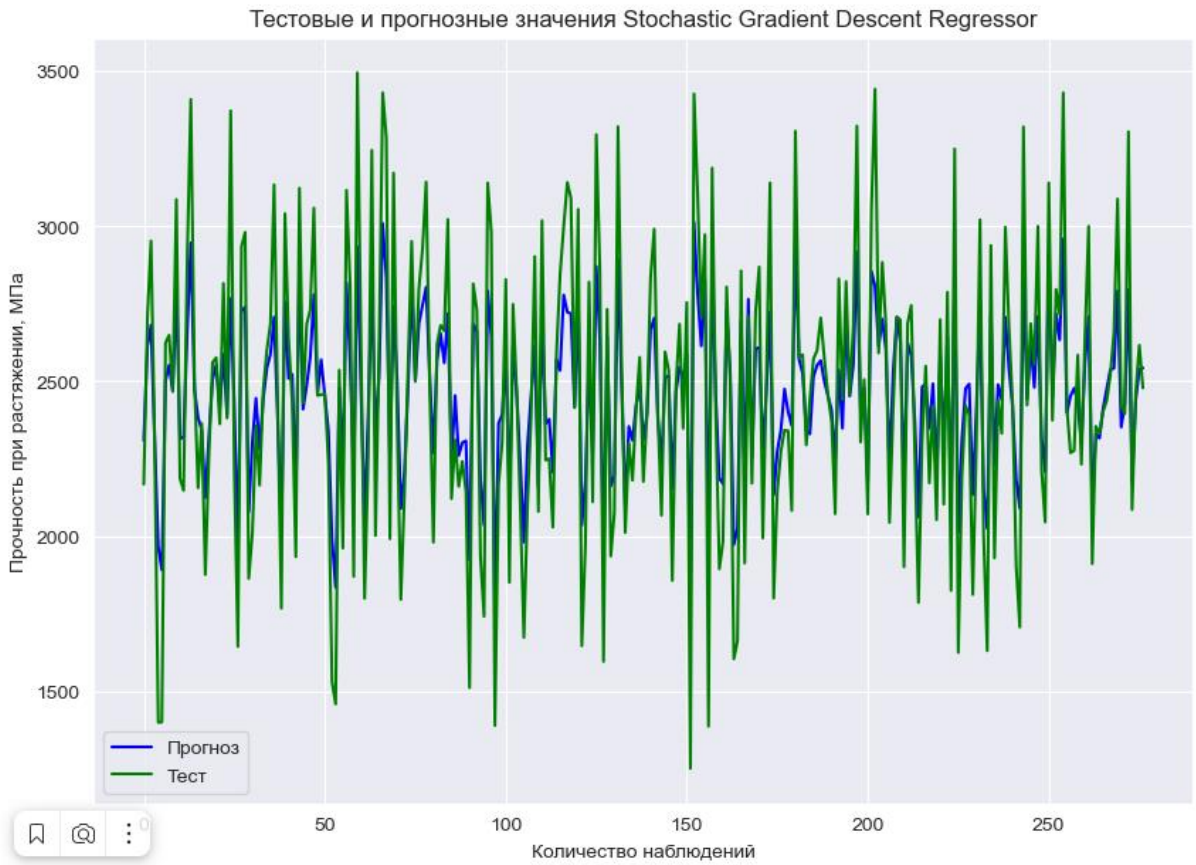


Рисунок 15 – график стохастического градиентного спуска для модуля упругости

Многослойный персептрон (MLPRegressor) (рисунок 16) — это алгоритм обучения с учителем, который изучает функцию f(·):Rm→Ro обучением на наборе данных, где m — количество измерений для ввода и o- количество размеров для вывода. Это искусственная нейронная сеть, имеющая 3 или более слоёв персептронов. Эти слои - один входной слой, 1 или более скрытых слоёв и один выходной слой персептронов.

Достоинства метода: построение сложных разделяющих поверхностей; возможность осуществления любого отображения входных векторов в выходные; легко обобщает входные данные; не требует распределения входных векторов; изучает нелинейные модели.

Недостатки метода: имеет невыпуклую функцию потерь; разные инициализации случайных весов могут привести к разной точности

проверки; требует настройки ряда гиперпараметров; чувствителен к масштабированию функций.

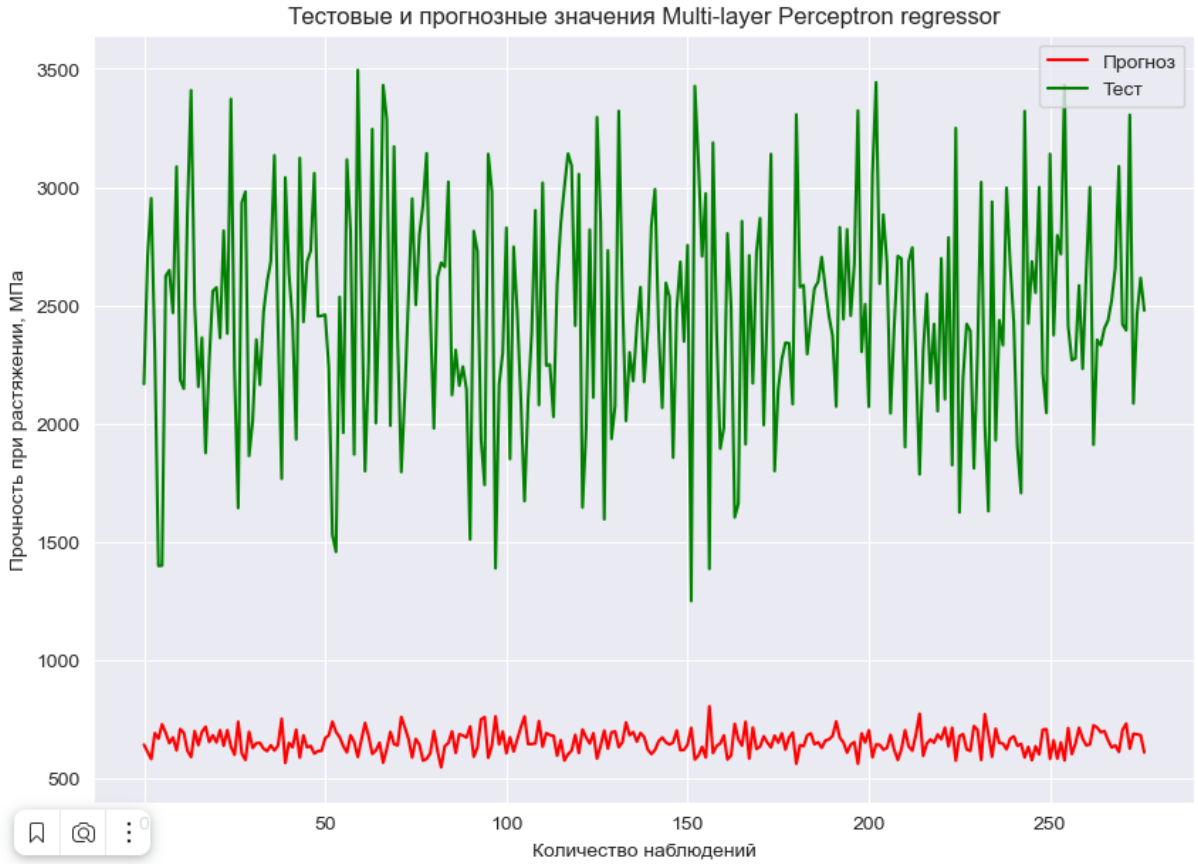


Рисунок 16 – график многослойного персептрона для модуля упругости, ГПа

Лассо регрессия (Lasso) (рисунок 17) — это линейная модель, которая оценивает разреженные коэффициенты. Это простой метод, позволяющий уменьшить сложность модели и предотвратить переопределение, которое может возникнуть в результате простой линейной регрессии. Данный метод вводит дополнительное слагаемое регуляризации в оптимизацию модели. Это даёт более устойчивое решение. В регрессии лассо добавляется условие смещения в функцию оптимизации для того, чтобы уменьшить коллинеарность и, следовательно, дисперсию модели. Но вместо квадратичного смещения, используется смещение абсолютного значения.

Лассо регрессия хорошо прогнозирует модели временных рядов на основе регрессии, таким как авторегрессии.

Достоинства метода: легко полностью избавляется от шумов в данных; быстро работает; не очень энергоёмко; способно полностью убрать признак из датасета; доступно обнуляет значения коэффициентов.

Недостатки метода: выбор модели не помогает и обычно вредит; часто страдает качество прогнозирования; выдаёт ложное срабатывание результата; случайным образом выбирает одну из коллинеарных переменных; не оценивает правильность формы взаимосвязи между независимой и зависимой переменными; не всегда лучше, чем пошаговая регрессия.

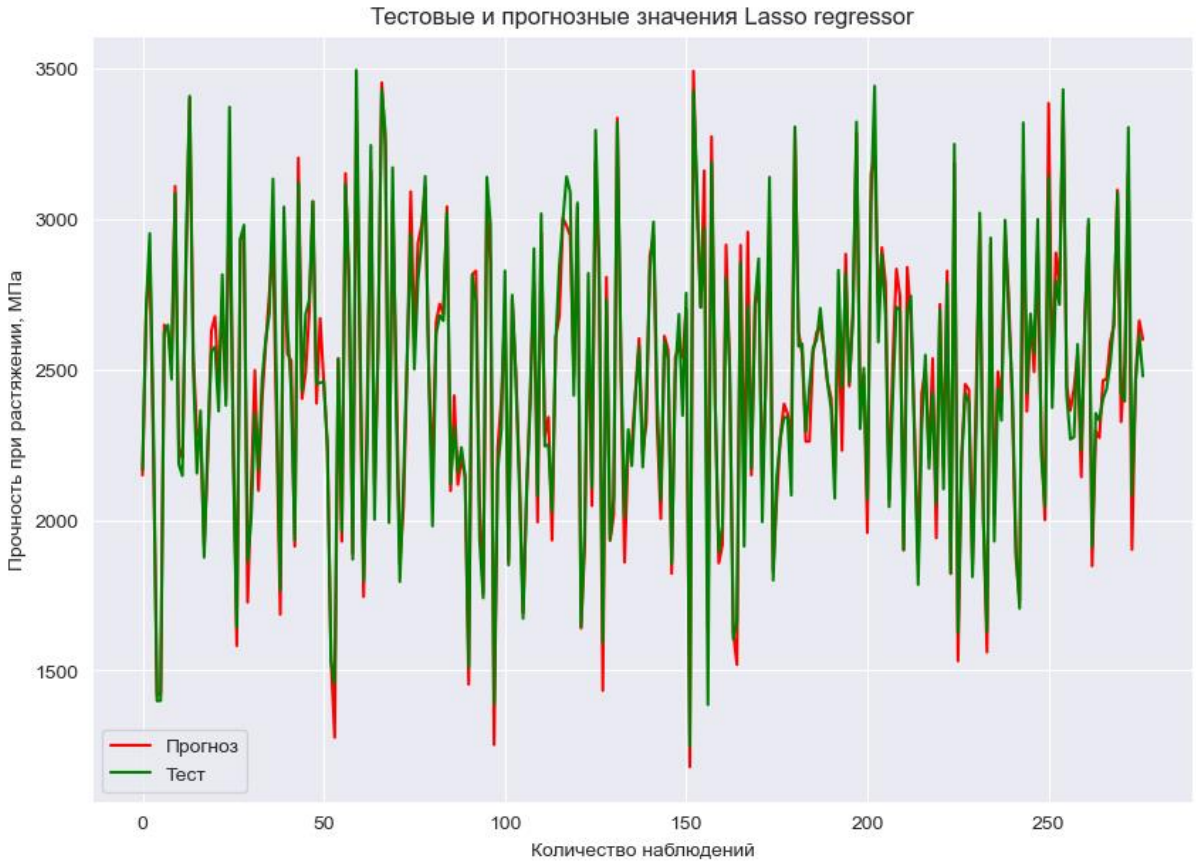


Рисунок 17 - график метода Лассо для модуля упругости, Гпа

Немного расскажем об используемых метриках качества моделей: R2 или коэффициент детерминации измеряет долю дисперсии, объяснённую моделью, в общей дисперсии целевой переменной (рисунок 18).

```
#Выводим гиперпараметры для оптимальной модели
print(grid.best_estimator_)
knr_upr = grid.best_estimator_
print(f'R2-score RFR для прочности при растяжении, МПа: {knr_upr.score(x_test_1, y_test_1).round(3)}')
```

```
RandomForestRegressor(criterion='mse', max_depth=15, n_estimators=200,
                      random_state=33)
R2-score RFR для прочности при растяжении, МПа: 0.963
```

Рисунок 18 – часть кода для результата метрики R2 для метода «Случайный лес»

Если он близок к единице, то модель хорошо объясняет данные, если же он близок к нулю, то качество прогноза идентично средней величине целевой переменной (т.е. очень низкое). Отрицательные значение коэффициента детерминации означают плохую объясняющую способность модели.

MSE (Mean Squared Error) или средняя квадратичная ошибка принимает значениях в тех же единицах, что и целевая переменная. Чем ближе к нулю MSE, тем лучше работают предсказательные качества модели (рисунок 19).

```
In [200]: #подставим оптимальные гиперпараметры в нашу модель метода к ближайших соседей
          knn_grid = KNeighborsRegressor(algorithm = 'brute', n_neighbors = 7, weights = 'distance')
          #Обучаем модель
          knn_grid.fit(x_train_1, y_train_1)
          predictions_knn_grid = knn_grid.predict(x_test_1)
          #Оцениваем точность на тестовом наборе
          mae_knn_grid = mean_absolute_error(predictions_knn_grid, y_test_1)
          mae_knn_grid
Out[200]: 99.2816938003547
```

Рисунок 19 - код для вывода различных метрик для метода опорных векторов

1.3 Разведочный анализ данных

Прежде чем передать данные в работу моделей машинного обучения, необходимо обработать и очистить их. Очевидно, что «грязные» и необработанные данные могут содержать искажения и пропущенные значения – это ненадёжно, поскольку способно привести к крайне неверным результатам по итогам моделирования. Но безосновательно удалять что-

либо тоже неправильно. Именно поэтому сначала набор данных надо изучить (рисунок 20).

```
In [24]: a = df.describe()
         a.T
```

Out[24]:

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки	1023.0	0.491691	0.500175	0.000000	0.000000	0.000000	1.000000	1.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Рисунок 20 - описательная статистика датасета

Цель разведочного анализа - получение первоначальных представлений о характерах распределений переменных исходного набора данных, формирование оценки качества исходных данных (наличие пропусков, выбросов), выявление характера взаимосвязи между переменными с целью последующего выдвижения гипотез о наиболее подходящих для решения задачи моделях машинного обучения (рисунок 21).

```
In [30]: # Проверим на дубликаты
         df.duplicated().sum()

Out[30]: 0
```

Рисунок 21 - проверка датасета на наличие дубликатов

В качестве инструментов разведочного анализа используется: оценка статистических характеристик датасета; гистограммы распределения каждой из переменной (несколько различных вариантов); диаграммы ящика с усами (несколько интерактивных вариантов); попарные графики рассеяния точек (несколько вариантов); график «квантиль-квантиль»; тепловая карта (несколько вариантов); описательная статистика для каждой

переменной; анализ и полное исключение выбросов (5 повторных итераций); проверка наличия пропусков и дубликатов; ранговая корреляция Кендалла и Пирсона рисунок 22, 23.

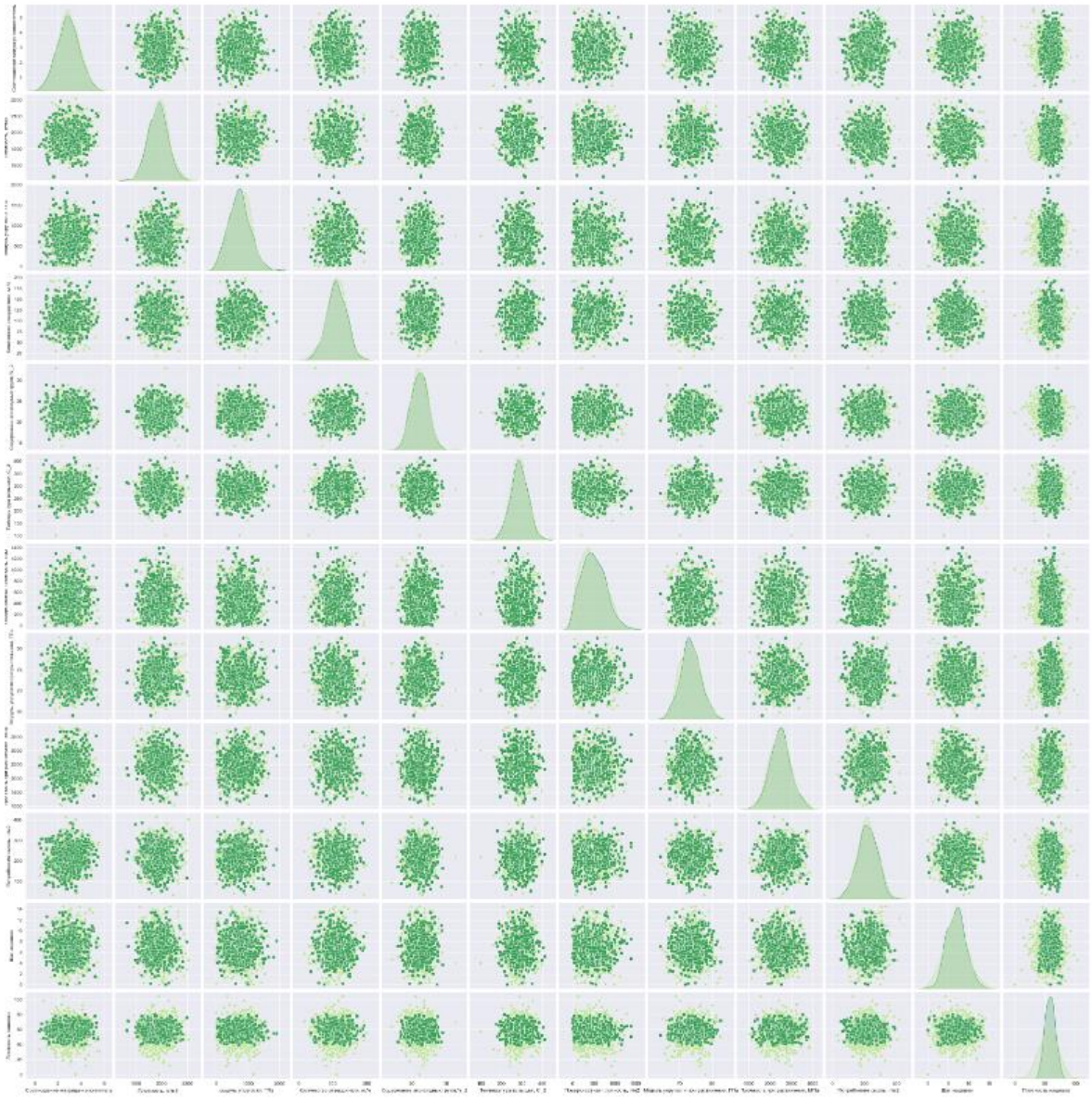


Рисунок 22 - попарные графики рассеяния точек (два разных варианта)

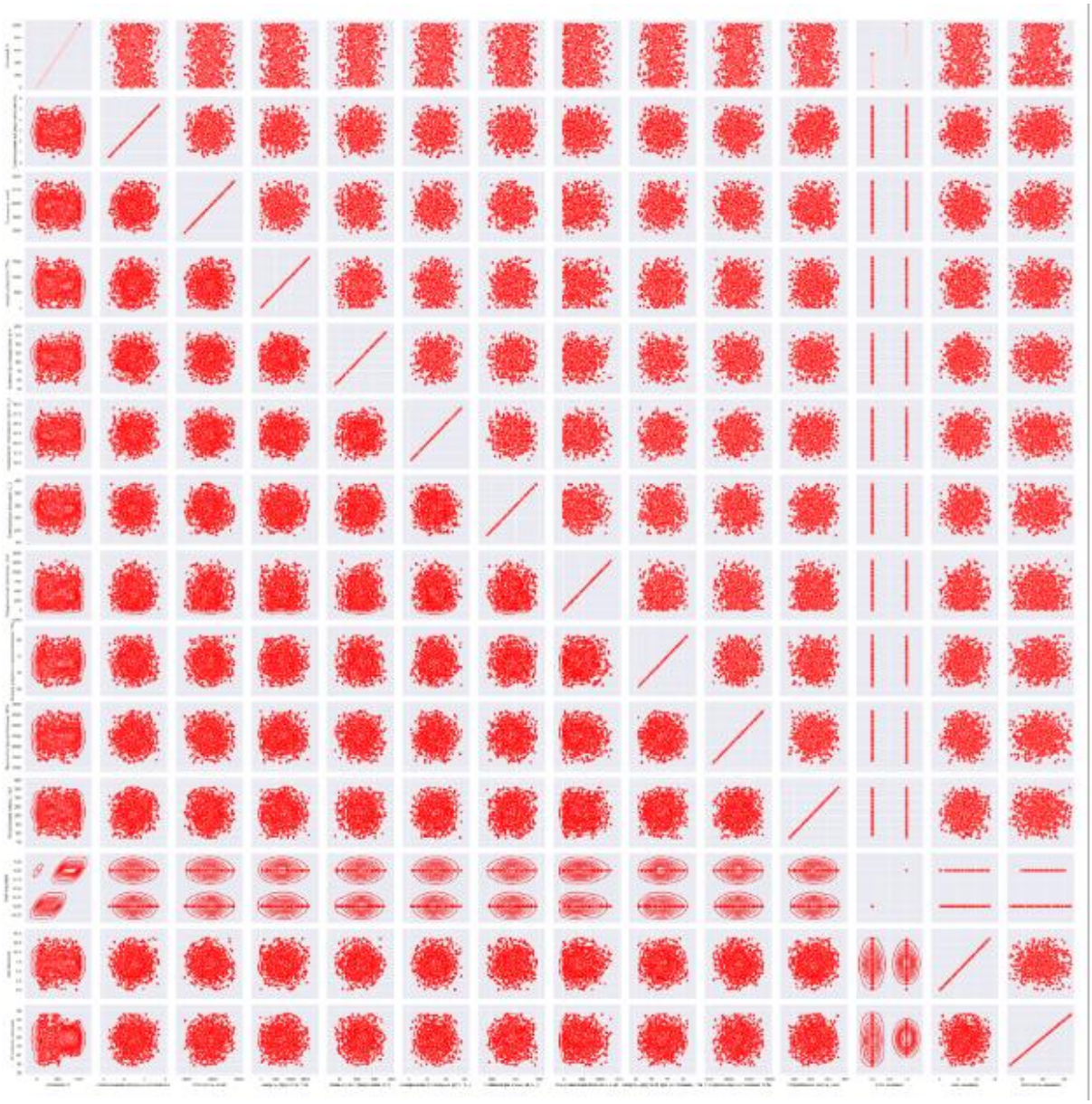


Рисунок 23 - попарные графики рассеяния точек (два разных варианта)

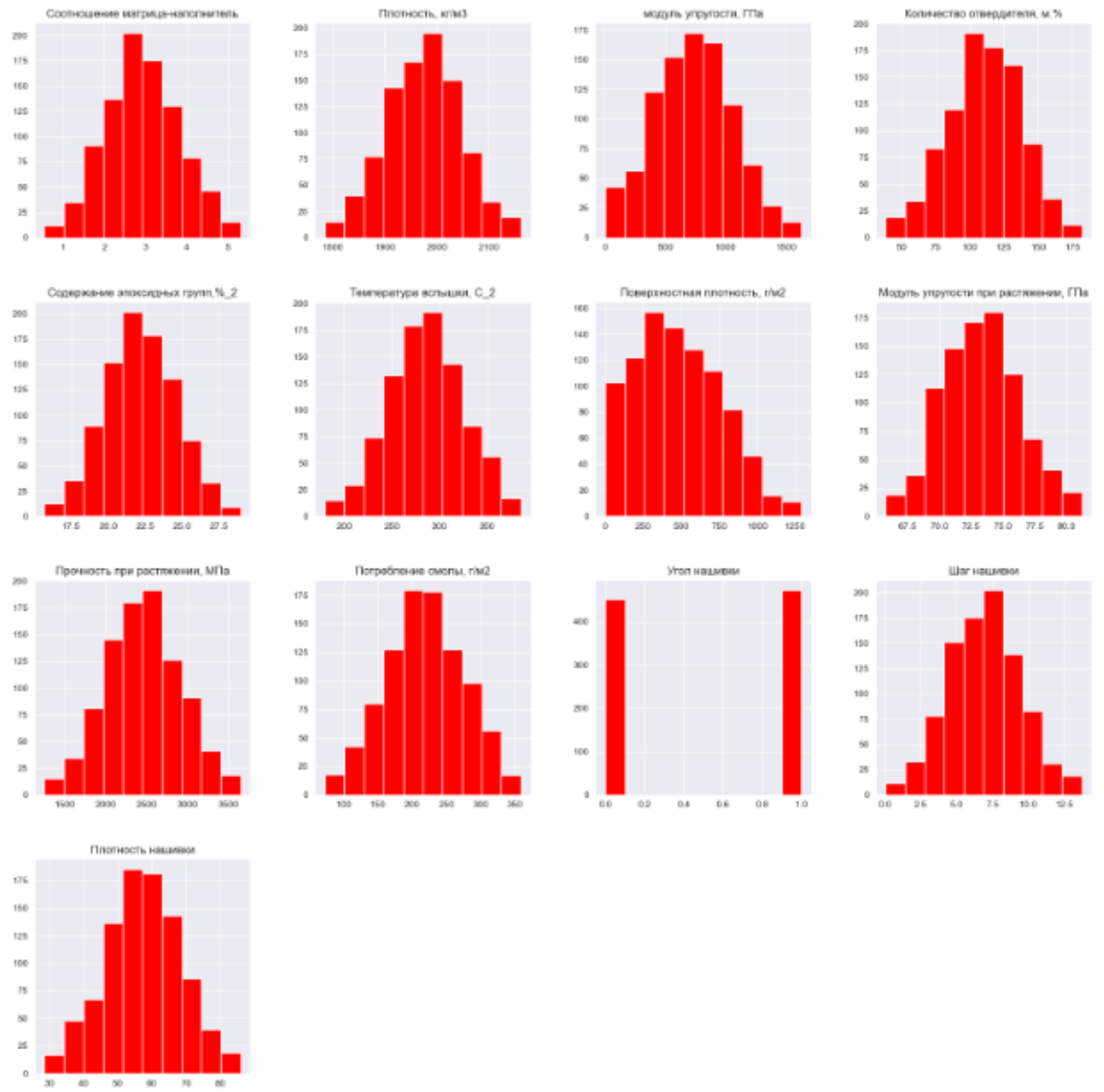
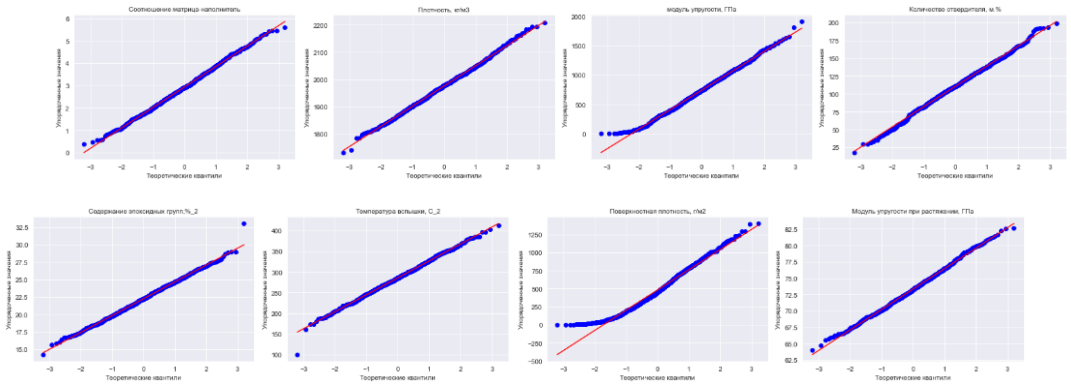


Рисунок 24 - гистограммы распределения

Гистограммы используются для изучения распределений частот значений переменных. Мы видим очень слабую корреляцию между переменными.



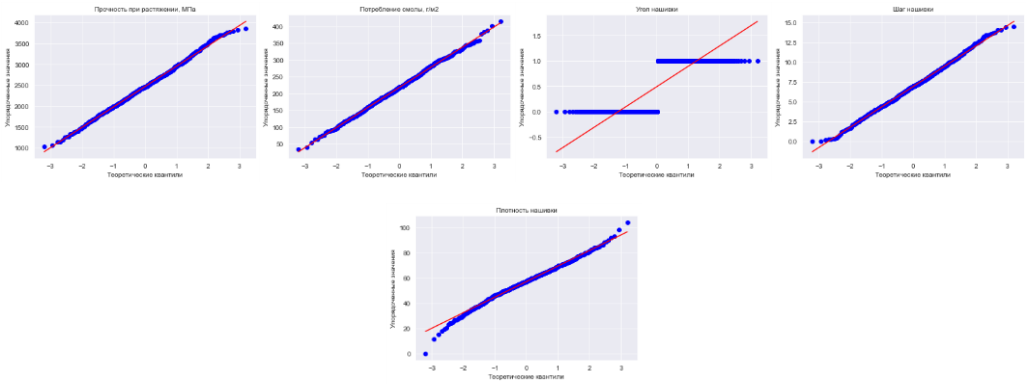


Рисунок 25 – графики «квантиль-квантиль»

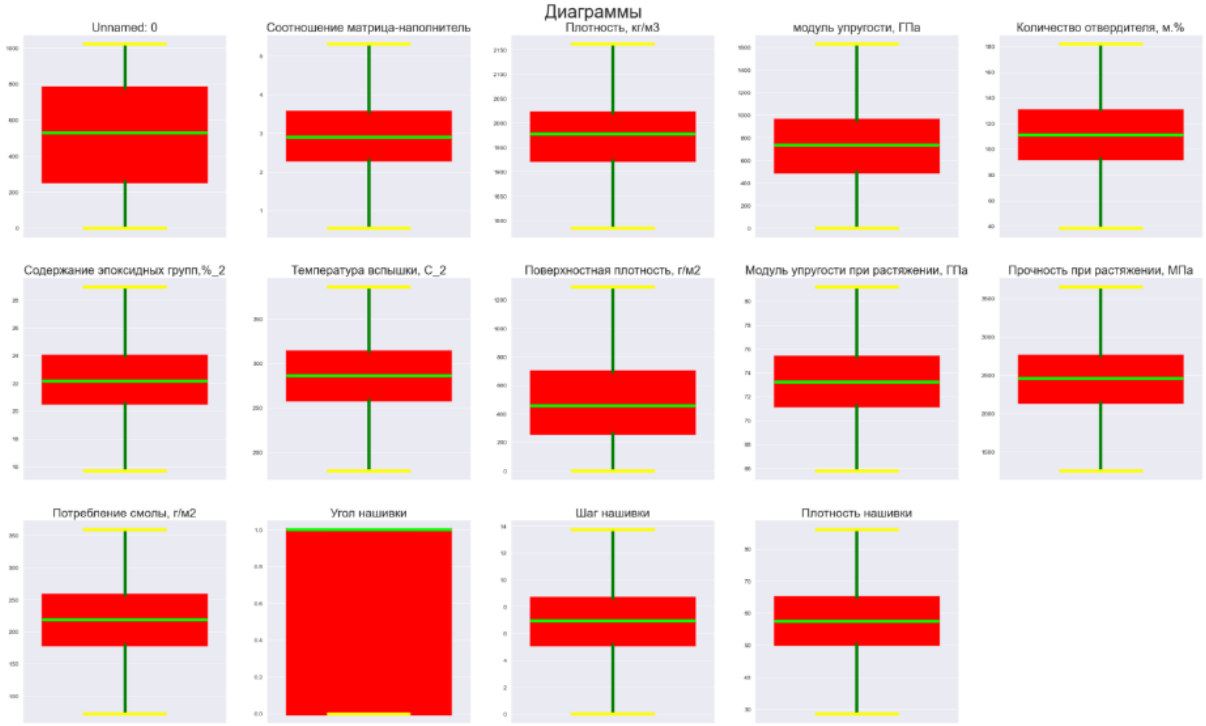


Рисунок 26 – график "ящиков с усами" для всех переменных

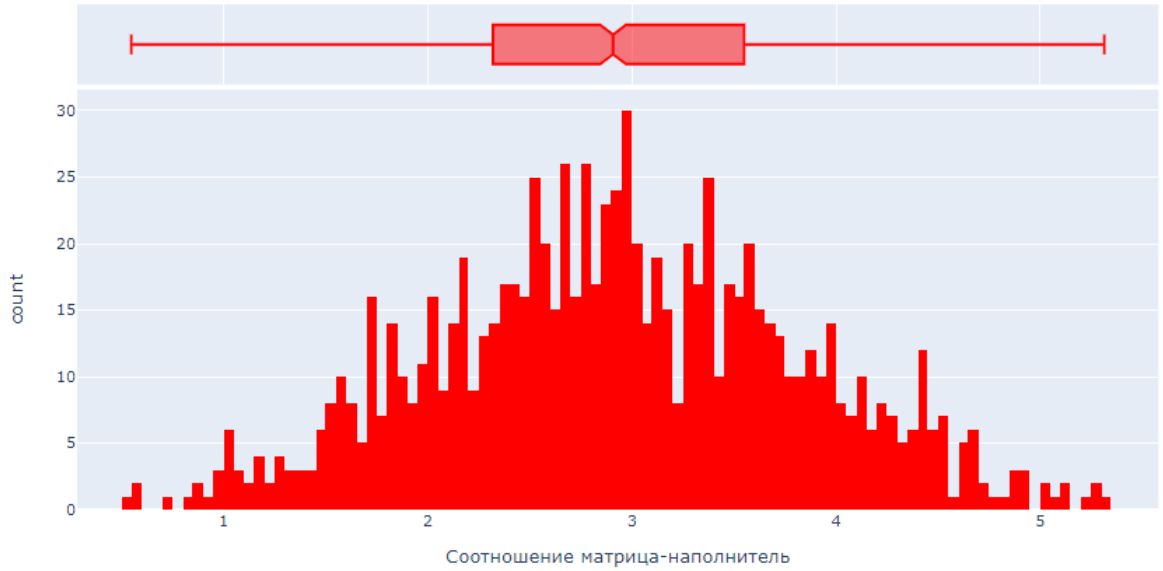


Рисунок 27 - интерактивный график и гистограммы распределения на примере двух переменных

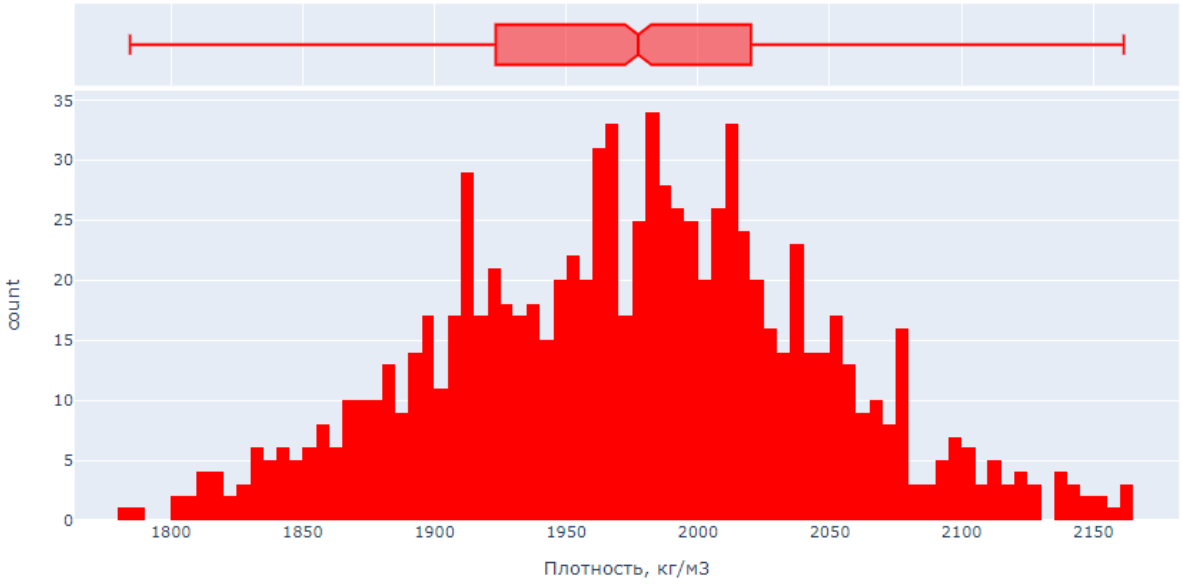


Рисунок 28 - интерактивный график "ящиков с усами" и гистограммы распределения на примере двух переменных

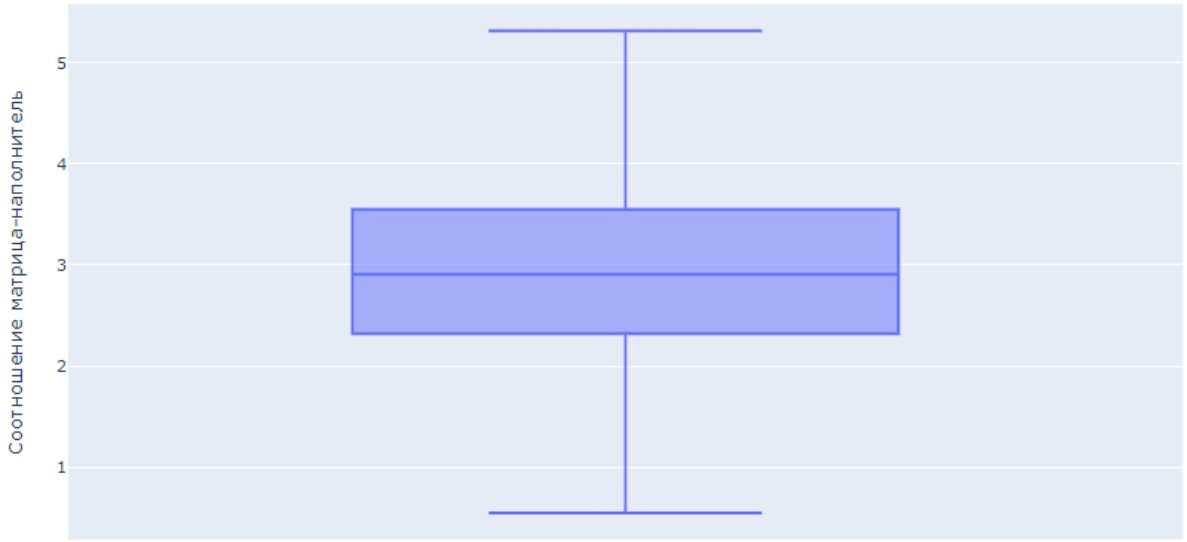


Рисунок 29 - график для одной переменной

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

Данные объединённого датасета не имеют чётко выраженной зависимости, что подтверждает тепловая карта с матрицей корреляции и матрицы диаграмм рассеяния (рисунок 30).

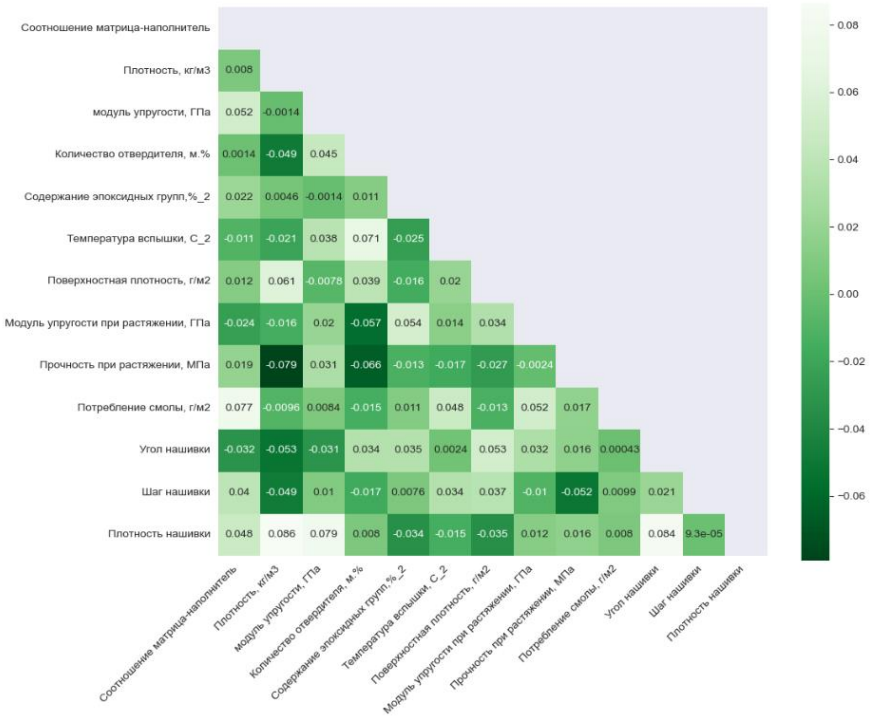


Рисунок 30 - тепловая карта с корреляцией данных

Максимальная корреляция между плотностью нашивки и углом нашивки 0.11, значит нет зависимости между этими данными. Корреляция между всеми параметрами очень близка к 0, корреляционные связи между переменными не наблюдаются.

2. Практическая часть

2. 1. Предобработка данных

По условиям задания нормализуем значения:

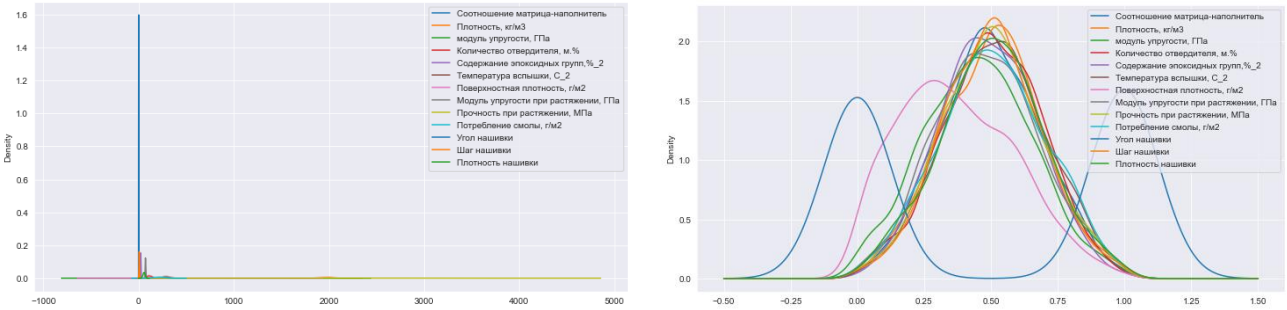


Рисунок 31,32 - визуализированные данные до и после нормализации

2.2. Разработка и обучение модели

Разработка и обучение моделей машинного обучения осуществлялась для двух выходных параметров: «Прочность при растяжении» и «Модуль упругости при растяжении» отдельно. Для решения применим все методы, описанные выше.

```
# Проведем поиск  по сетке гиперпараметров с перекрестной проверкой, количество блоков равно 10 (cv = 10), для
# модели случайного леса - Random Forest Regressor - 2

parametrs = { 'n_estimators': [200, 300],
              'max_depth': [9, 15],
              'max_features': ['auto'],
              'criterion': ['mse'] }
grid = GridSearchCV(estimator = rfr, param_grid = parametrs, cv = 10)
grid.fit(x_train_1, y_train_1)
```

```
GridSearchCV(cv=10,
             estimator=RandomForestRegressor(max_depth=7, n_estimators=15,
                                             random_state=33),
             param_grid={'criterion': ['mse'], 'max_depth': [9, 15],
                         'max_features': ['auto'], 'n_estimators': [200, 300]})
```

```
grid.best_params_
```

```
{'criterion': 'mse',
 'max_depth': 15,
 'max_features': 'auto',
 'n_estimators': 200}
```

```
#Выводим гиперпараметры для оптимальной модели
print(grid.best_estimator_)
knr_upr = grid.best_estimator_
print(f'R2-score RFR для прочности при растяжении, МПа: {knr_upr.score(x_test_1, y_test_1).round(3)}')
```

```
RandomForestRegressor(criterion='mse', max_depth=15, n_estimators=200,
                      random_state=33)
R2-score RFR для прочности при растяжении, МПа: 0.963
```

```
#подставим оптимальные гиперпараметры в нашу модель случайного леса
rfr_grid = RandomForestRegressor(n_estimators = 200, criterion = 'mse', max_depth = 15, max_features = 'auto')
#Обучаем модель
rfr_grid.fit(x_train_1, y_train_1)

predictions_rfr_grid = rfr_grid.predict(x_test_1)
#Оцениваем точность на тестовом наборе
mae_rfr_grid = mean_absolute_error(predictions_rfr_grid, y_test_1)
mae_rfr_grid
```

```
67.60356685553326
```

```
new_row_in_mae_df = {'Perpeccop': 'RandomForest_GridSearchCV', 'MAE': mae_rfr_grid}

mae_df = mae_df.append(new_row_in_mae_df, ignore_index=True)
```

Рисунок 33- поиск гиперпараметров

Порядок разработки модели для каждого параметра и для каждого выбранного метода можно разделить на следующие этапы: разделение нормализованных данных на обучающую и тестовую выборки (в соотношении 70 на 30%); проверка моделей при стандартных значениях; сравнение с результатами модели, выдающей среднее значение; создание графика; сравнение моделей по метрике MAE; поиск сетки гиперпараметров, по которым будет происходить оптимизация модели. В качестве параметра оценки выбран коэффициент детерминации (R2); оптимизация подбора гиперпараметров модели с помощью выбора по сетке и перекрёстной проверки; подстановка оптимальных гиперпараметров в модель и обучение модели на тренировочных данных; оценка полученных данных; сравнение со стандартными значениями.

```
Наилучшие параметры:
{'preprocessing': StandardScaler(), 'regressor': SGDRegressor()}

Наилучшее значение правильности перекрестной проверки: 0.97
Правильность на тестовом наборе: 0.97
```

Рисунок 34 - наилучшие гиперпараметры

Модель после настройки гиперпараметров показала результат немного лучше. Однако, ниже, чем базовая модель. Прочность при растяжении и модуль упругости не имеет линейной зависимости. Все использованные модели не справились с задачей. Результат неудовлетворительный. Свойства композитных материалов в первую очередь зависят от используемых материалов.

Таблица 1. Результаты построения и обучения моделей

	Model	**MAE**	**R2 score**
Модуль упругости при растяжении	KNeighborsRegressor_upr	2.4277	-0.0072
Прочность при растяжении	KNeighborsRegressor_pr	369.52782	-0.0272

	Model	MAE	R2 score
Модуль упругости при растяжении	LinearRegression_upr	2.1782	-0.021000
Прочность при растяжении	LinearRegression_pr	327.278	-0.015
Модуль упругости при растяжении	RandomForestRegressor_upr	2.2828	-0.0178
Прочность при растяжении	RandomForestRegressor_pr	370.4728	-0.0127
Модуль упругости при растяжении	MLPRegressor_upr	2.4527	-0.00827
Прочность при растяжении	MLPRegressor_pr	367.17	-0.04278

2.3. Тестирование модели

После обучения моделей была проведена оценка точности этих моделей на обучающей и тестовых выборках. В качестве параметра оценки модели использовалась средняя квадратическая ошибка (MSE). Результат неудовлетворительный.

	target_var	model_name	MSE	R2
0	Модуль упругости при растяжении, ГПа	LinearRegression	0.02798	-0.027603
1	Прочность при растяжении, МПа	LinearRegression	0.028816	-0.011552
2	Модуль упругости при растяжении, ГПа	KNeighborsRegressor	0.02771	-0.017687
3	Прочность при растяжении, МПа	KNeighborsRegressor	0.02899	-0.017683
4	Модуль упругости при растяжении, ГПа	RandomForestRegressor	0.027618	-0.014302
5	Прочность при растяжении, МПа	RandomForestRegressor	0.028608	-0.004257

Рисунок 35 - результат оценки точности по MSE и R2

Out[226]:

	Регрессор	MAE
0	Support Vector	78.477914
1	RandomForest	76.589025
2	Linear Regression	61.986894
3	GradientBoosting	64.728717
4	KNeighbors	102.030259
5	DecisionTree	107.158013
6	SGD	181.624450
7	MLP	1808.547264
8	Lasso	69.474334
9	RandomForest_GridSearchCV	67.603567
10	KNeighbors_GridSearchCV	99.281694
11	DecisionTree_GridSearchCV	168.624997
12	RandomForest1_GridSearchCV	2.627032
13	KNeighbors1_GridSearchCV	2.757781

Рисунок 36 - результат оценки точности по MAE

Хотя в целом при таких результатах можно применять среднее значение переменной в качестве прогнозного.

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

Обучение нейронной сети— это такой процесс, при котором происходит подбор оптимальных параметров модели, с точки зрения минимизации функционала ошибки. Начнём стоить нейронную сеть с помощью класса keras.Sequential.

```
# Сформируем входы и выход для модели

tv = df['Соотношение матрица-наполнитель']
tr_v = df.loc[:, df.columns != 'Соотношение матрица-наполнитель']

# Разбиваем выборки на обучающую и тестовую
x_train, x_test, y_train, y_test = train_test_split(tr_v, tv, test_size = 0.3, random_state = 14)
```

```
# Нормализуем данные

x_train_n = tf.keras.layers.Normalization(axis =-1)
x_train_n.adapt(np.array(x_train))
```

Рисунок 37 - создание нейронной сети

Определим параметры, поищем оптимальные параметры, посмотрим на результаты. С помощью KerasClassifier выйдем на наилучшие параметры для нашей нейронной сети и построим окончательную нейросеть.

```
# построение окончательной модели
model = create_model(lyrs=[128, 64, 16, 3], dr=0.05)

print(model.summary())
```

```
Model: "sequential_195"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_493 (Dense)           (None, 128)               1664

 dense_494 (Dense)           (None, 64)                8256

 dense_495 (Dense)           (None, 16)                1040

 dense_496 (Dense)           (None, 3)                 51

 dropout_195 (Dropout)       (None, 3)                 0

 dense_497 (Dense)           (None, 3)                 12

=================================================================
Total params: 11,023
Trainable params: 11,023
Non-trainable params: 0
_________________________________________________________________
None
```

Рисунок 38 - построение первой нейросети

Обучим и оценим модель, посмотрим на потери, зададим функцию для визуализации факт/прогноз для результатов моделей.

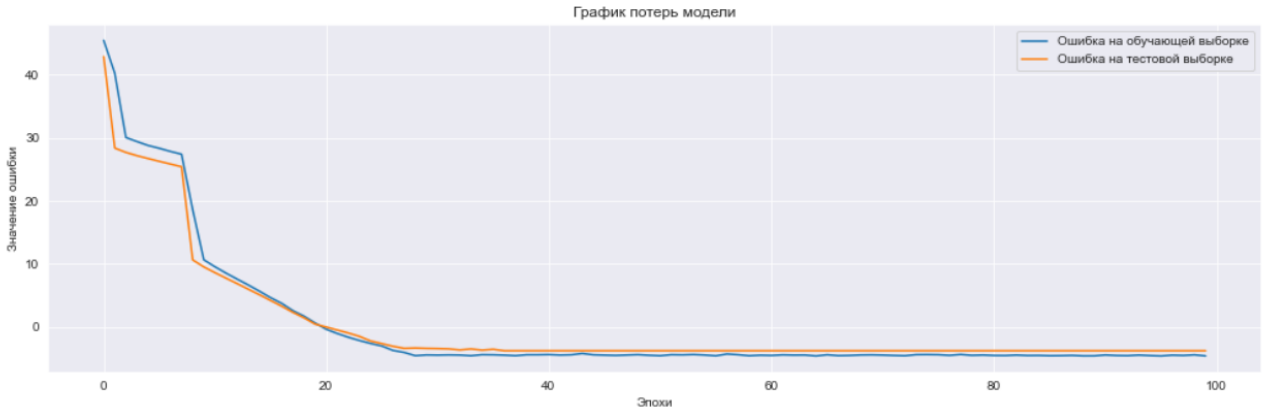


Рисунок 39 - график потерь модели 1

Рисунок 40 - тестовые и прогнозные значения модели 1

Не удовлетворившись таким результатом, создадим другую простую модель глубокого обучения с другой архитектурой. Обучим её, посмотрим на потери, оценим MSE, построим график.

```
# Сконфигурируем модель, зададим слои

model = tf.keras.Sequential([x_train_n, layers.Dense(128, activation='relu'),
                                        layers.Dense(128, activation='relu'), Dropout(0.8),
                                        layers.Dense(128, activation='relu'),
                                        layers.Dense(64, activation='relu'),
                                        layers.Dense(32, activation='relu'),
                                        layers.Dense(16, activation='relu'),
                                        layers.Dense(1)
                             ])

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

model.summary()
```

Рисунок 41 - создание второй модели

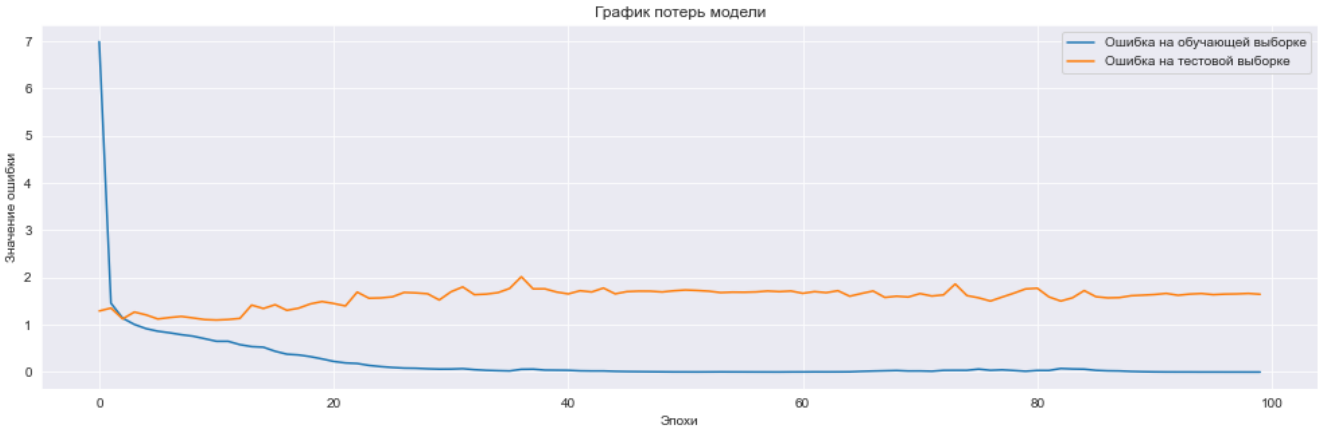


Рисунок 42 - график потерь второй модели

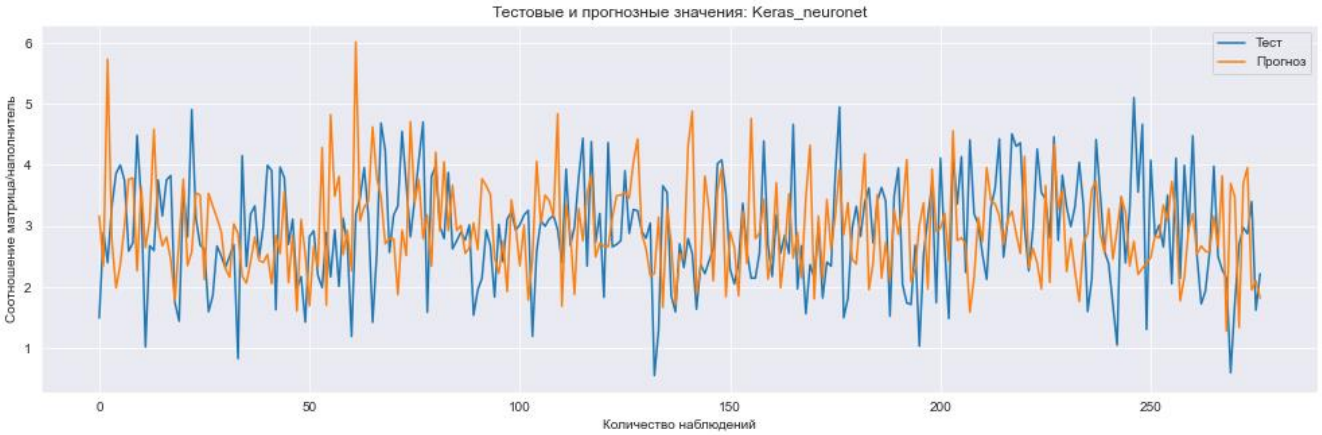


Рисунок 43- тестовые и прогнозные значения второй модели

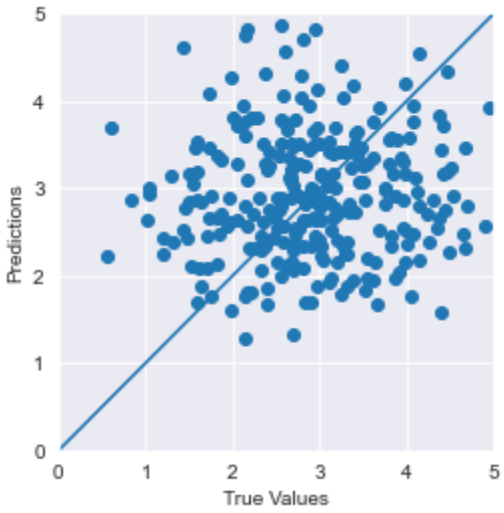


Рисунок 44 – график прогнозных и настоящих значений

Разработка приложения

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

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

```
def app_calculation():
    param_lst = []
    message = ''
    if request.method == 'POST':

       # получим данные из наших форм и кладем их в список, который затем передадим функции set_params
        for i in range(1,13,1):
            param = request.form.get(f'param{i}')
            param_lst.append(float(param))

        message = set_params(*param_lst)
```

Рисунок 45 - часть кода приложения

```
# Запускаем приложение
app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
```

Рисунок 46 - ссылка для открытия html - файла

В открывшемся окне пользователю необходимо ввести в соответствующие ячейки требуемые значения и нажать на кнопку «Готово».

Файлы загружены на: https://github.com/FelixDryaev/Dryaev_Felix_DATA_DIPLOM

3. Заключение

Распределение полученных данных в объединённом датасете близко к нормальному, но коэффициенты корреляции между парами признаков стремятся к нулю. Использованные при разработке моделей подходы не позволили получить сколько-нибудь достоверных прогнозов. Лучшие метрики для модуля упругости при растяжении, ГПа – метод опорных векторов, для прочности при растяжении, МПа - лассо-регрессия.

Сделан вывод, что невозможно определить из свойств материалов соотношение «матрица – наполнитель». Это указывает на то, что прогнозирование характеристик композитных материалов на основании предоставленного набора данных невозможно, но может указывать на недостатки базы данных, подходов, использованных при прогнозе, необходимости пересмотра инструментов для прогнозирования.

Прогнозирование конечных свойств композитных материалов без изучения материаловедения, погружения в вопрос экспериментального анализа характеристик композитных материалов не демонстрирует сколько-нибудь удовлетворительных результатов. Проработка моделей и построение прогнозов требует внедрения в процесс производных от имеющихся показателей для выявления иного уровня взаимосвязей.

4. Список используемой литературы и веб ресурсы.

1. Devpractice Team. Python. Визуализация данных. Matplotlib. Seaborn. Mayavi. - devpractice.ru. 2020. - 412 с.: ил.
2. Абросимов Н.А.: Методика построения разрешающей системы уравнений динамического деформирования композитных элементов конструкций (Учебно-методическое пособие), ННГУ, 2010
3. Грас Д. Data Science. Наука о данных с нуля: Пер. с англ. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2021. - 416 с.: ил.
4. Документация по библиотеке keras: – Режим доступа: https://keras.io/api/.(дата обращения: 13.04.2023).
5. Документация по библиотеке matplotlib: – Режим доступа: https://matplotlib.org/stable/users/index.html. (дата обращения: 12.03.2023)
6. Документация по библиотеке numpy: – Режим доступа: https://numpy.org/doc/1.22/user/index.html#user. (дата обращения: 25.03.2023).
7. Документация по библиотеке seaborn: – Режим доступа: https://seaborn.pydata.org/tutorial.html. (дата обращения: 08.02.2023).
8. Документация по библиотеке Tensorflow: – Режим доступа: https://www.tensorflow.org/overview (дата обращения: 15.03.2023).
9. Документация по языку программирования python: – Режим доступа: https://docs.python.org/3.8/index.html. (дата обращения: 17.04.2023).
10. Иванов Д.А., Ситников А.И., Шляпин С.Д – Композиционные материалы: учебное пособие для вузов, 2019. 13 с.