ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по курсу «Data Science»

Слушатель: Зюликов Олег Александрович

Постановка задачи

- Загрузить датасет из файлов
- Провести разведочный анализ и очистку данных
- Обучить алгоритмы машинного обучения для определения модуля упругости при растяжении и прочности при растяжении
- Написать нейронную сеть, которая будет рекомендовать соотношение матрица-наполнитель
- Разработать приложение на Flask
- Создать профиль на GitHab
- Сделать commit приложения на GitHab

Предварительная подготовка

- Импорт библиотек
- Загрузка файлов
- Объединение в один датасет

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter('ignore')
%matplotlib inline

from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn import svm
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
import sklearn.metrics as metrics
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, mean_absolute_percentage_error

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.wrappers.scikit_learn import KerasClassifier, KerasRegressor
from tensorflow.keras.utils import normalize
from keras.utils.np_utils import normalize
from tensorflow.keras import optimizers
```

```
df = df1.merge(df2, how='inner', left_index=True, right_index=True)
df
```

	Unnamed: 0_x	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2
0	0.0	1.857143	2030.000000	738.736842	30.000000	22.267857	100.000000
1	1.0	1.857143	2030.000000	738.736842	50.000000	23.750000	284.615385
2	2.0	1.857143	2030.000000	738.736842	49.900000	33.000000	284.615385
3	3.0	1.857143	2030.000000	738.736842	129.000000	21.250000	300.000000
4	4.0	2.771331	2030.000000	753.000000	111.860000	22.267857	284.615385
...	...	...	...	...	...	...	...
1018	1018.0	2.271346	1952.087902	912.855545	86.992183	20.123249	324.774576
1019	1019.0	3.444022	2050.089171	444.732634	145.981978	19.599769	254.215401
1020	1020.0	3.280604	1972.372865	416.836524	110.533477	23.957502	248.423047
1021	1021.0	3.705351	2066.799773	741.475517	141.397963	19.246945	275.779840
1022	1022.0	3.808020	1890.413468	417.316232	129.183416	27.474763	300.952708

1023 rows × 15 columns

```
df1 = pd.read_excel('X_bp.xlsx')
df1
```

Разведочный анализ данных

- Поиск пропущенных значений
- Проверка на дубликаты
- Описательные статистики

```
df.describe()
```

	соот_матр_нап	плотность	мод_упр	кол_отв	сод_эп_гр	темп_всп	пов_плотн	мод_упр_раст	проч_раст	потр_смолы	угол_наш	шаг_наш	плотн_наш
count	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000	1023.000000
mean	2.930366	1975.734888	739.923233	110.570769	22.244390	286.882151	482.731833	73.328571	2466.922843	218.423144	44.252199	6.899222	57.153929
std	0.913222	73.729231	330.231581	28.295911	2.406301	40.943260	281.314690	3.118983	485.628006	59.735931	45.015793	2.563467	12.350969
min	0.389403	1731.764635	2.436909	17.740275	14.254985	100.000000	0.603740	64.054061	1036.856605	33.803026	0.000000	0.000000	0.000000
25%	2.317887	1924.155467	500.047452	92.443497	20.608034	259.066528	266.816645	71.245018	2135.850448	179.627520	0.000000	5.080033	49.799212
50%	2.906878	1977.621657	739.664328	110.564840	22.230744	285.896812	451.864365	73.268805	2459.524526	219.198882	0.000000	6.916144	57.341920
75%	3.552660	2021.374375	961.812526	129.730366	23.961934	313.002106	693.225017	75.356612	2767.193119	257.481724	90.000000	8.586293	64.944961
max	5.591742	2207.773481	1911.536477	198.953207	33.000000	413.273418	1399.542362	82.682051	3848.436732	414.590628	90.000000	14.440522	103.988901

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1023 entries, 0 to 1022
Data columns (total 13 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   соот_матр_нап   1023 non-null   float64
 1   плотность       1023 non-null   float64
 2   мод_упр         1023 non-null   float64
 3   кол_отв         1023 non-null   float64
 4   сод_эп_гр       1023 non-null   float64
 5   темп_всп        1023 non-null   float64
 6   пов_плотн       1023 non-null   float64
 7   мод_упр_раст    1023 non-null   float64
 8   проч_раст       1023 non-null   float64
 9   потр_смолы      1023 non-null   float64
 10  угол_наш        1023 non-null   float64
 11  шаг_наш         1023 non-null   float64
 12  плотн_наш       1023 non-null   float64
dtypes: float64(13)
memory usage: 111.9 KB
```

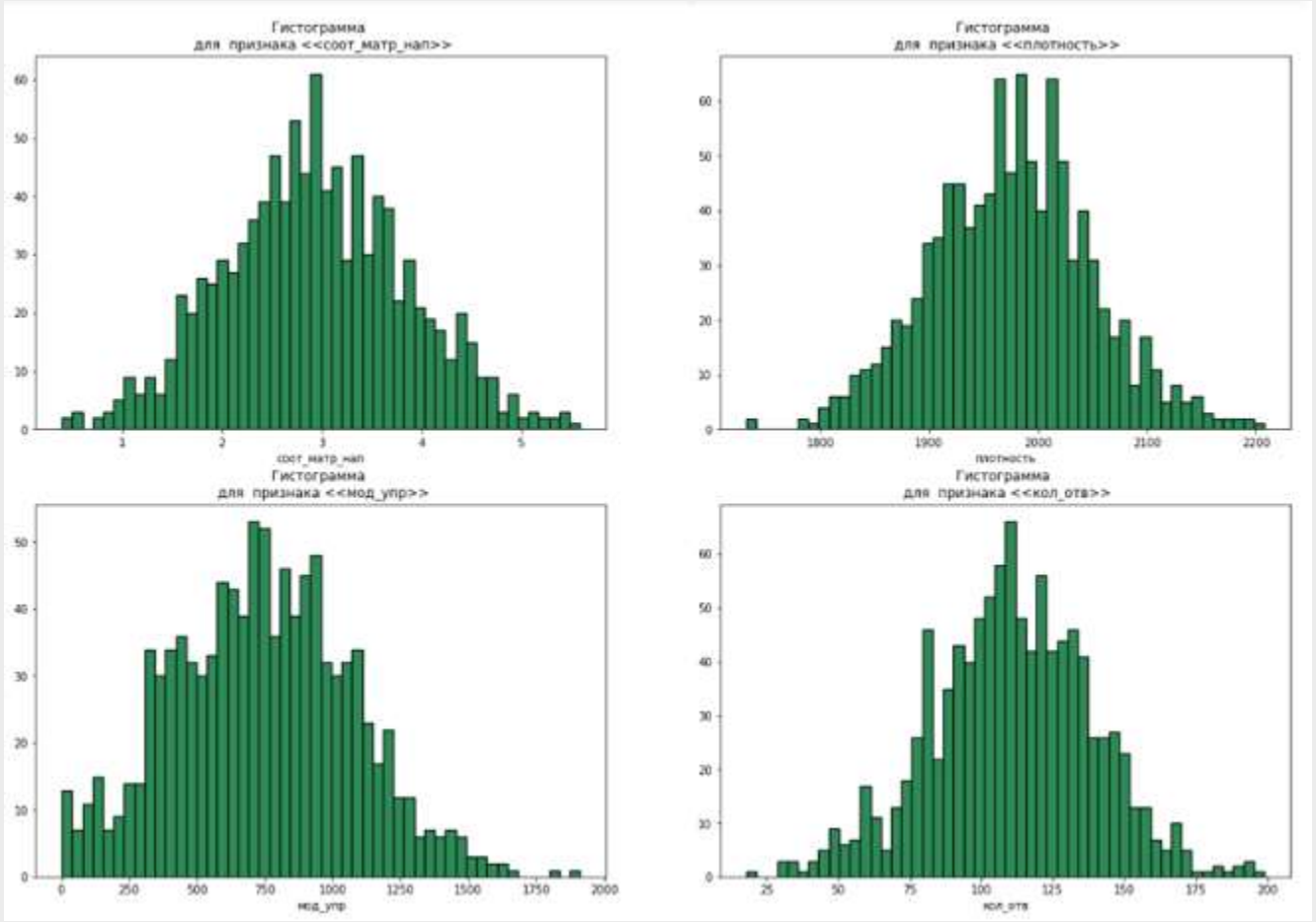
```
df.duplicated().sum()
```

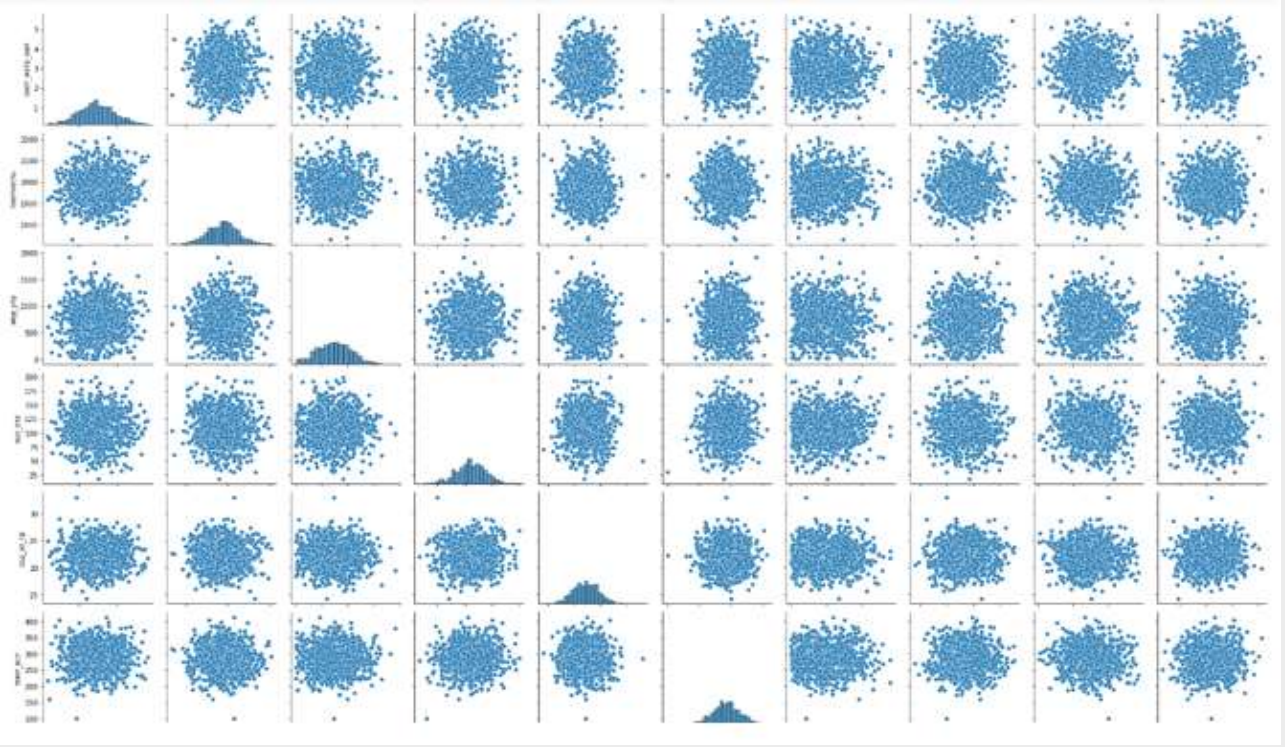
```
0
```

Визуализация

- Matplotlib
- Seaborn

- Гистограммы распределения
- Графики рассеяния
- Диаграммы ящик с усами
- Удаление выбросов
- Нормализация
- Посмотрели корреляцию



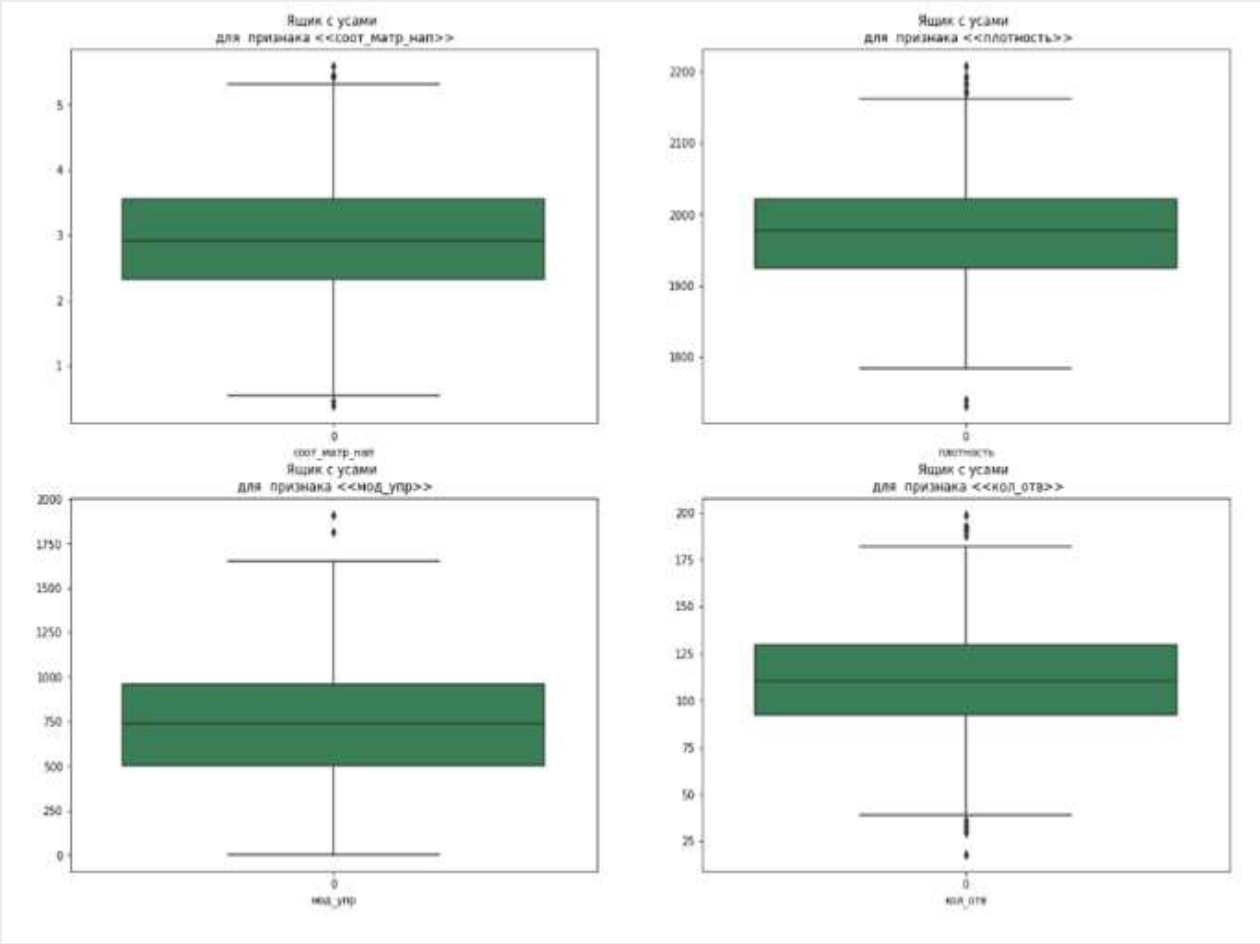


```
mms = MinMaxScaler()
data_norm = mms.fit_transform(np.array(df))
```

```
df_norm = pd.DataFrame(data = data_norm, columns =['соот_матр_нап', 'плотность', 'мод_упр', 'кол_отв', 'сод_эп_гр', 'темп_всп', 'пов_плотн', '
df_norm
```

	соот_матр_нап	плотность	мод_упр	кол_отв	сод_эп_гр	темп_всп	пов_плотн	мод_упр_раст	проч_раст	потр_смолы	угол_наш	шаг_наш	плотн_наш
0	0.282131	0.617489	0.447061	0.068506	0.596696	0.511042	0.169158	0.319194	0.698235	0.514688	0.0	0.291000	0.573774
1	0.282131	0.617489	0.447061	0.626716	0.402989	0.591181	0.169158	0.319194	0.698235	0.514688	0.0	0.364441	0.350716
2	0.457857	0.617489	0.455721	0.505606	0.481855	0.511042	0.169158	0.319194	0.698235	0.514688	0.0	0.364441	0.522299
3	0.457201	0.529218	0.452685	0.505606	0.481855	0.511042	0.169158	0.319194	0.698235	0.514688	0.0	0.364441	0.573774
4	0.419084	0.264403	0.488508	0.505606	0.481855	0.511042	0.169158	0.319194	0.698235	0.514688	0.0	0.364441	0.745356
...	...	...	...	...	...	...	...	...	...	...	...	...	...
919	0.361750	0.388242	0.552781	0.329891	0.315685	0.720233	0.168511	0.485125	0.480312	0.183151	1.0	0.663815	0.351056
920	0.587163	0.676600	0.268550	0.746710	0.275125	0.352687	0.282789	0.475992	0.470745	0.157752	1.0	0.773186	0.466548
921	0.555750	0.447928	0.251612	0.496233	0.612773	0.322515	0.597427	0.573346	0.578340	0.572648	1.0	0.302836	0.704686
922	0.637396	0.725769	0.448724	0.714320	0.247787	0.465017	0.517714	0.536217	0.368070	0.434855	1.0	0.460884	0.543937
923	0.657131	0.206771	0.251903	0.628012	0.885300	0.596144	0.612457	0.550550	0.647135	0.426577	1.0	0.443677	0.872919

924 rows × 13 columns



Обучение моделей

Линейная регрессия

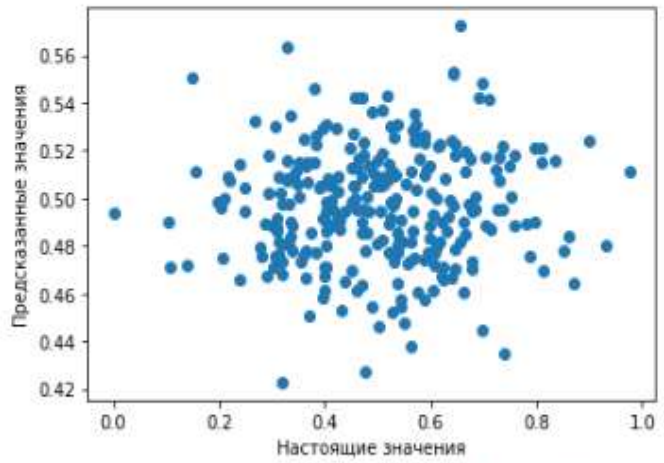
```
X = df_norm.drop(['соот_матр_нап', 'мод_упр_раст', 'проч_раст'], axis=
y_mod = df_norm[['мод_упр_раст']]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y_mod,
                                                    test_size=0.3,
                                                    random_state=0)
```

```
lr_mod = LinearRegression()
lr_mod.fit(X_train, y_train)
y_mod_pred = lr_mod.predict(X_test)
```

```
plt.scatter(y_test, y_mod_pred)
plt.xlabel('Настоящие значения')
plt.ylabel('Предсказанные значения')
plt.show()
```

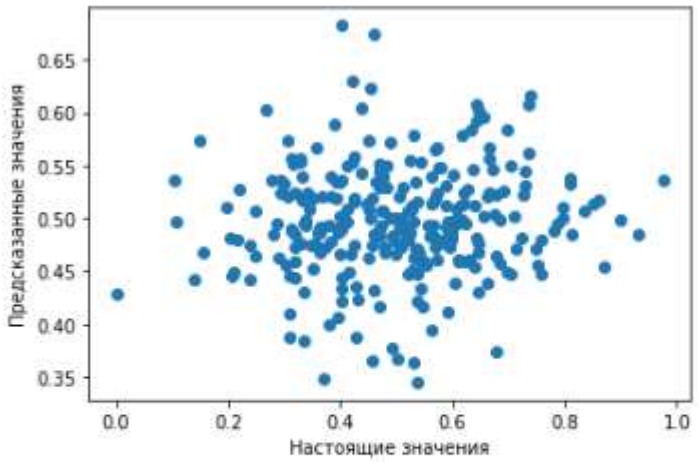
```
print('Средняя абсолютная ошибка:', mean_absolute_error(y_test, y_mod_pred))
print('Средняя квадратическая ошибка:', metrics.mean_squared_error(y_test, y_mod_pred))
print('Корень из среднеквадратичной ошибки:', np.sqrt(metrics.mean_squared_error(y_test, y_mod_pred)))
print('Train score:', lr_mod.score(X_train, y_train))
print('Test score:', lr_mod.score(X_test, y_test))
```



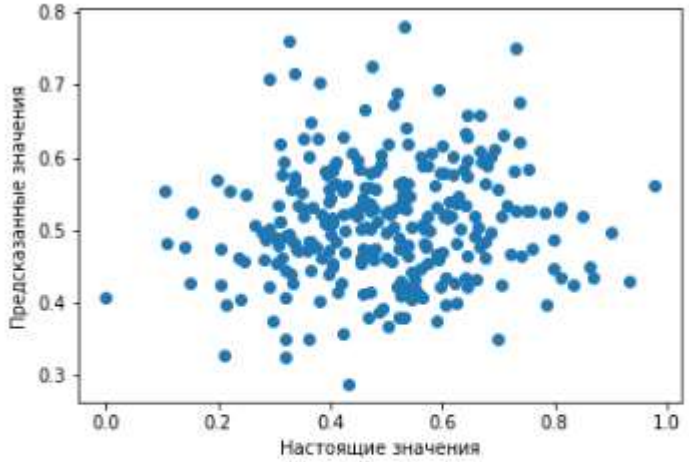

```
Средняя абсолютная ошибка: 0.1314576268724851
Средняя квадратическая ошибка: 0.02653044502216099
Корень из среднеквадратичной ошибки: 0.16288169026063362
Train score: 0.02354957271791802
Test score: -0.013409993292819111
```

Модели машинного обучения

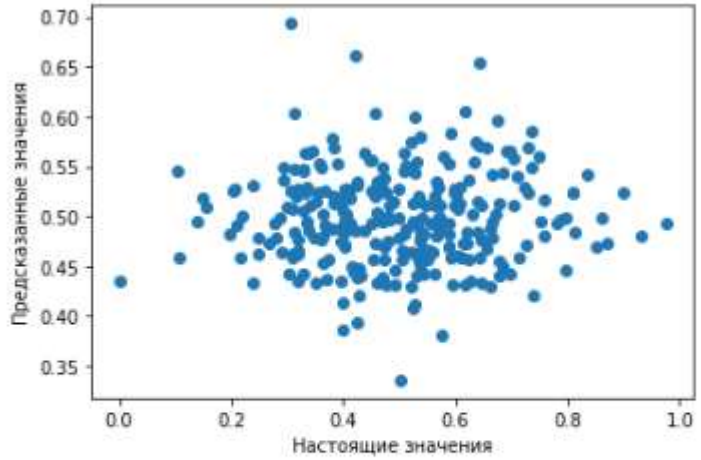
- Метод опорных векторов
- Градиентный бустинг
- Случайный лес




```
Средняя абсолютная ошибка: 0.13383360968515376
Средняя квадратическая ошибка: 0.027403045830818352
Корень из среднеквадратичной ошибки: 0.16553865358525288
Train score: 0.4906933248119718
Test score: -0.04674160076906397
```




```
Средняя абсолютная ошибка: 0.14080290512299304
Средняя квадратическая ошибка: 0.03074812596151863
Корень из среднеквадратичной ошибки: 0.17535143558442465
Train score: 0.4319282793427546
Test score: -0.17451697845252467
```




```
Средняя абсолютная ошибка: 0.13528209334394553
Средняя квадратическая ошибка: 0.028025463599794666
Корень из среднеквадратичной ошибки: 0.16740807507344044
Train score: 0.6427939276105532
Test score: -0.07051671598317877
```

Поиск гиперпараметров, MAE

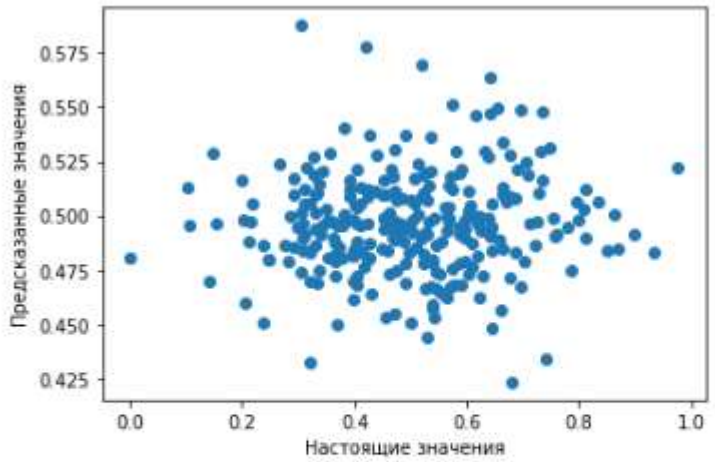
- GridSearchCV

```
parametrs = { 'n_estimators': [100, 500],
              'max_depth': [5, 20],
              'max_features': ['auto'],
              'criterion': ['mse'] }
grid = GridSearchCV(estimator = rfr_mod, param_grid = parametrs, cv = 10)
grid.fit(X_train, y_train)
```

```
GridSearchCV(cv=10,
             estimator=RandomForestRegressor(max_depth=10, n_estimators=20),
             param_grid={'criterion': ['mse'], 'max_depth': [5, 20],
                         'max_features': ['auto'], 'n_estimators': [100, 500]})
```

```
grid.best_params_
```

```
{'criterion': 'mse',
 'max_depth': 5,
 'max_features': 'auto',
 'n_estimators': 500}
```




```
Средняя абсолютная ошибка: 0.13085290669282332
Средняя квадратическая ошибка: 0.026335264811484127
Корень из среднеквадратичной ошибки: 0.16228143705145123
Train score: 0.2690318405781257
Test score: -0.00595449920564084
```

Средняя абсолютная ошибка для прогноза модуля упругости при растяжении:

- Линейная регрессия - 0,1314.
- Метод опорных векторов - 0,1408.
- Метод градиентного бустинга - 0,1338.
- Случайный лес - 0,1353.
- Случайный лес с подборкой гиперпараметров - 0,1309.

Средняя абсолютная ошибка для прогноза прочности при растяжении:

- Линейная регрессия - 0,1467.
- Метод опорных векторов - 0,1528.
- Метод градиентного бустинга - 0,1502.
- Случайный лес - 0,1484.
- Случайный лес с подборкой гиперпараметров - 0,1471.

Построение нейронной сети

```
train_dataset = df.sample(frac=0.8, random_state=0)
test_dataset = df.drop(train_dataset.index)
```

```
train_features = train_dataset.copy()
test_features = test_dataset.copy()

train_labels = train_features.pop('соот_матр_нап')
test_labels = test_features.pop('соот_матр_нап')
```

```
normalizer = tf.keras.layers.Normalization(axis=-1)
normalizer.adapt(np.array(train_features))
```

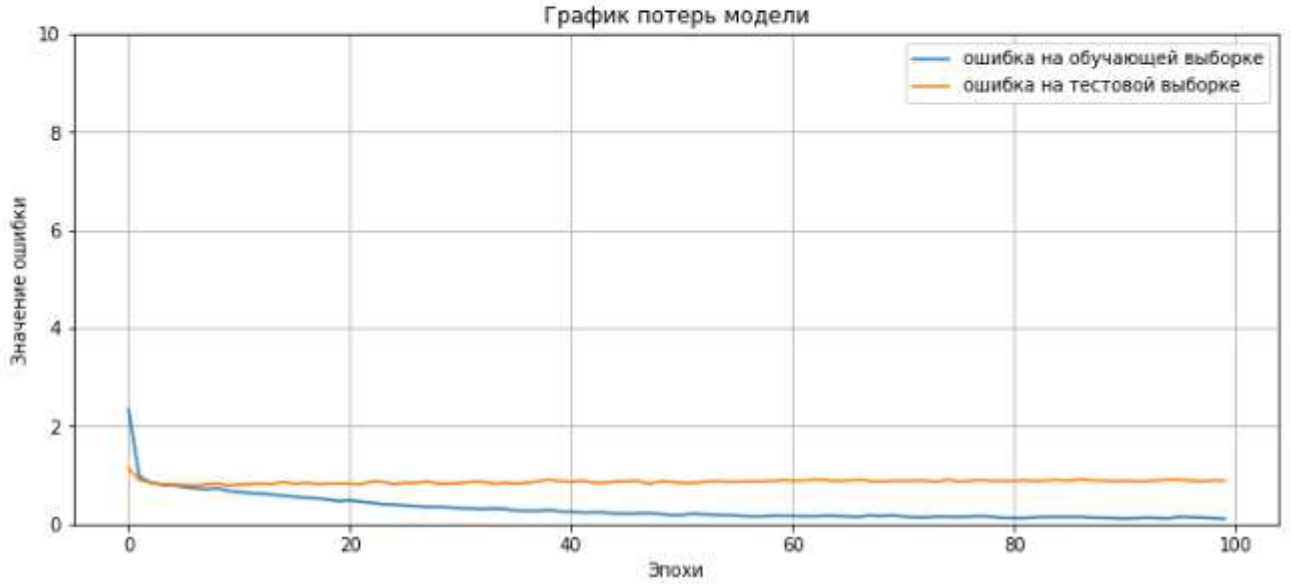
```
# Построим многослойный персептрон
def compile_model(norm):
  model = keras.Sequential([
      norm,
      layers.Dense(128, activation='relu'),
      layers.Dense(64, activation='relu'),
      layers.Dense(64, activation='relu'),
      layers.Dense(32, activation='relu'),
      layers.Dense(16, activation='relu'),
      layers.Dense(1),
])

  model.compile(loss='mean_absolute_error',
                optimizer=tf.keras.optimizers.Adam(0.001))
  return model
```

```
itog_model = compile_model(normalizer)
itog_model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_2 (Normalizat  (None, 12)               25
 ion)

 dense_12 (Dense)            (None, 128)               1664

 dense_13 (Dense)            (None, 64)                8256

 dense_14 (Dense)            (None, 64)                4160

 dense_15 (Dense)            (None, 32)                2080

 dense_16 (Dense)            (None, 16)                528

 dense_17 (Dense)            (None, 1)                 17

=================================================================
Total params: 16,730
Trainable params: 16,705
Non-trainable params: 25
_________________________________________________________________
```

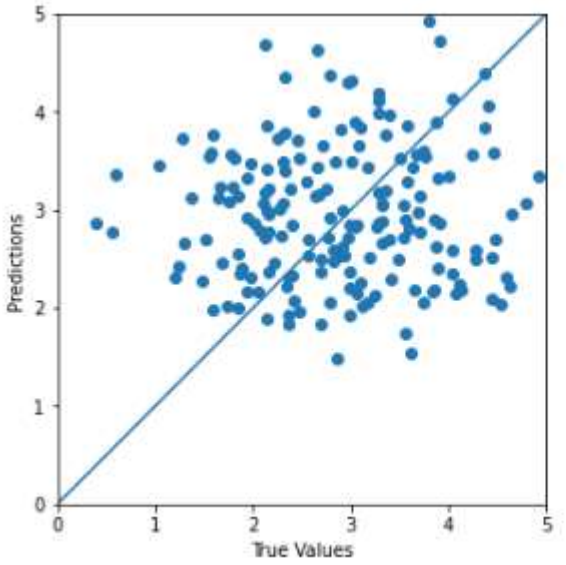
	loss	val_loss	epoch
95	0.148348	0.902362	95
96	0.139414	0.883634	96
97	0.129413	0.866384	97
98	0.119439	0.893686	98
99	0.109648	0.879466	99

```
plt.figure(figsize = (12,5))
plt.plot(history.history['loss'], label = 'ошибка на обучающей выборке')
plt.plot(history.history['val_loss'], label = 'ошибка на тестовой выборке')
plt.ylim([0, 10])
plt.title('График потерь модели')
plt.ylabel('Значение ошибки')
plt.xlabel('Эпохи')
plt.legend()
plt.grid(True)
plt.show()
```




```
test_predictions = itog_model.predict(test_features).flatten()

plt.figure(figsize = (17,5))
a = plt.axes(aspect = 'equal')
plt.scatter(test_labels, test_predictions)
plt.xlabel('True Values')
plt.ylabel('Predictions')
lims = [0, 5]
plt.xlim(lims)
plt.ylim(lims)
_ = plt.plot(lims, lims)
```

```
6/6 [==============================] - 0s 3ms/step
```

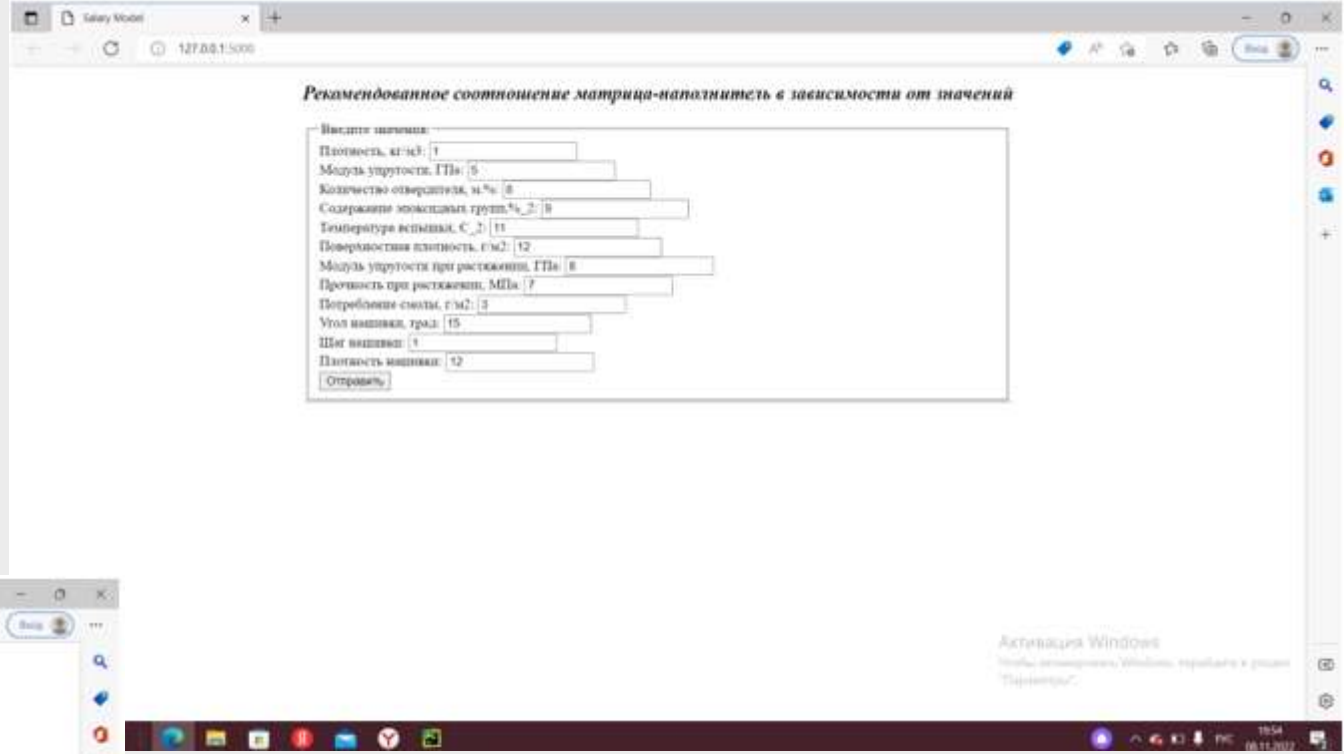


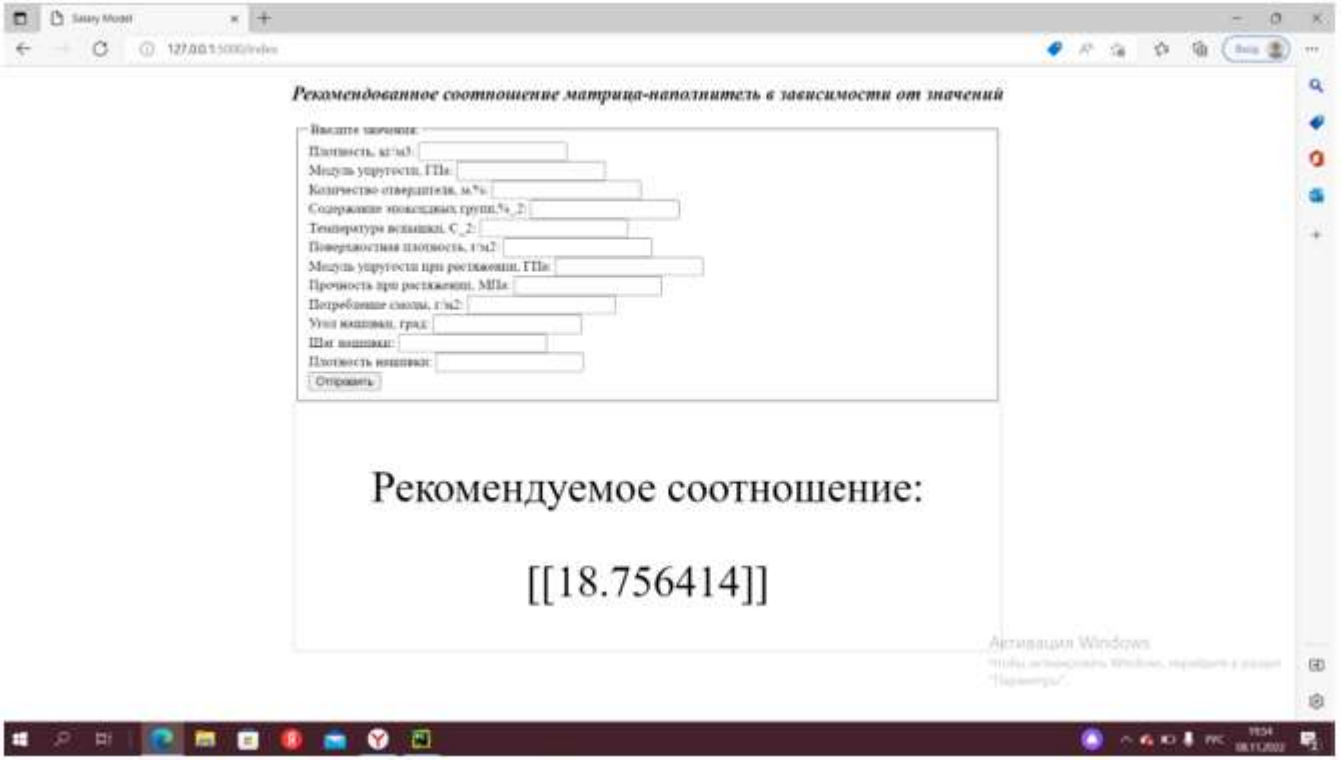
```
var = ['плотность']
mae_min = 1
for i1 in range(4):
  for i2 in range(4):
    for i3 in range(4):
      for i4 in range(4):
        for i5 in range(4):

          df_pol[var] = df[var].copy()
          df_pol[var] = i1 * (df_pol[var] ** 5) + i2 * (df_pol[var] ** 4) + i3 * (df_pol[var] ** 3) + i4 * (df_pol[var] ** 2) + i5 * df_pol[var]

          X = df_pol.drop(['мод_упр_раст'], axis=1)
          y = df_pol[['мод_упр_раст']]

          X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                  test_size=0.3,
                                                  random_state=0)

          lr_mod = LinearRegression()
          lr_mod.fit(X_train, y_train)
          y_pred = lr_mod.predict(X_test)

          mae = mean_absolute_error(y_test, y_pred)

          if mae < mae_min:
            mae_min = mae
            i1m = i1
            i2m = i2
            i3m = i3
            i4m = i4
            i5m = i5

            print('Средняя абсолютная ошибка:', mean_absolute_error(y_test, y_pred))
            print('Train score:', lr_mod.score(X_train, y_train))
            print('Test score:', lr_mod.score(X_test, y_test))

            print(i1, i2, i3, i4, i5)

df_pol[var] = df[var].copy()
df_pol[var] = i1m * (df_pol[var] ** 5) + i2m * (df_pol[var] ** 4) + i3m * (df_pol[var] ** 3) + i4m * (df_pol[var] ** 2) + i5m * df_pol[var]
df_pol
```

Признак	Коэф	MAE
плотность	0 0 0 0 1	0.1314576
мод_упр	3 0 0 0 1	0.1314098
кол_отв	2 0 0 0 0	0.1311428
сод_эп_гр	0 0 0 0 1	0.1311428
темп_всп	1 0 0 0 0	0.1309151
пов_плотн	0 0 0 0 0	0.1304085
потр_смолы	2 0 0 0 0	0.1303725
угол_наш	0 0 0 0 0	0.1302636
шаг_наш	0 0 0 0 0	0.1294901
плотн_наш	1 0 0 0 0	0.1286853

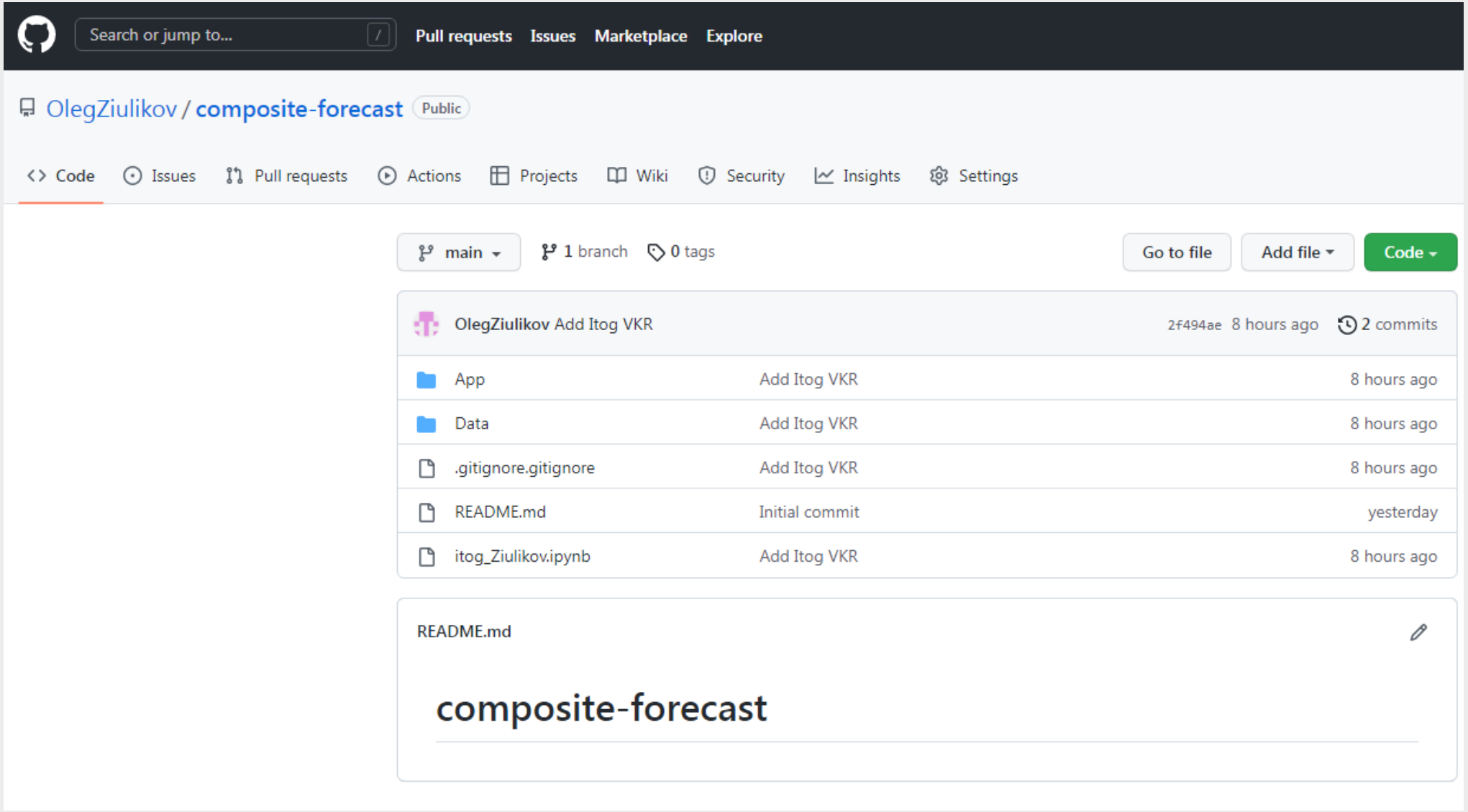
Признак	Коэф	MAE	Коэф	MAE	Коэф	MAE
плотность	0 0 0 0 1	0.1314576	0 0 0 1 2	0.1286830	0 0 0 3 1	0.1286698
мод_упр	3 0 0 0 1	0.1314098	3 0 0 0 1	0.1286830	2 0 0 0 0	0.1286660
кол_отв	2 0 0 0 0	0.1311428	1 0 0 0 0	0.1286830	2 1 0 0 0	0.1286617
сод_эп_гр	0 0 0 0 1	0.1311428	0 0 0 0 1	0.1286830	0 0 0 0 1	0.1286617
темп_всп	1 0 0 0 0	0.1309151	1 0 0 0 0	0.1286830	1 0 0 0 0	0.1286617
пов_плотн	0 0 0 0 0	0.1304085	0 0 0 0 0	0.1286830	0 0 0 0 0	0.1286617
потр_смолы	2 0 0 0 0	0.1303725	3 0 0 0 1	0.1286726	3 0 0 0 1	0.1286617
угол_наш	0 0 0 0 0	0.1302636	0 0 0 0 0	0.1286726	0 0 0 0 0	0.1286617
шаг_наш	0 0 0 0 0	0.1294901	0 0 0 0 0	0.1286726	0 0 0 0 0	0.1286617
плотн_наш	1 0 0 0 0	0.1286853	1 0 0 0 0	0.1286726	1 0 0 0 0	0.1286617
плотность	0 0 0 2 1	0.1286615	0 0 0 1 1	0.1286598		0,0028
мод_упр	1 0 0 0 0	0.1286615				
кол_отв	3 0 1 0 0	0.1286600				

Приложение на Flask





Создание удаленного репозитория



edu.bmstu.ru

+7 495 182-83-85

edu@bmstu.ru

Москва, Госпитальный переулок , д. 4-6, с.3

ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР МГТУ им. Н. Э. Баумана