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»

Прогнозирование физико-механических свойств композиционных материалов на основе углеродных волокон

Слушатель Денисов А.Э.

Москва, 2023

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

ВВЕДЕНИЕ

Композиты — это материалы, состоящие из нескольких компонентов, обладающие физико-химическими свойствами, отличными от свойств составляющих.

На сегодняшний день, исследования их свойств стали особенно актуальны, так как эти материалы находятся в списке государственных приоритетов.

Наиболее популярные из композитов — полимерные.

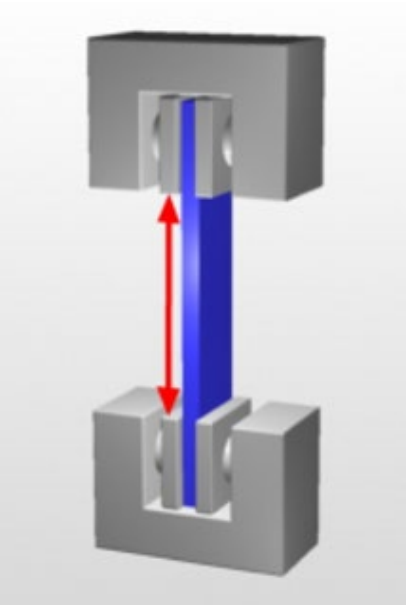
ВВЕДЕНИЕ

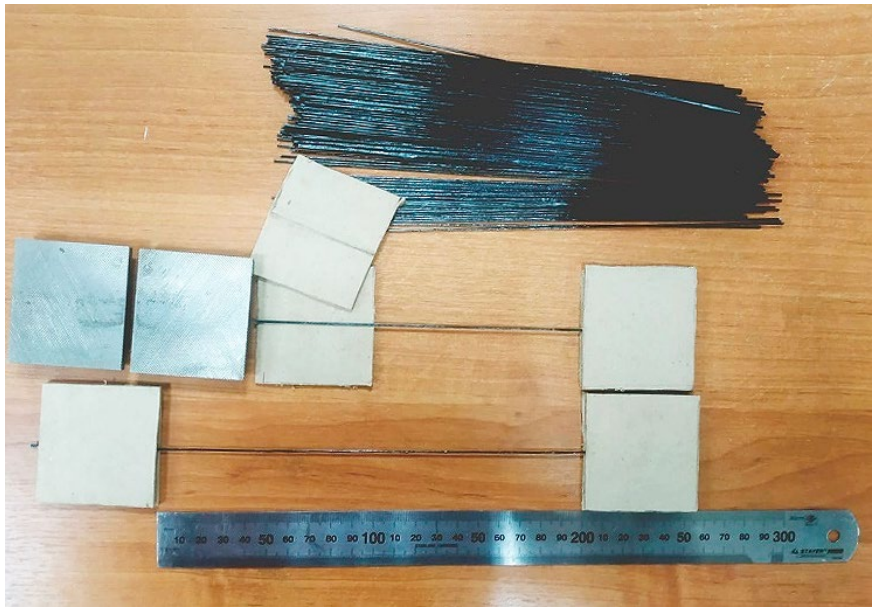
В работу приняты данные **лаборатории функциональных полимерных материалов института новых материалов и нанотехнологий** Университета науки и технологий МИСИС.

Проводились исследования поведения при растяжении образцов непрерывных высокопрочных и высокомодульных одиночных углеродных волокон, пропитанных раствором полисульфона.

Заведующий Лабораторией К.ф.-м.н. Чердынцев В.В., Старший научный сотрудник К.т.н. Степашкин А.А.

Машина для испытаний.



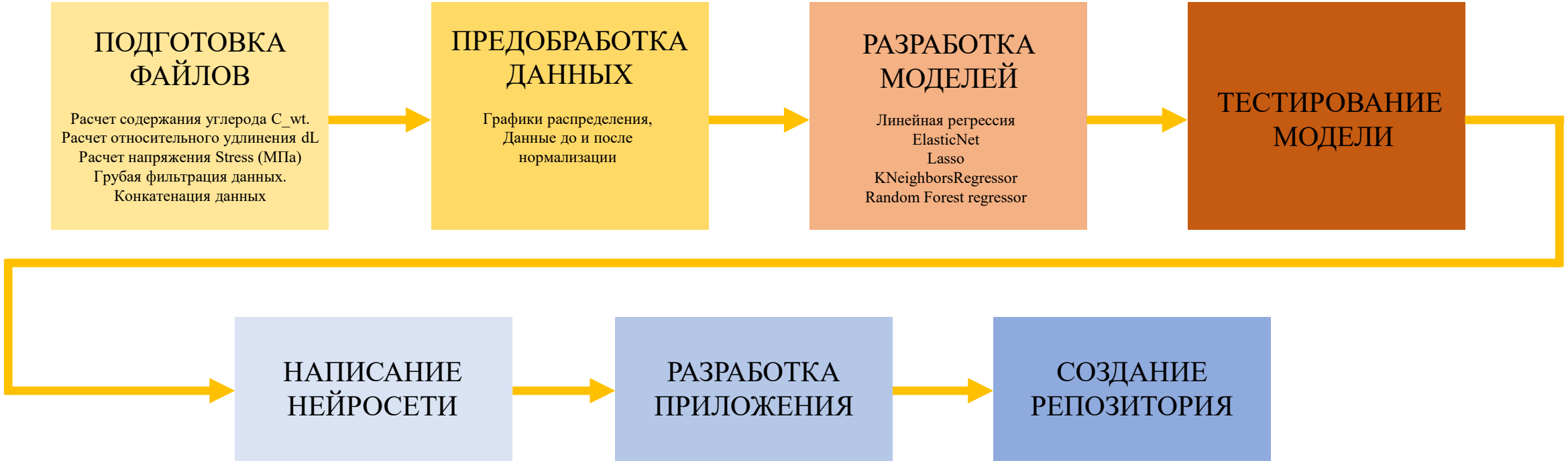


Образцы композита с углеродным волокном.

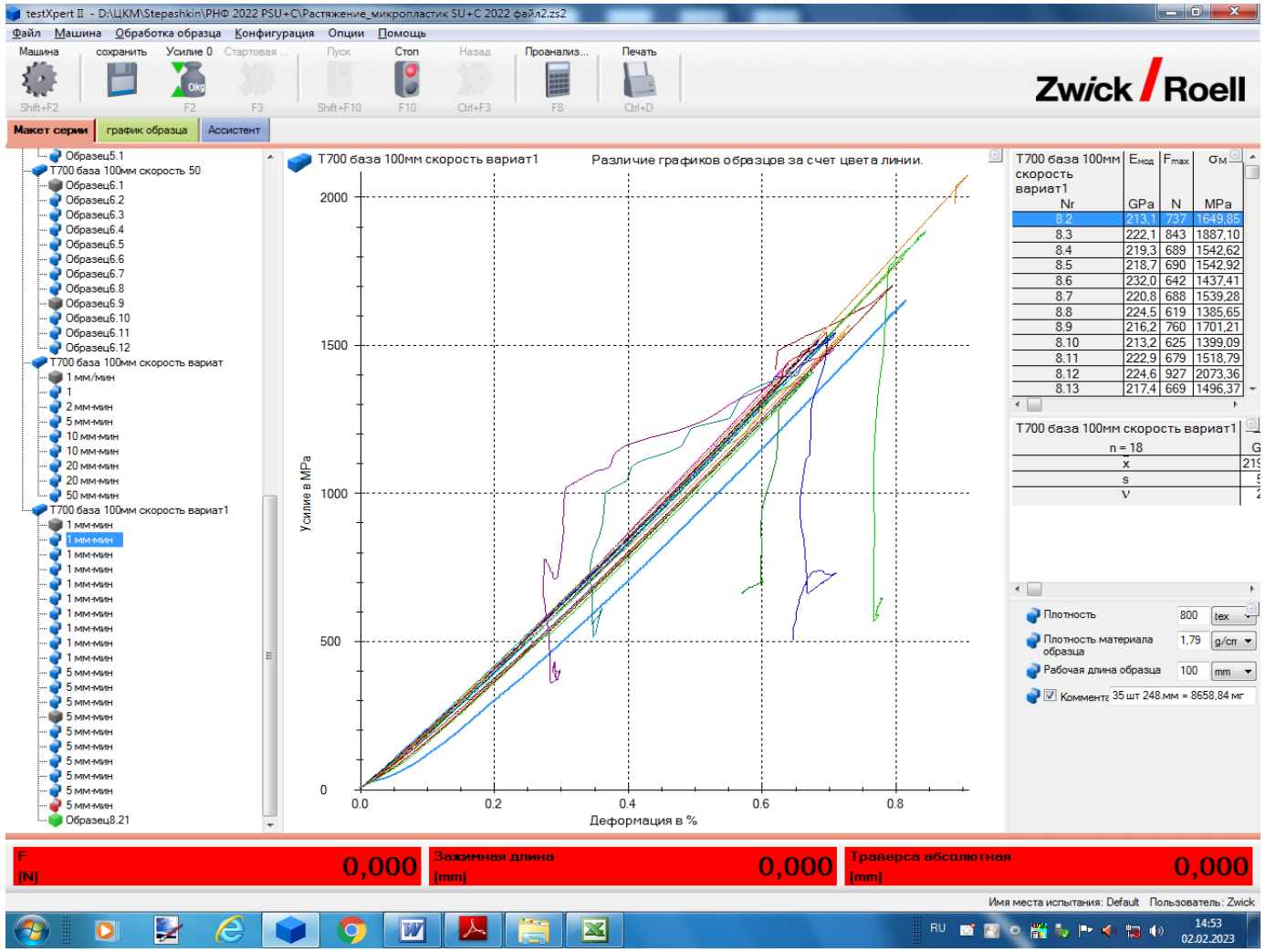
ВВЕДЕНИЕ

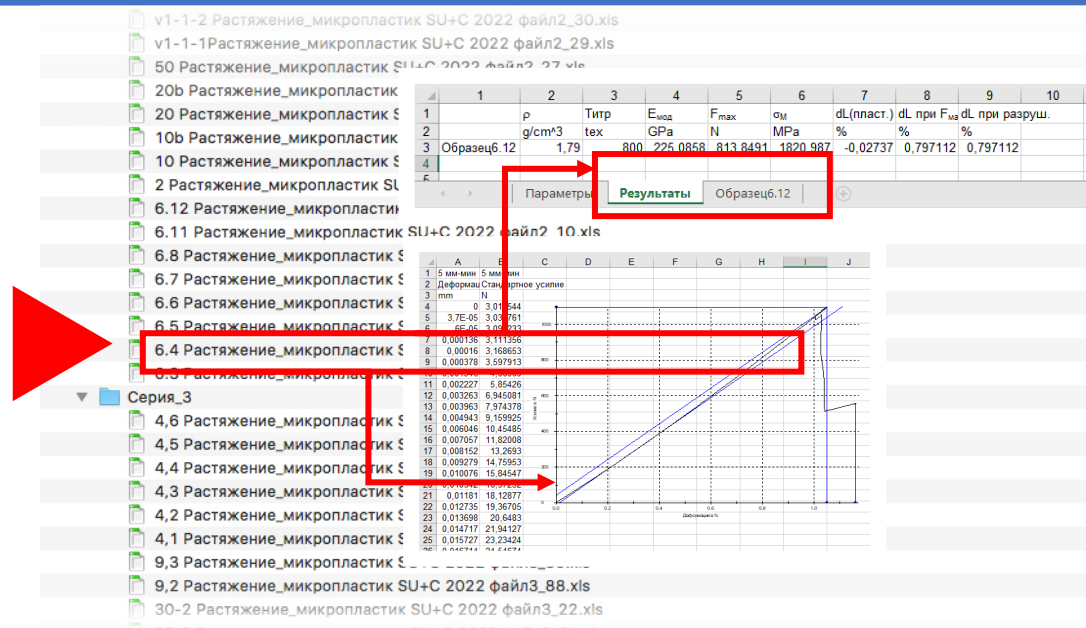
Цели работы: подготовить и изучить экспериментальные данные для дальнейшего прогнозирования средствами машинного обучения физико-механических свойств композиционных материалов на основе углеродных волокон.

Схема процесса



ПОДГОТОВИТЕЛЬНАЯ РАБОТА С ФАЙЛАМИ





Общее число экспериментов – 168,
общее число строк – 93720

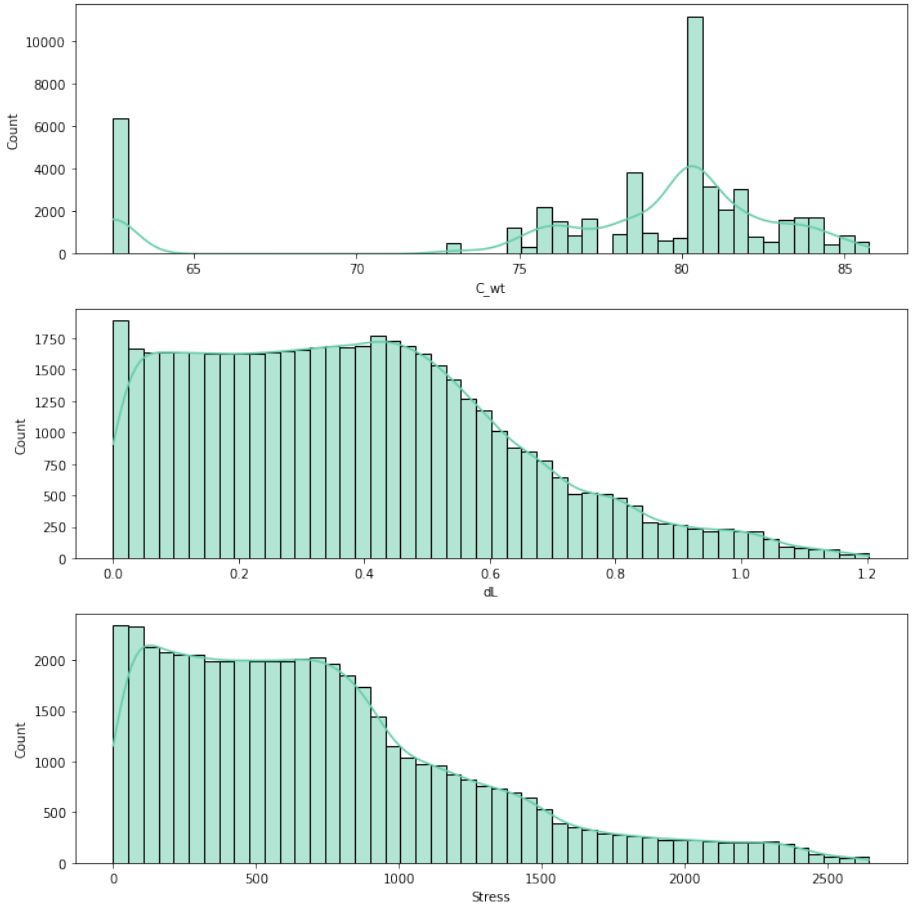
Снимок экрана программы testXpert®

Full_Stress_df.csv

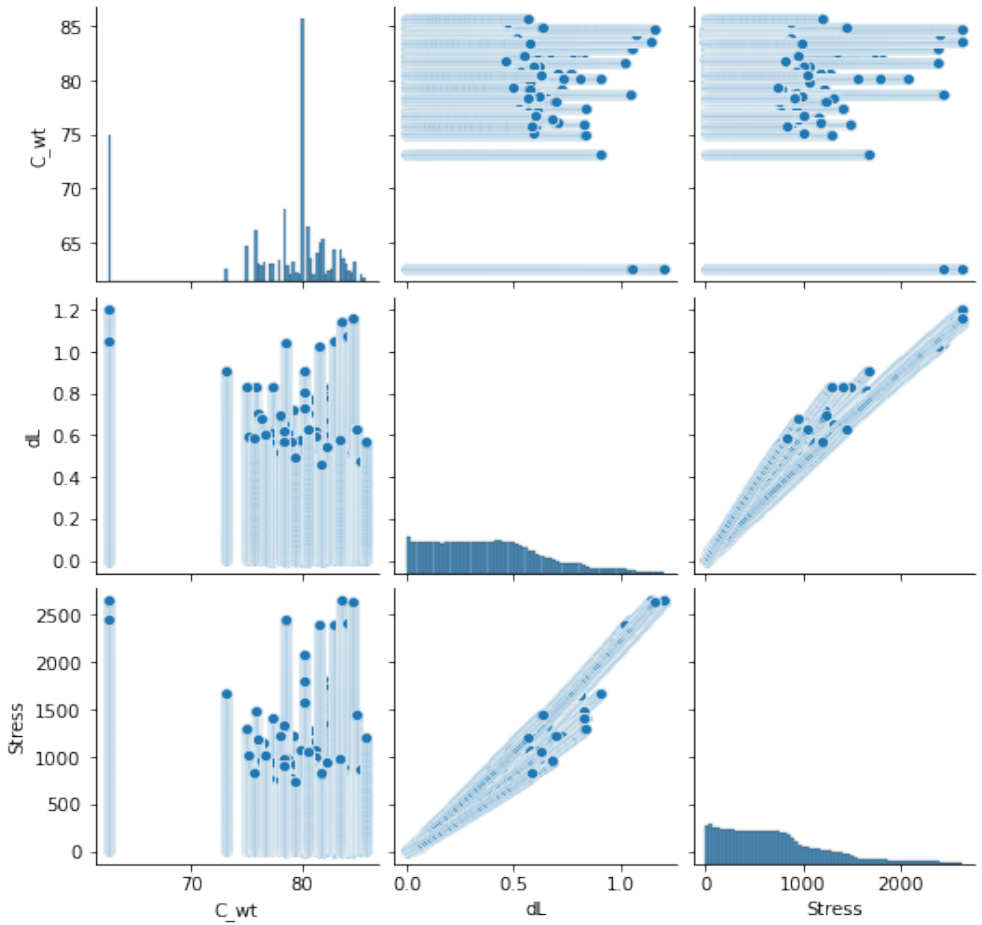
2. ПРАКТИЧЕСКАЯ ЧАСТЬ

ПРЕДОБРАБОТКА

Выполним разведочный анализ - поиск закономерностей в подготовленных данных с помощью визуализации

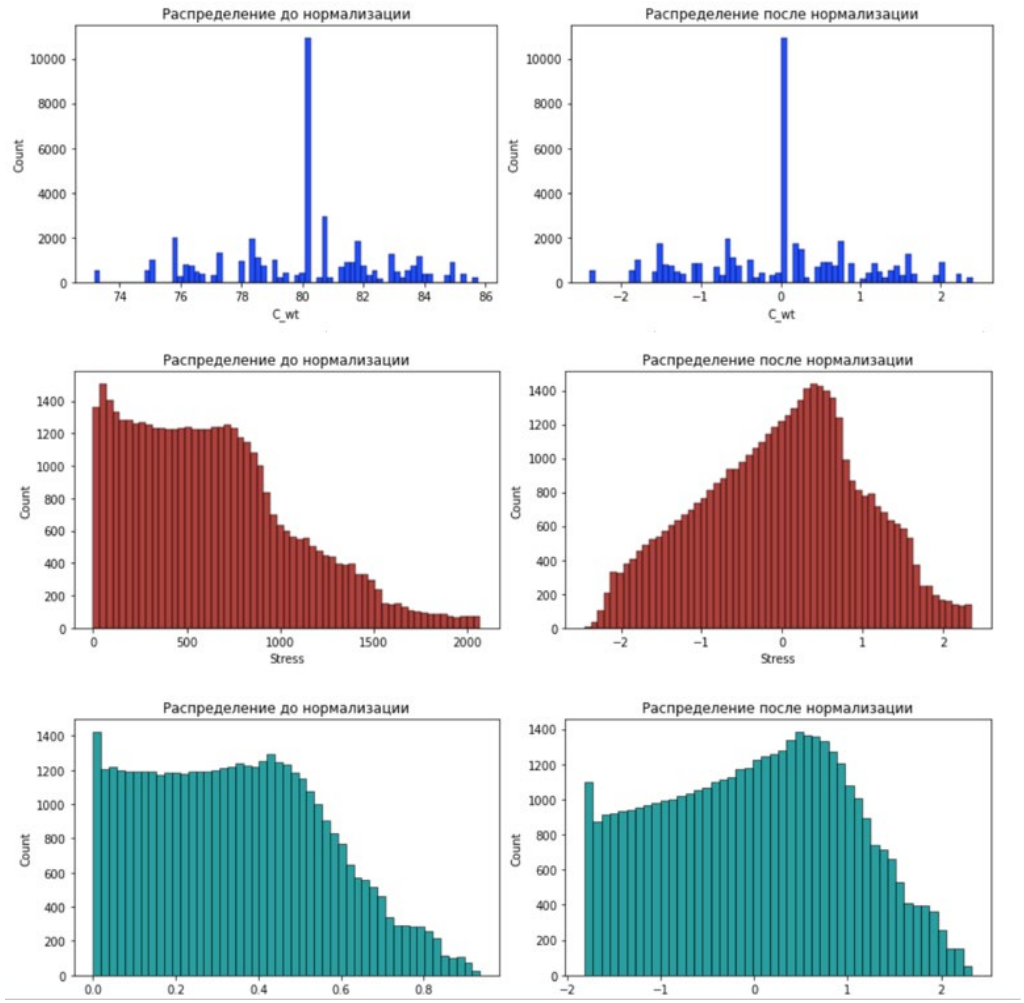


Гистограммы признаков.

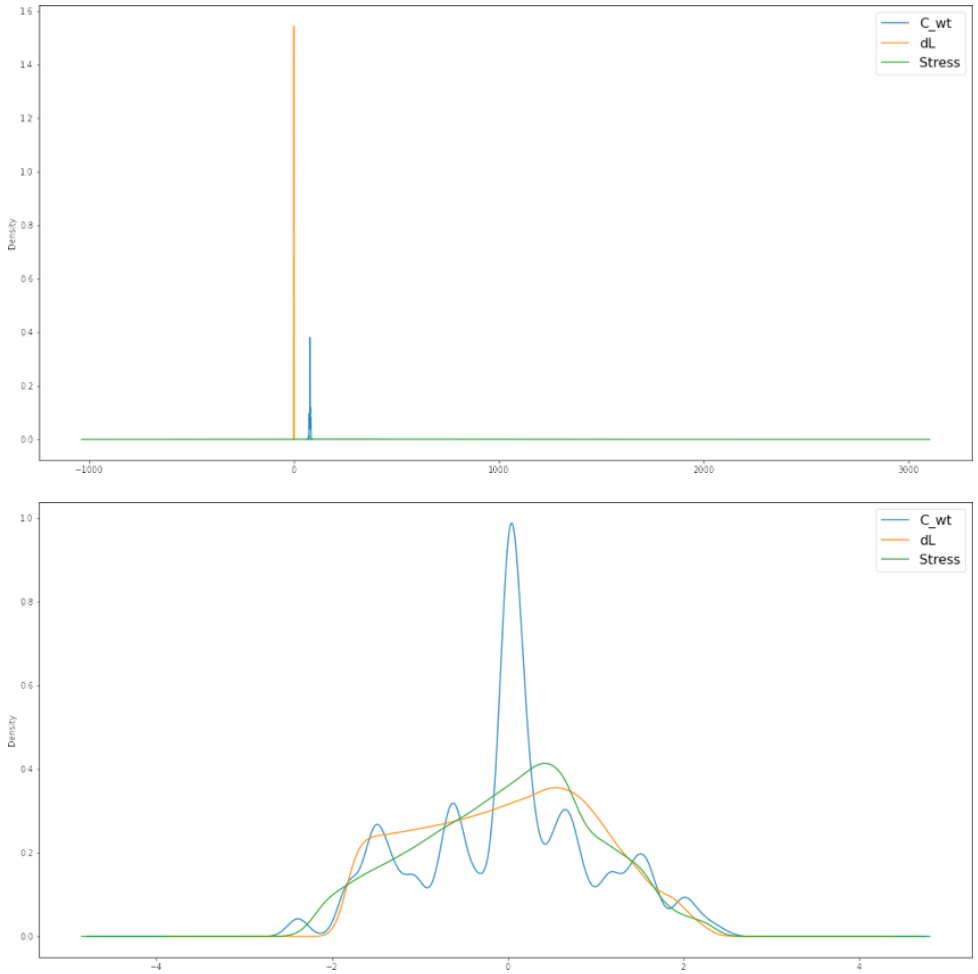


Парные диаграммы распределения всех признаков.

ДАННЫЕ ДО И ПОСЛЕ НОРМАЛИЗАЦИИ

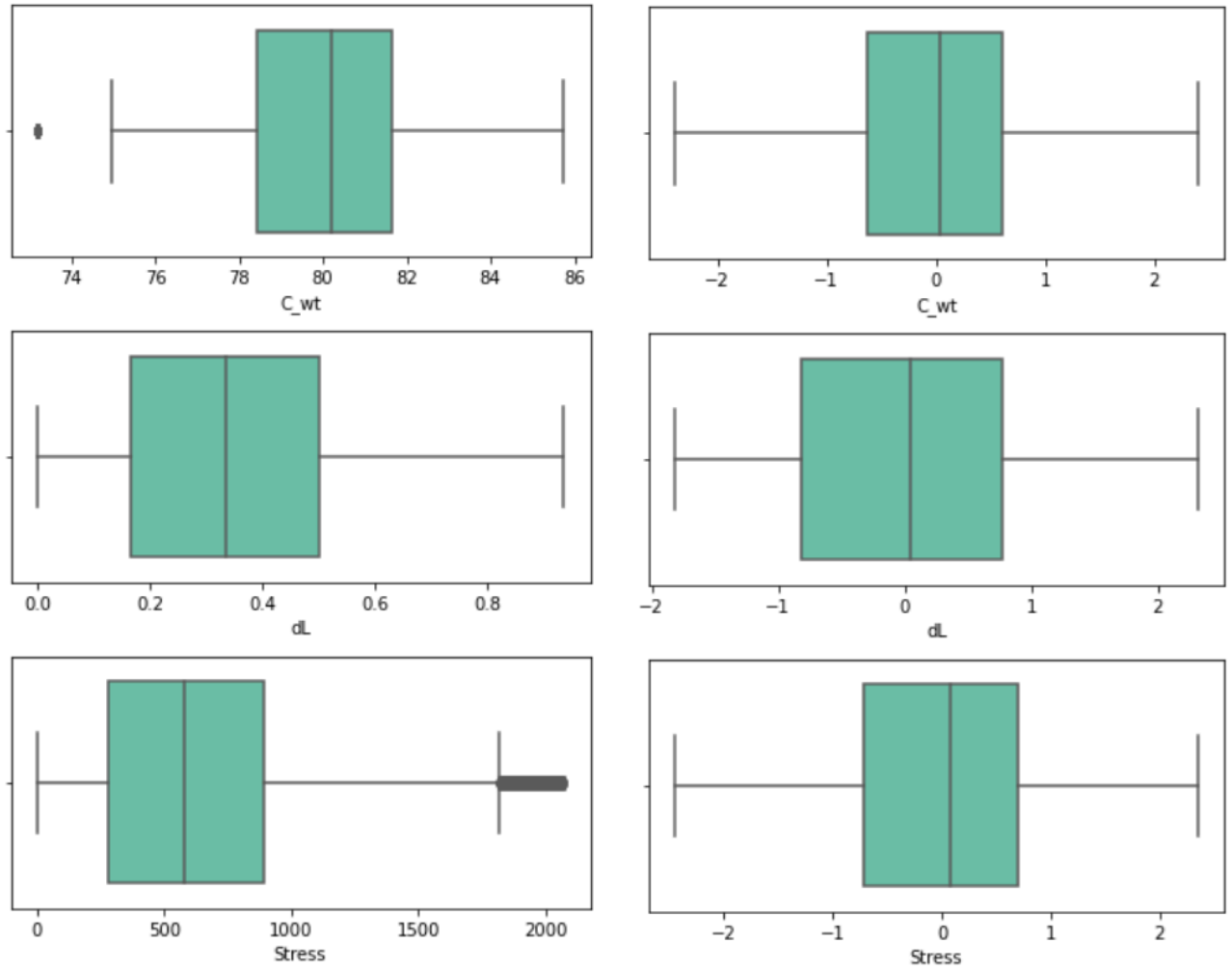


Гистограммы признаков до и после нормализации.

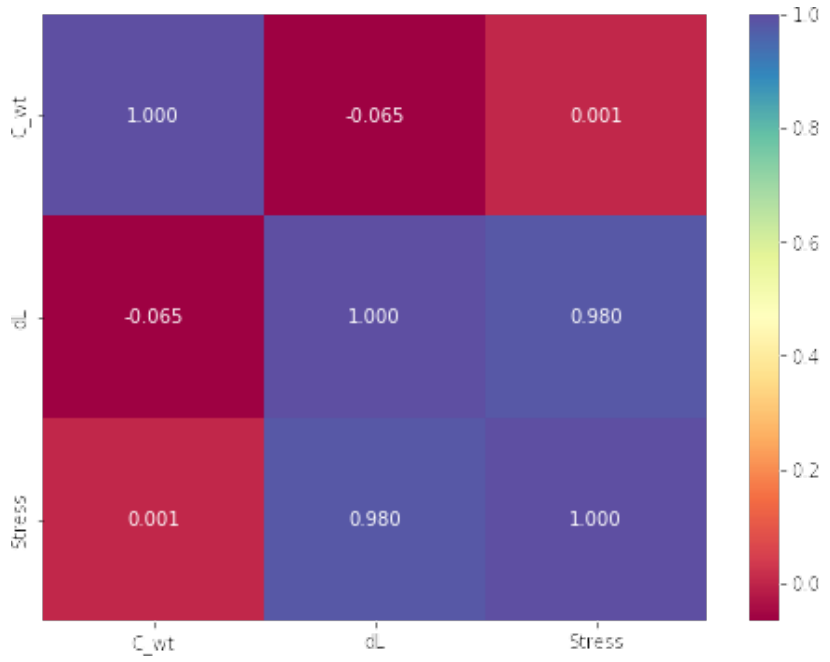


Ядерная плотность до и после нормализации.

ДАННЫЕ ДО И ПОСЛЕ НОРМАЛИЗАЦИИ



Boxplot до и после нормализации.



	C_wt	dL	Stress
0	-1.026045	-1.814900	-2.247307
1	-1.026045	-1.814860	-2.247055
2	-1.026045	-1.814207	-2.239318
3	-1.026045	-1.811071	-2.227549
4	-1.026045	-1.805147	-2.210352

	C_wt	dL	Stress
count	4.248500e+04	4.248500e+04	4.248500e+04
mean	1.425611e-14	6.722278e-16	3.385547e-16
std	1.000012e+00	1.000012e+00	1.000012e+00
min	-2.389737e+00	-1.814995e+00	-2.449682e+00
25%	-6.304612e-01	-8.204129e-01	-7.205830e-01
50%	3.221095e-02	4.534098e-02	7.788915e-02
75%	6.063542e-01	7.707815e-01	6.977880e-01
max	2.407360e+00	2.324179e+00	2.354996e+00

Тепловая карта и описательная статистика.

РАЗРАБОТКА И ОБУЧЕНИЕ МОДЕЛЕЙ

Для прогнозирования значения напряжения при растяжении обучим несколько моделей (объем тестовой выборки 30%, обучающей – 70%). Выполним разработку следующих моделей:

ЛИНЕЙНАЯ РЕГРЕССИЯ – метод машинного обучения, который позволяет определить линейное уравнение, лучше всего описывающее корреляцию между зависимой (целевой) и независимыми переменными. Коэффициенты линейной модели минимизируют остаточную сумму квадратов между наблюдаемыми целями в датасете и целевой переменной, предсказанной линейным приближением.

ELASTICNET – это модель линейной регрессии, обученная как с регуляризацией коэффициентов L1 и L2. Данная комбинация позволяет наиболее эффективно изучать разреженную модель, в которой лишь немногие веса отличны от нуля, как у Лассо, сохраняя при этом свойства регуляризации Ridge.

LASSO (Least absolute shrinkage and selection operator) – это вариация линейной регрессии, которая оценивает коэффициенты линейной регрессионной модели. Это метод выполняет как выбор переменных, так и регуляризацию с целью повышения точности прогнозирования и интерпретируемости результирующей статистической модели.

K NEIGHBORS REGRESSOR – метод решения задач классификации и задач регрессии, основанный на поиске ближайших объектов с известными значения целевой переменной. Значения целевой переменной прогнозируются путем локальной интерполяции целей, связанных с ближайшими соседями в обучающем наборе. В данном методе реализовано обучение на основе k ближайших соседей каждой точки запроса, где k – целочисленное значение, указанное пользователем.

RANDOM FOREST REGRESSOR – метод случайного леса – алгоритм машинного обучения, использующий ансамбли решающих деревьев. Его можно применять как для классификации, так и для регрессии. Он создает деревья решений для случайно выбранных семплов данных, получает прогноз от каждого.

РАЗРАБОТКА И ОБУЧЕНИЕ МОДЕЛЕЙ

Для оценки качества моделей рассчитаем следующие метрики:

- **MAE** – средняя абсолютная ошибка

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

(y_i – фактический ожидаемый результат,
$\hat{y}_i$ – прогноз модели, N – количество примеров)

- **RMSE** – среднеквадратичная ошибка

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

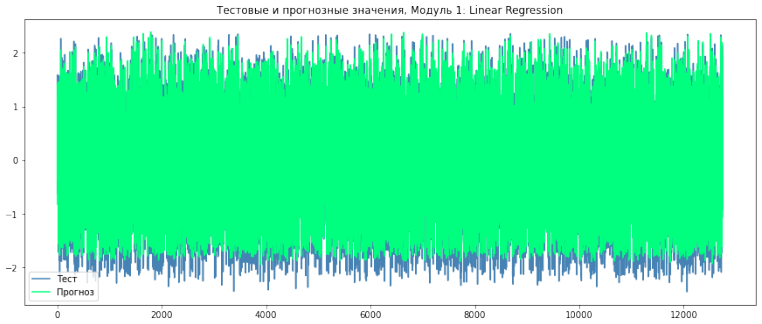
(y_i – фактический ожидаемый результат,
$\hat{y}_i$ – прогноз модели, N – количество примеров)

- **R2** – коэффициент детерминации

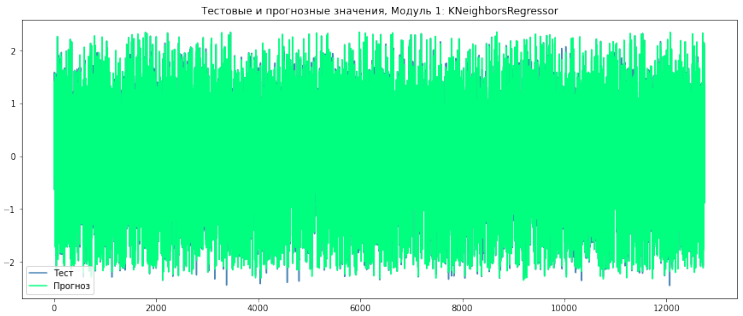
$$R^2 = \frac{\sum_i (\hat{y}_i - \bar{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2},$$

(y_i – фактический ожидаемый результат,
$\hat{y}_i$ – прогноз модели, N – количество примеров)

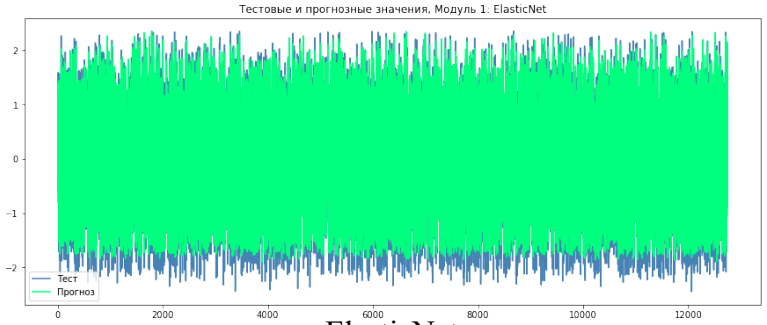
ТЕСТИРОВАНИЕ МОДЕЛЕЙ



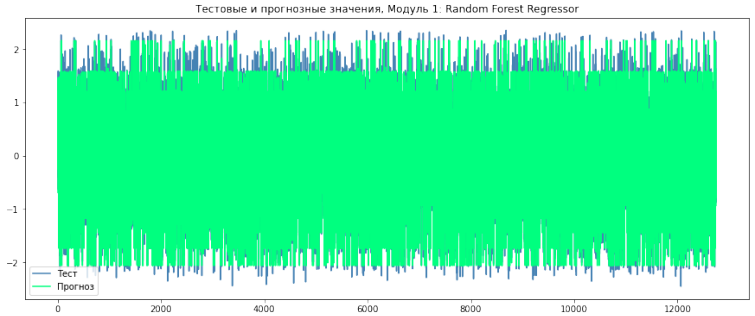
Линейная регрессия



Регрессия К ближайших соседей



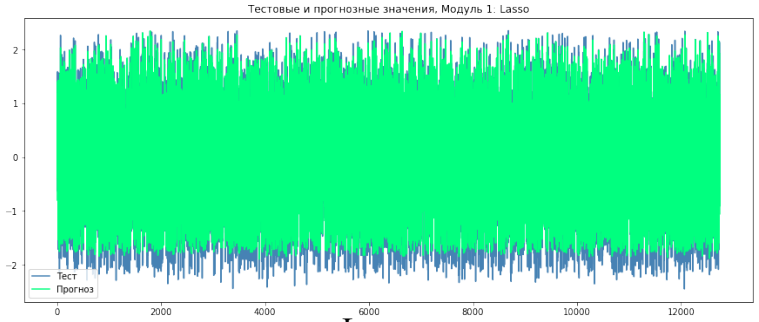
ElasticNet



Random Forest Regressor

прогноз

тест



Lasso

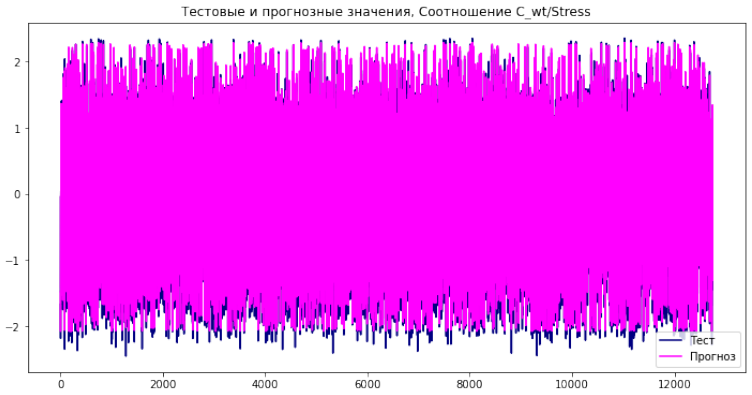
	model	target	MAE	RSME	R2
0	Linear Regression	Stress	0.155514	0.036080	0.963962
1	ElasticNet	Stress	0.155909	0.036211	0.963831
2	Lasso	Stress	0.156013	0.036300	0.963743
3	KNeighborsRegressor	Stress	0.042737	0.005516	0.994491
4	Random Forest Regressor	Stress	0.145526	0.033752	0.966288

Сравнение точности работы протестированных моделей

СОЗДАНИЕ НЕЙРОННОЙ СЕТИ

```python
from keras import layers
from keras import models
def plot_loss(history):
    plt.plot(history.history['loss'], label='loss')
    plt.plot(history.history['val_loss'], label='val_loss')
    plt.ylim([0, 1])
    plt.xlabel('Эпоха')

    plt.ylabel('MAE [MPG]')
    plt.legend()
    plt.grid(True)
model = Sequential()
model.add(layers.Dense(50, input_dim=X.shape[1], activation='relu', name="layer1"))
model.add(layers.Dropout(0.12))
model.add(layers.Dense(128, activation='relu', name="layer2"))
model.add(layers.Dropout(0.12))
model.add(layers.Dense(19, activation='tanh', name="layer3"))
model.add(layers.Dense(1))
```



Сравнение прогноза и теста

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 layer1 (Dense)              (None, 50)                150

 dropout (Dropout)           (None, 50)                0

 layer2 (Dense)              (None, 128)               6528

 dropout_1 (Dropout)         (None, 128)               0

 layer3 (Dense)              (None, 19)                2451

 dense (Dense)               (None, 1)                 20

=================================================================
Total params: 9,149
Trainable params: 9,149
```

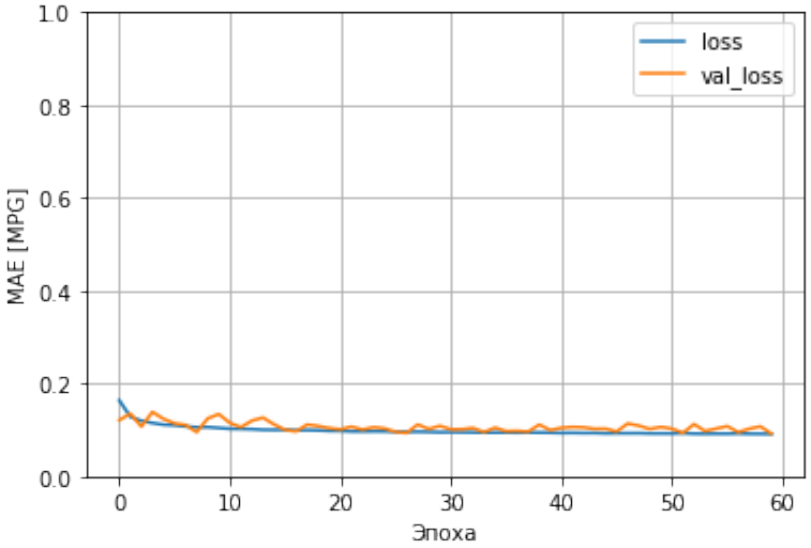


График потерь.

РАЗРАБОТКА ПРИЛОЖЕНИЯ

Приложение, предсказывающее значение напряжения в зависимости от введенных параметров концентрации углерода и относительного удлинения, создавалось с помощью инструмента **Flask** в **Google Colaboratory** и сервиса, предоставляющего возможность создания публичного адреса **ngrok.**

```
!pip install flask-ngrok
!pip install pyngrok
```

```
import pandas as pd
import numpy as np
import copy
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, mean_absolute_percentage_error, r2_score
from sklearn.preprocessing import MinMaxScaler, Normalizer, StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression, LogisticRegression, SGDRegressor, Lasso
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import r2_score
from sklearn.multioutput import MultiOutputRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.preprocessing import PowerTransformer
import tensorflow as tf
from tensorflow import keras
from keras import layers
from keras import models
from keras.engine.sequential import Sequential
from keras.metrics import Precision, Recall
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn import metrics
from sklearn.metrics import mean_absolute_error
import pickle
```

```
[ ] !pip install pyngrok
```

```
ngrok.set_auth_token(auth)
public_url = ngrok.connect(5000).public_url
app = flask.Flask(__name__, template_folder = 'templates')


@app.route('/', methods = ['POST', 'GET'])
@app.route('/index', methods = ['POST', 'GET'])
def main():
    if flask.request.method == 'GET':
        return render_template('main.html')

    if flask.request.method == 'POST':
        with open('KNN_model.pkl', 'r') as f:
            loaded_model = pickle.load(f)

        exp = float(flask.request.form['Stress'])
        y_pred = loaded_model.predict([[exp]])
#       x1 = powertransformer_y_l.inverse_transform(np.array([0.34,0.77]).reshape(1,-1))
#       x1[0][0], x1[0][1]

        return render_template('main.html', result = y_pred)

print(f" Приложение по ссылке :  {public_url}")
app.run(port = port_no)
```

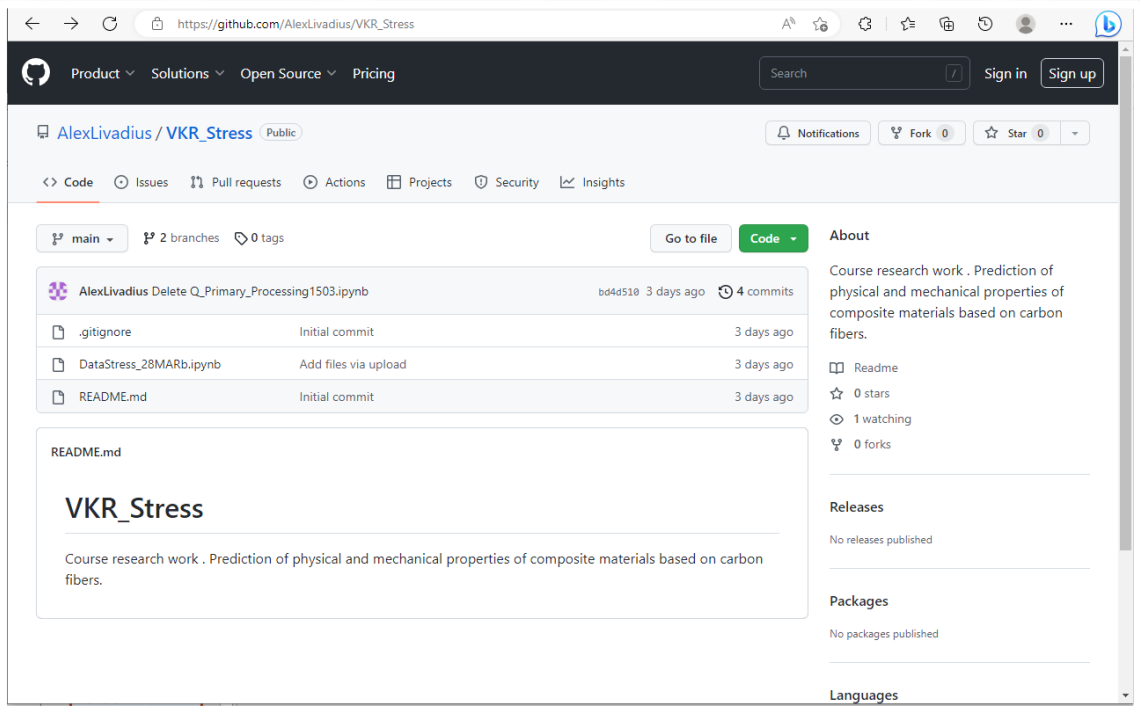
```
[ ] from pyngrok import ngrok
    from flask import Flask
    import flask
    import pickle
    from flask import render_template
    from joblib import load
    import sklearn
    from sklearn.linear_model import LinearRegression

    # ключ для авторизации
    auth = '2NewxdGSMpJeEdf94oPHTJbOhsj_4arPvNU1LaLmGvgsc9jiu'

    port_no = 5000
```

Приложение доступно по ссылке :
http://8f07-34-141-210-133.ngrok.io

СОЗДАНИЕ РЕПОЗИТОРИЯ И ЗАГРУЗКА РЕЗУЛЬТАТОВ

Результаты исследовательской работы загружены на удаленный репозиторий по ссылке:

https://github.com/AlexLivadius/VKR_Stress