ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по курсу «Data Science PRO»

ТЕМА: «ПРОГНОЗИРОВАНИЕ КОНЕЧНЫХ СВОЙСТВ НОВЫХ МАТЕРИАЛОВ (КОМПОЗИЦИОННЫХ МАТЕРИАЛОВ)»

СЛУШАТЕЛЬ:

РАСКИ ИРИНА ГЕННАДЬЕВНА

Постановка задачи

Спрогнозировать конечные свойства получаемых композиционных материалов при помощи:

алгоритмов машинного обучения, определяющих значения:

- Модуль упругости при растяжении, ГПа;
- Прочность при растяжении, МПа,

нейронной сети, которая будет рекомендовать:

- Соотношение матрица-наполнитель.

Исходные данные

Датасет со свойствами композитов сформирован путем объединения по индексу (тип - объединения INNER) двух документов в формате xslx:

Таблица «X_bp.xlsx» (1024 строки и 11 столбцов);

Таблица «X_nup.xlsx» (1041 строка и 4 столбца)

Ввод [4]:
```
# Объединение данных в датасет
dataset = dataset_1.merge(dataset_2,how="inner",on="Unnamed: 0")
dataset.drop(['Unnamed: 0'], axis=1, inplace=True)
dataset.head()
```

Out[4]:

	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, град	наш
0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0	0	
1	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0	0	
2	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0	0	
3	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0	0	
4	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0	0	

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1023 entries, 0 to 1022
Data columns (total 13 columns):
 #   Column                                 Non-Null Count  Dtype  
---  ------                                 --------------  -----  
 0   Соотношение матрица-наполнитель        1023 non-null   float64
 1   Плотность, кг/м3                       1023 non-null   float64
 2   модуль упругости, ГПа                  1023 non-null   float64
 3   Количество отвердителя, м.%            1023 non-null   float64
 4   Содержание эпоксидных групп,%_2        1023 non-null   float64
 5   Температура вспышки, С_2               1023 non-null   float64
 6   Поверхностная плотность, г/м2          1023 non-null   float64
 7   Модуль упругости при растяжении, ГПа   1023 non-null   float64
 8   Прочность при растяжении, МПа          1023 non-null   float64
 9   Потребление смолы, г/м2                1023 non-null   float64
 10  Угол нашивки, град                     1023 non-null   int64  
 11  Шаг нашивки                            1023 non-null   float64
 12  Плотность нашивки                      1023 non-null   float64
dtypes: float64(12), int64(1)
memory usage: 104.0 KB
```

Объем выборки составил 1023 строки и 13 столбцов.

Разведочный анализ и предобработка

Описательная статистика

	Количество	Среднее значение	Стандартное отклонение	Минимальное значение	25-й процентиль, 1-й квартиль	Медианное значение	75-й процентиль, 3-й квартиль	Максимальное значение
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, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
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

Проверка наличия пропусков и дубликатов

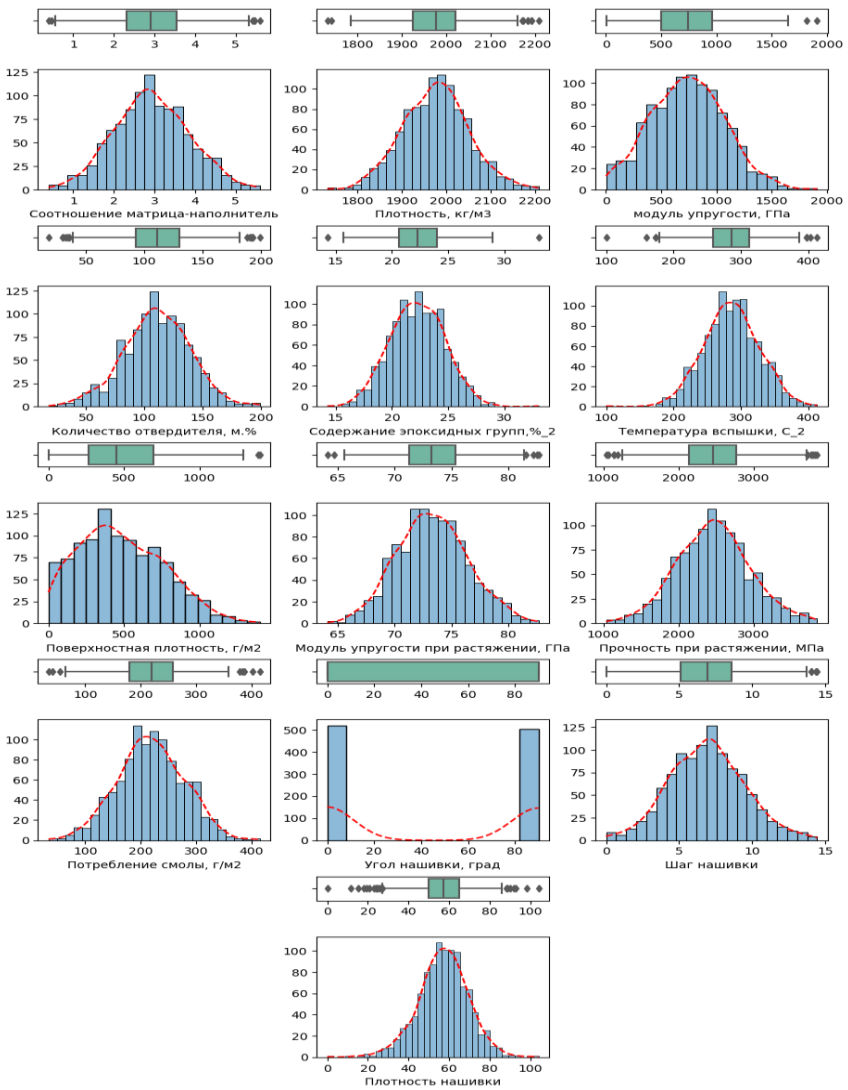
```
# Наличие дубликатов
dataset.duplicated().sum()
```

```
0
```

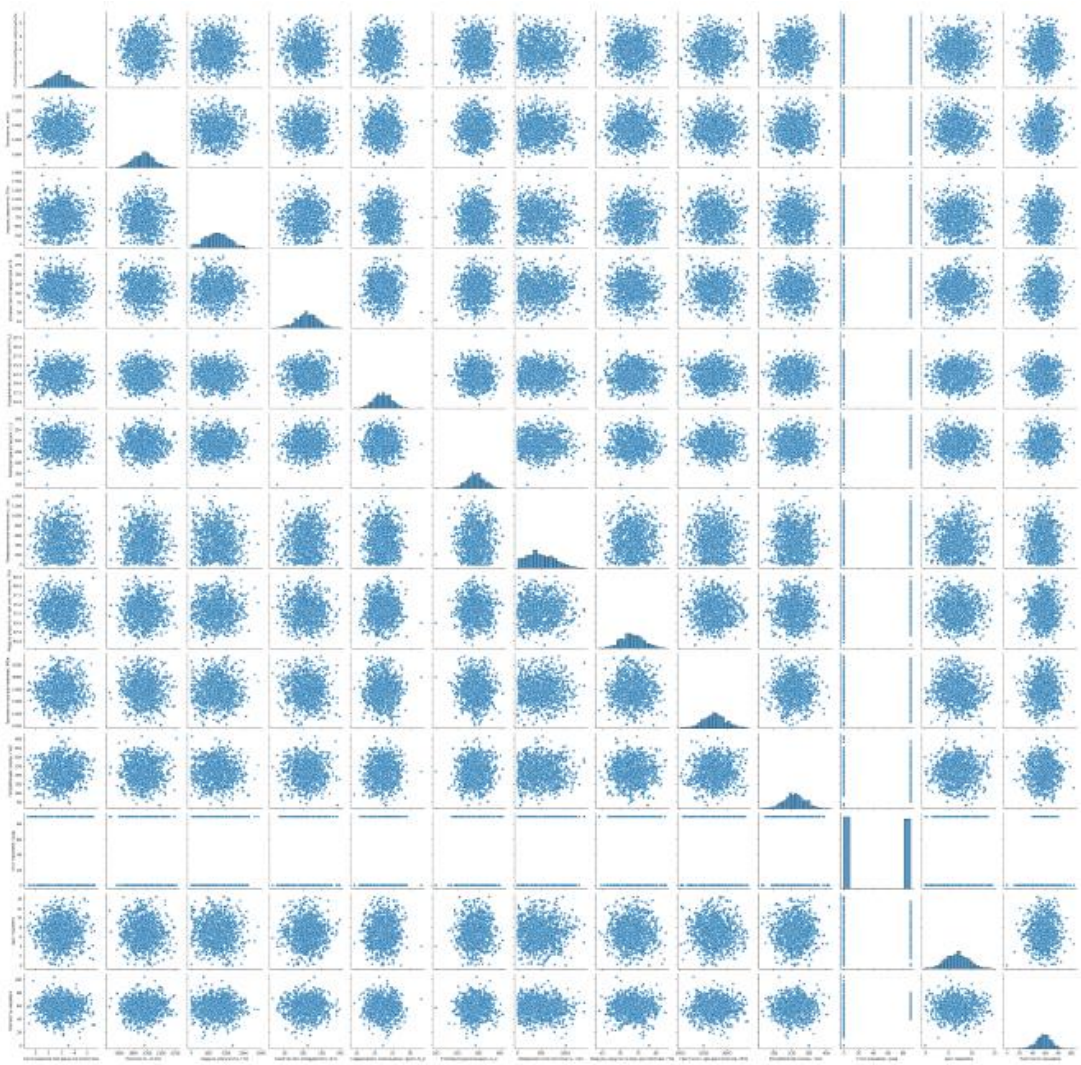
```
#пропуски
dataset.isnull().sum()
```

```
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, град                       0
Шаг нашивки                              0
Плотность нашивки                        0
dtype: int64
```

Анализ выбросов

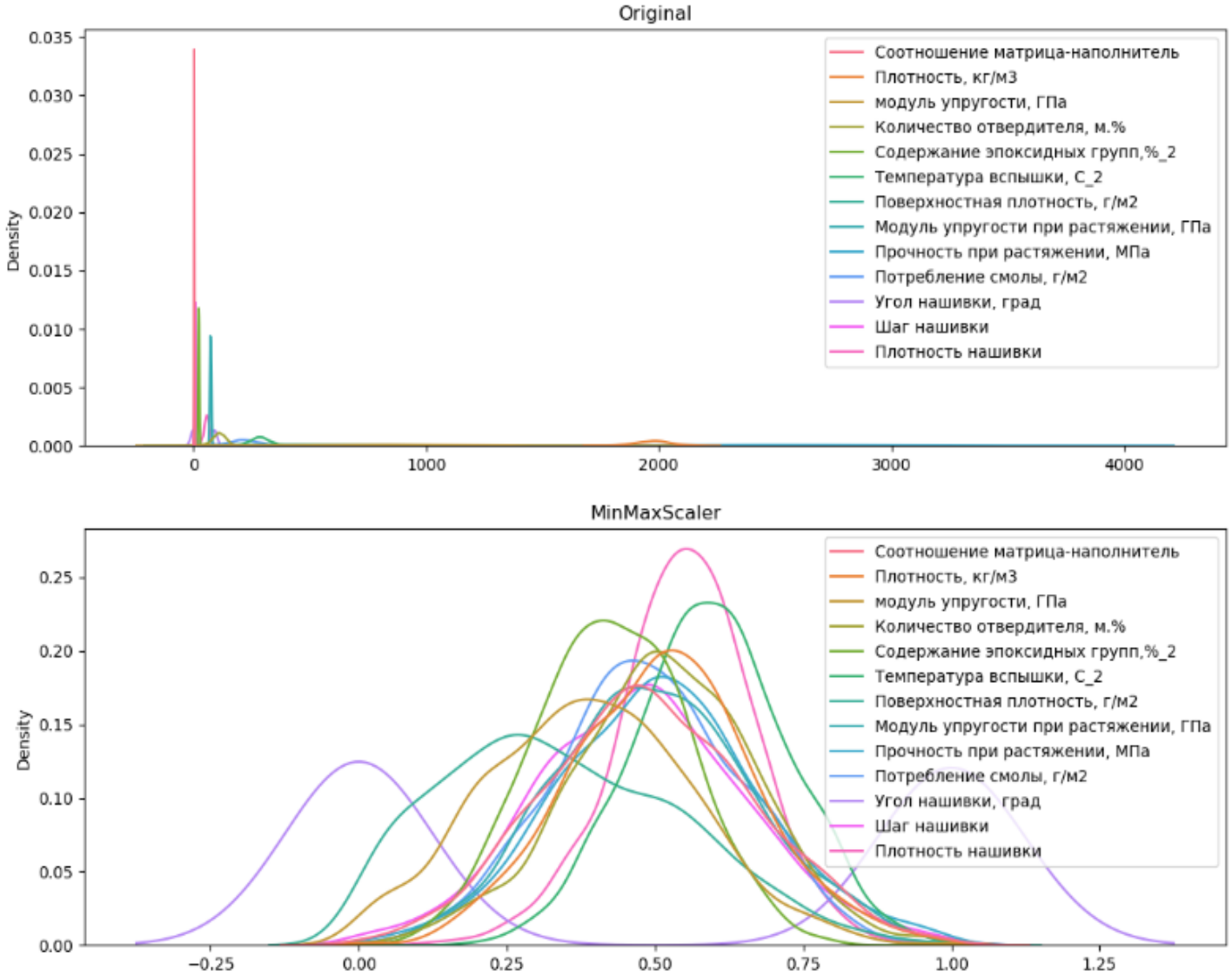


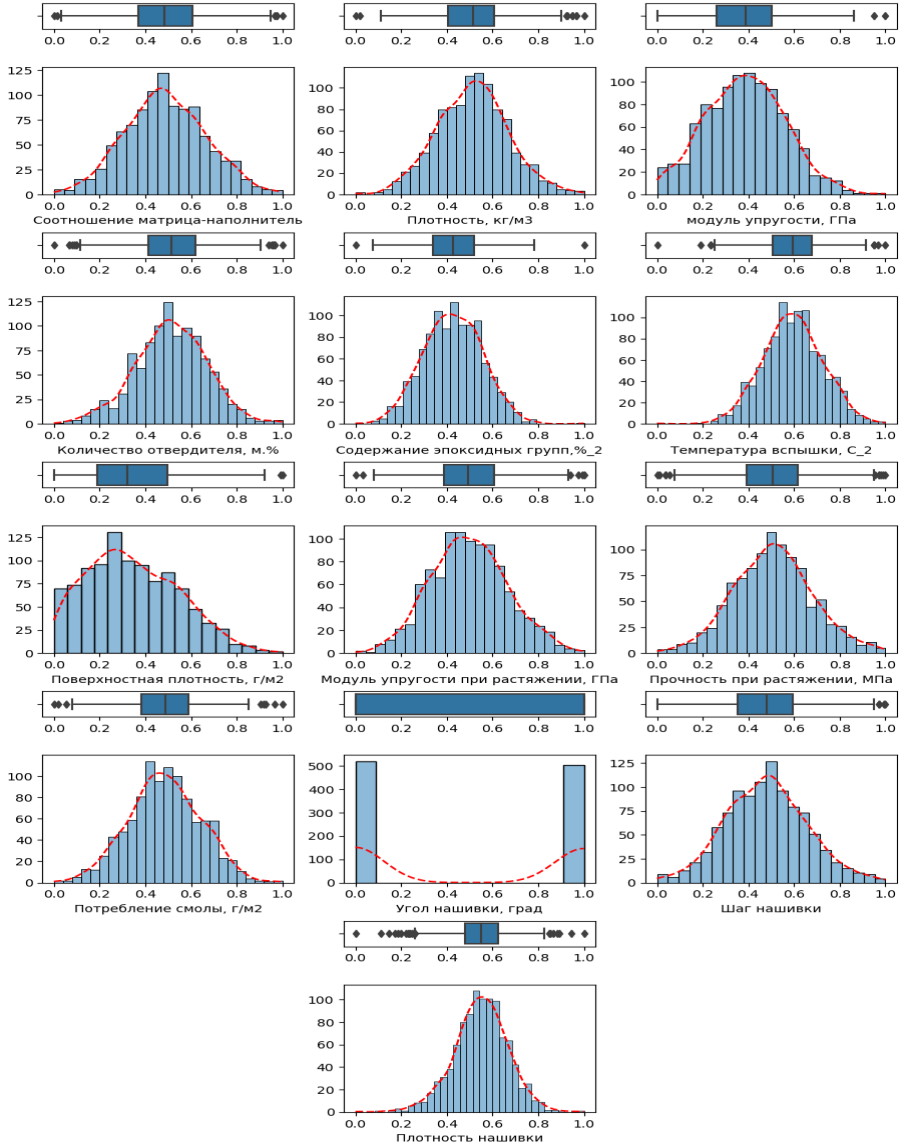
Гистограммы распределения каждой из переменной, диаграммы ящика с усами



Попарные графиков рассеяния точек

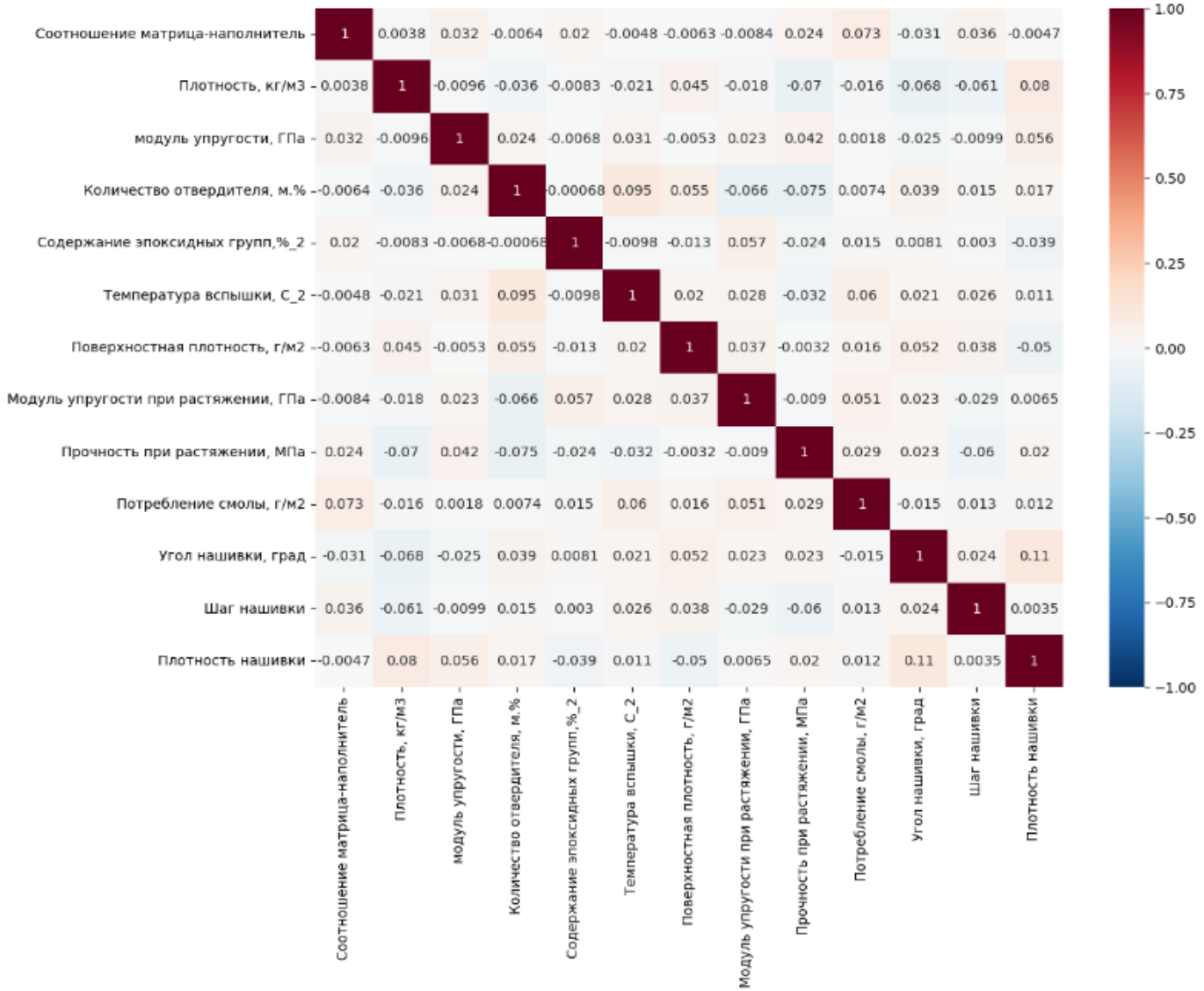
Преобразование данных (нормализация)





Гистограммы распределения каждой из переменной, диаграммы ящика с усами (после нормализации)

Корреляционный анализ



Параметры датасета не имеют чётко выраженной зависимости, корреляция между всеми параметрами слабая.

Функции для поиска и удаления выбросов

```python
# поиск выбросов
def outliers_search(dataset):
    outlier_list = list()
    result_outliers = pd.DataFrame()
    for col in dataset:
        result_outliers['z_score'] = (((dataset[col] - dataset[col].mean()) / dataset[col].std()).abs()>3)
        outliers_z = result_outliers['z_score'].sum()

        IQR = dataset[col].quantile(0.75) - dataset[col].quantile(0.25)
        lower_column = dataset[col].quantile(0.25) - 1.5 * IQR
        upper_column = dataset[col].quantile(0.75) + 1.5 * IQR
        result_outliers['IQR'] = (dataset[col] <= lower_column) | (dataset[col] >= upper_column)
        outliers_IQR = result_outliers['IQR'].sum()

        fig, ax = plt.subplots(1, 2, figsize=(16, 4))
        plt.suptitle(f'{col}:', fontsize = 13)
        sns.histplot(data=result_outliers, x=dataset[col], hue='z_score', multiple='stack', bins=50, legend=False, ax=ax[0],
                     palette='Set2').set_title(f'Сумма выбросов по методу Z-score = {outliers_z}')
        sns.histplot(data=result_outliers, x=dataset[col], hue='IQR', multiple='stack', bins=50, legend=False, ax=ax[1],
                     palette='Set2').set_title(f'Сумма выбросов по методу IQR = {outliers_IQR}')
        ax[0].set(xlabel='')
        ax[1].set(xlabel='')
        outlier_list.append({'': col, 'Метод стандартных отклонений(Z-score)': outliers_z,
                             'Метод межквартильных расстояний (IQR)': outliers_IQR})
    print('Число обнаруженных выбросов:')
    outliers = pd.DataFrame(outlier_list).set_index('')
    outliers.loc['ИТОГО']= outliers[outliers.columns].sum()
    display(outliers)
```

```python
# поиск и удаление выбросов по Методу стандартных отклонений (Z-score)
def outliers_zscore(dataset):
    outliers_all = {'Метод стандартных отклонений (Z-score)': 0,
                    'Выбросы после очистки': 0,       }
    outlier_list = list()
    result_outliers = pd.DataFrame()
    for col in dataset:

        result_outliers['z_score'] = (((dataset[col] - dataset[col].mean()) / dataset[col].std()).abs()>3)
        outliers_z = result_outliers['z_score'].sum()
        outliers_all['Метод стандартных отклонений (Z-score)'] += outliers_z
    dataset[(np.abs(zscore(dataset)) > 3).any(axis = 1)]=np.nan
    dataset_clean=dataset.dropna()

    for col in dataset_clean:
        result_outliers['z_score'] = (((dataset_clean[col] - dataset_clean[col].mean()) / dataset_clean[col].std()).abs()>3)
        outliers_z = result_outliers['z_score'].sum()
        outliers_all['Выбросы после очистки'] += outliers_z
    print('Число обнаруженных выбросов:')
    for key, val in outliers_all.items():
        print(f'{key} - {val}')
    print(f'Размер очищенного датасета: {dataset_clean.shape}')
    return dataset_clean
```

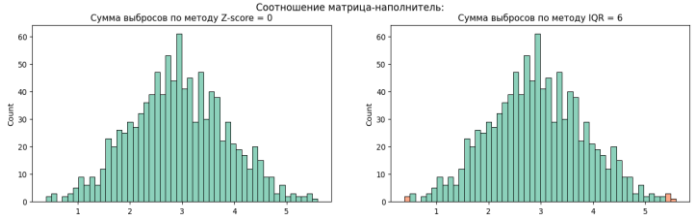
```python
# поиск и удаление выбросов по Методу межквартильных расстояний (IQR)
def outliers_IQR(dataset):
    outliers_all = {'Метод межквартильных расстояний (IQR)': 0,
                    'Выбросы после очистки': 0,       }
    outlier_list = list()
    result_outliers = pd.DataFrame()
    for col in dataset:
        IQR = dataset[col].quantile(0.75) - dataset[col].quantile(0.25)
        lower_column = dataset[col].quantile(0.25) - 1.5 * IQR
        upper_column = dataset[col].quantile(0.75) + 1.5 * IQR
        result_outliers['IQR'] = (dataset[col] <= lower_column) | (dataset[col] >= upper_column)
        outliers_IQR = result_outliers['IQR'].sum()
        outliers_all['Метод межквартильных расстояний (IQR)'] += outliers_IQR

    IQR = dataset[dataset.columns].quantile(0.75) - dataset[dataset.columns].quantile(0.25)
    lower_column = dataset[dataset.columns].quantile(0.25) - 1.5 * IQR
    upper_column = dataset[dataset.columns].quantile(0.75) + 1.5 * IQR
    dataset[((dataset[dataset.columns] <= lower_column) | (dataset[dataset.columns] >= upper_column))]=np.nan
    dataset_clean=dataset.dropna()

    for col in dataset_clean:
        IQR = dataset_clean[col].quantile(0.75) - dataset_clean[col].quantile(0.25)
        lower_column = dataset_clean[col].quantile(0.25) - 1.5 * IQR
        upper_column = dataset_clean[col].quantile(0.75) + 1.5 * IQR
        result_outliers['IQR'] = (dataset_clean[col] <= lower_column) | (dataset_clean[col] >= upper_column)
        outliers_IQR = result_outliers['IQR'].sum()
        outliers_all['Выбросы после очистки'] += outliers_IQR

    print('Число обнаруженных выбросов:')
    for key, val in outliers_all.items():
        print(f'{key} - {val}')
    print(f'Размер очищенного датасета: {dataset_clean.shape}')
    return dataset_clean
```

Поиск и удаление выбросов

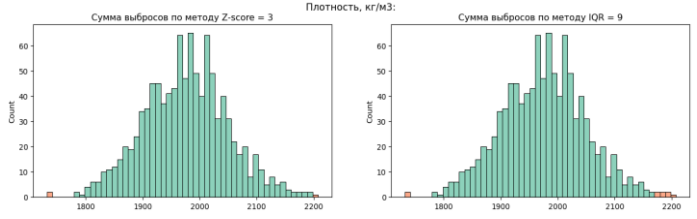
Число обнаруженных выбросов:

	Метод стандартных отклонений(Z-score)	Метод межквартильных расстояний (IQR)
Соотношение матрица-наполнитель	0	6
Плотность, кг/м3	3	9
модуль упругости, ГПа	2	2
Количество отвердителя, м.%	2	14
Содержание эпоксидных групп,%_2	2	2
Температура вспышки, С_2	3	8
Поверхностная плотность, г/м2	2	2
Модуль упругости при растяжении, ГПа	0	6
Прочность при растяжении, МПа	0	11
Потребление смолы, г/м2	3	8
Угол нашивки, град	0	0
Шаг нашивки	0	4
Плотность нашивки	7	21
ИТОГО	24	93

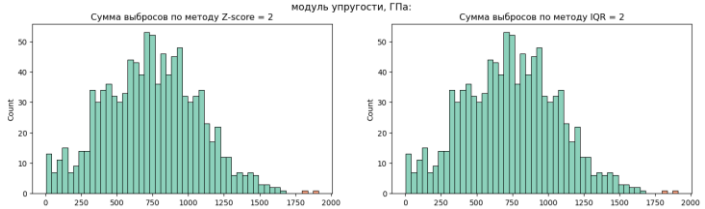
Соотношение матрица-наполнитель:



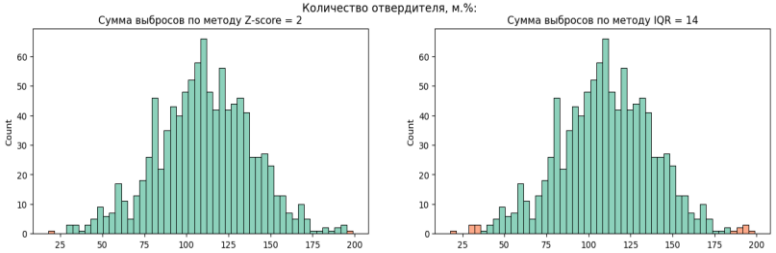
Плотность, кг/м3:



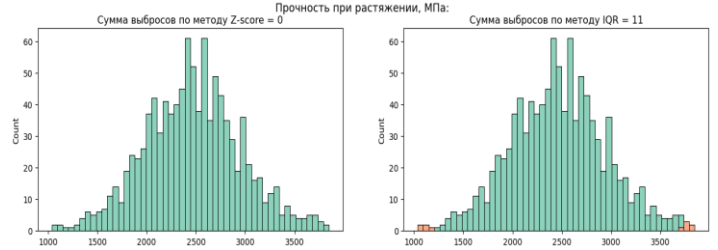
модуль упругости, ГПа:



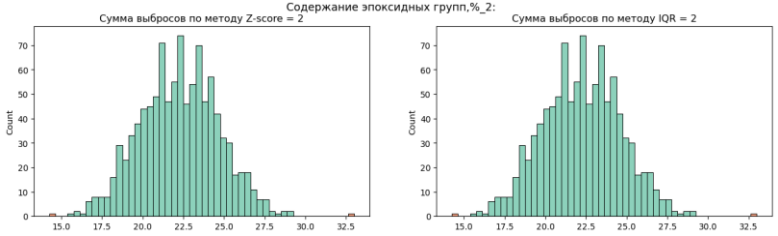
Количество отвердителя, м.%:



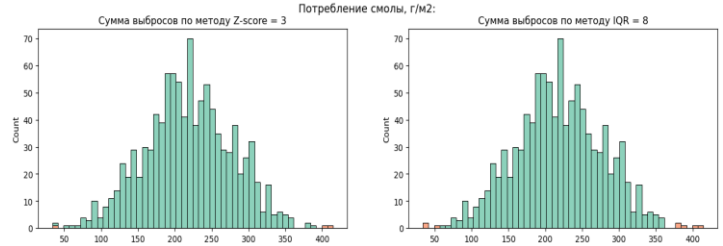
Прочность при растяжении, МПа:



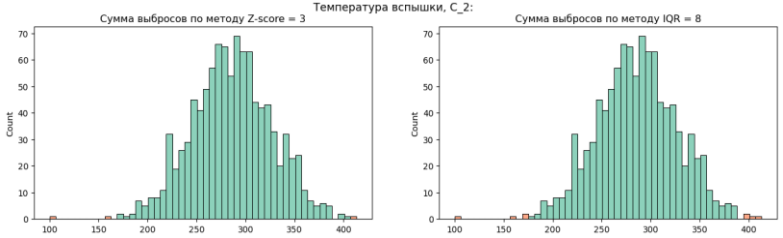
Содержание эпоксидных групп,%_2:



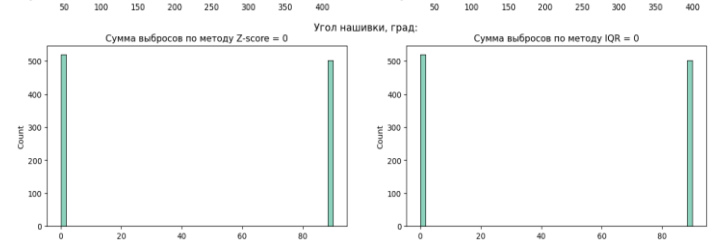
Потребление смолы, г/м2:



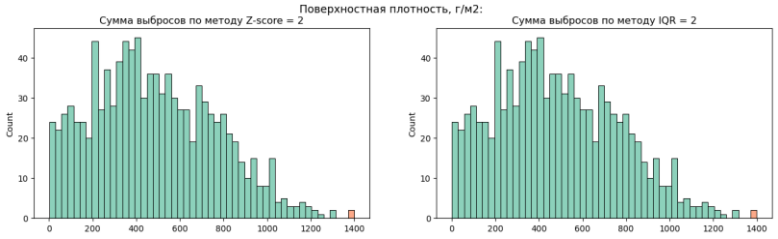
Температура вспышки, С_2:



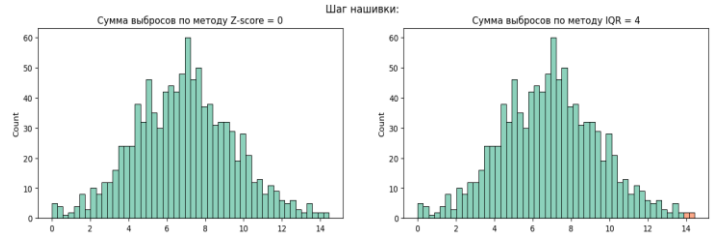
Угол нашивки, град:



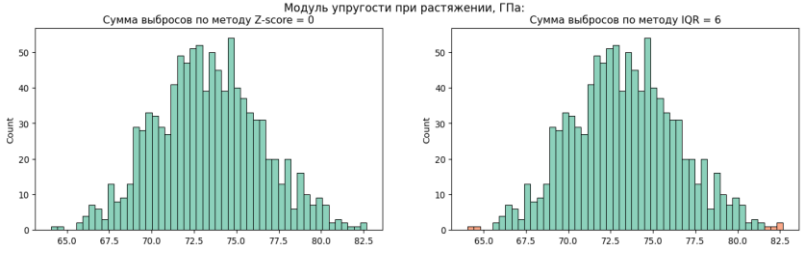
Поверхностная плотность, г/м2:



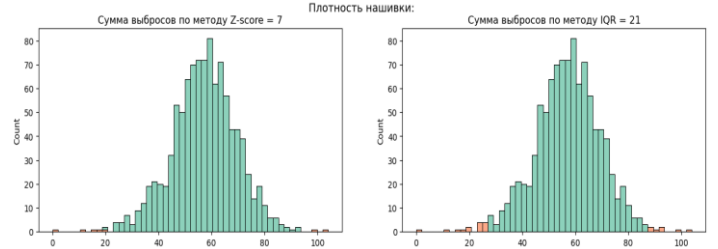
Шаг нашивки:



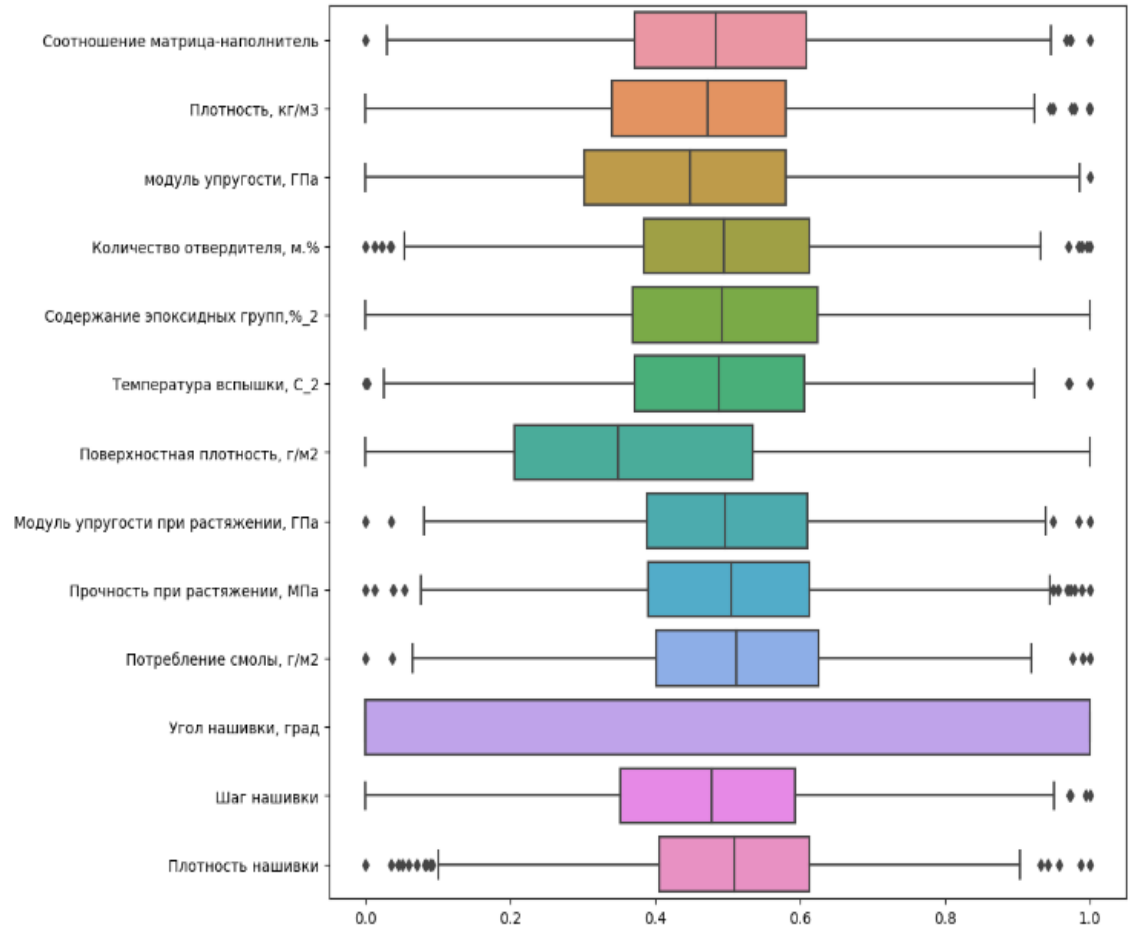
Модуль упругости при растяжении, ГПа:



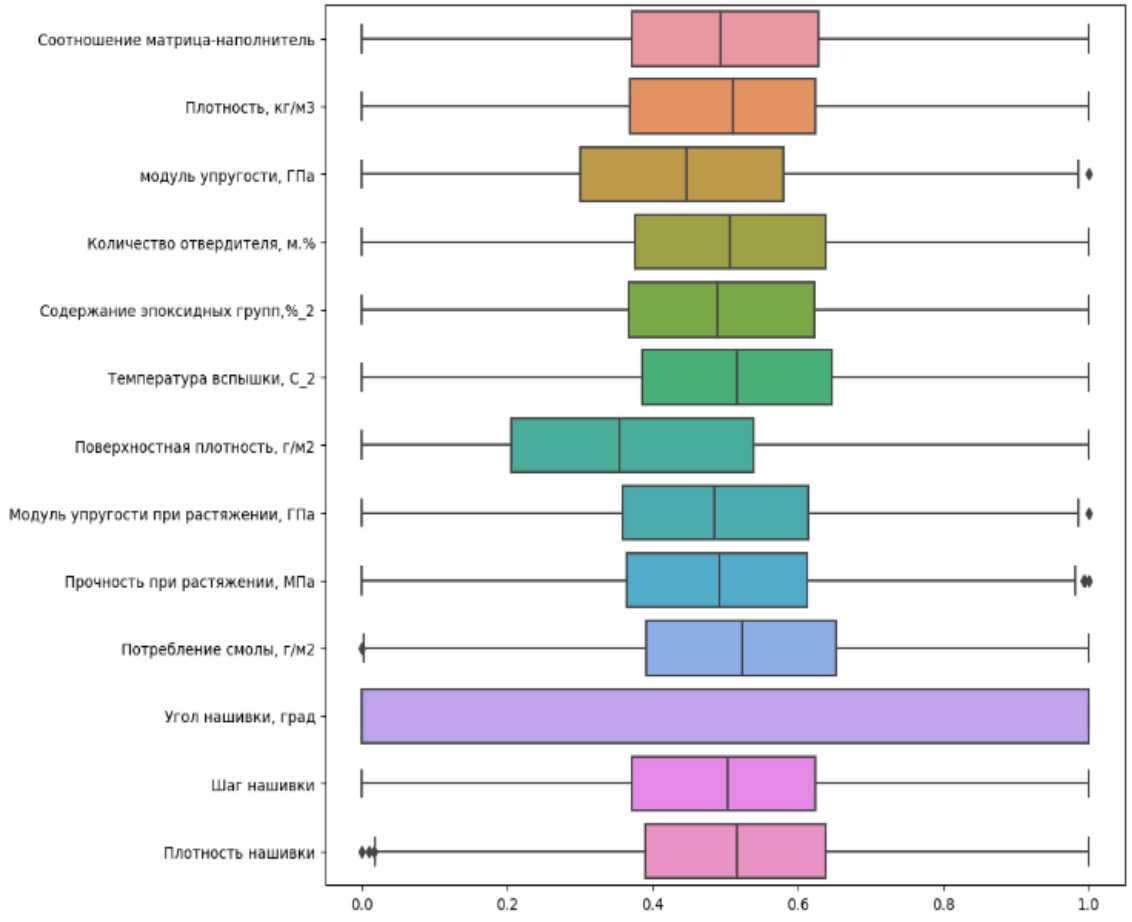
Плотность нашивки:



Число обнаруженных выбросов:
Метод стандартных отклонений (Z-score) - 24
Выбросы после очистки - 3
Размер очищенного датасета: (999, 13)



Число обнаруженных выбросов:
Метод межквартильных расстояний (IQR) - 93
Выбросы после очистки - 10
Размер очищенного датасета: (936, 13)



Разработка и обучение модели

Разработка и обучение моделей машинного обучения производились для двух параметров: «Модуль упругости при растяжении» и «Прочность при растяжении».

Для прогноза каждого из параметров использованы следующие модели:

- Линейная регрессия (LinearRegression);
- К-ближайших соседей (KNeighborsRegressor);
- Метод опорных векторов (SVR);
- Регрессия дерева решений (DecisionTreeRegressor);
- Случайный лес (RandomForestRegressor),
- Градиентный бустинг (GradientBoostingRegressor),
- Стохастический градиентный спуск (SGDRegressor),
- Нейронная сеть.

Выборка по каждому из параметров была разделена на обучающую (70%) и тестовую (30%).

Для оценки качества модели использовались следующие метрики:

- коэффициент детерминации (R2);
- средняя абсолютная ошибка (MAE);
- среднеквадратичная ошибка (MSE);
- корень из среднеквадратичной ошибки (RMSE);
- средняя абсолютная ошибка в процентах (MAPE).

Тестирование модели

В рамках тестирования модели проведено сравнение расчетов показателей на датасете:

- без удаления выбросов;
- с удалением выбросов по методу стандартных отклонений (Z-score);
- с удалением выбросов по методу межквартильных расстояний (IQR).

Разработка и обучение модели для показателя «Модуль упругости при растяжении, ГПа».

Обучение модели без удаления выбросов

Модели для прогноза показателя "Модуль упругости при растяжении, ГПа" и результаты обучения

Размер обучающей выборки: 716
Размер тестовой выборки: 307

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.000s	-0.02	2.56	10.12	3.18	0.03
KNeighborsRegressor	0.000s	0.016s	-0.27	2.83	12.66	3.56	0.04
SVR	0.024s	0.020s	-0.05	2.59	10.44	3.23	0.04
DecisionTreeRegressor	0.031s	0.000s	-1.14	3.67	21.25	4.61	0.05
RandomForestRegressor	2.067s	0.000s	-0.09	2.62	10.83	3.29	0.04
GradientBoostingRegressor	0.729s	0.000s	-0.11	2.65	10.98	3.31	0.04
SGDRegressor	0.069s	0.000s	-0.05	2.61	10.45	3.23	0.04

Результаты кросс-валидации

	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.04	2.51	9.76	3.12	0.03
KNeighborsRegressor	-0.24	2.72	11.67	3.41	0.04
SVR	-0.05	2.51	9.86	3.13	0.03
DecisionTreeRegressor	-1.26	3.67	20.79	4.56	0.05
RandomForestRegressor	-0.09	2.53	10.11	3.17	0.03
GradientBoostingRegressor	-0.13	2.61	10.71	3.27	0.04
SGDRegressor	-0.11	2.60	10.39	3.22	0.04

CPU times: total: 2min 12s
Wall time: 2min 14s

Обучение модели с удалением выбросов по методу стандартных отклонений (Z-score)

Модели для прогноза показателя "Модуль упругости при растяжении, ГПа" и результаты обучения

Размер обучающей выборки: 699
Размер тестовой выборки: 300

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.000s	-0.03	2.52	9.55	3.09	0.03
KNeighborsRegressor	0.000s	0.016s	-0.14	2.65	10.63	3.26	0.04
SVR	0.027s	0.020s	-0.05	2.53	9.73	3.12	0.03
DecisionTreeRegressor	0.038s	0.000s	-1.29	3.73	21.28	4.61	0.05
RandomForestRegressor	2.126s	0.016s	-0.08	2.56	10.03	3.17	0.04
GradientBoostingRegressor	0.722s	0.000s	-0.12	2.60	10.38	3.22	0.04
SGDRegressor	0.050s	0.000s	-0.10	2.62	10.20	3.19	0.04

Результаты кросс-валидации

	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.05	2.52	9.96	3.14	0.03
KNeighborsRegressor	-0.28	2.78	11.95	3.45	0.04
SVR	-0.07	2.53	10.06	3.16	0.03
DecisionTreeRegressor	-1.41	3.83	23.11	4.70	0.05
RandomForestRegressor	-0.11	2.59	10.57	3.22	0.03
GradientBoostingRegressor	-0.19	2.64	11.19	3.33	0.04
SGDRegressor	-0.10	2.56	10.42	3.22	0.04

CPU times: total: 2min 12s
Wall time: 2min 17s

Обучение модели с удалением выбросов по методу межквартильных расстояний (IQR)

Модели для прогноза показателя "Модуль упругости при растяжении, ГПа" и результаты обучения

Размер обучающей выборки: 655
Размер тестовой выборки: 281

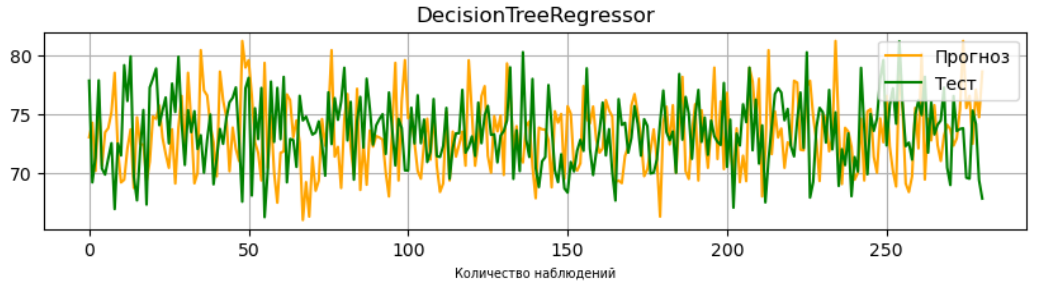
	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.003s	0.002s	-0.01	2.41	8.76	2.96	0.03
KNeighborsRegressor	0.000s	0.000s	-0.21	2.64	10.53	3.24	0.04
SVR	0.016s	0.031s	-0.03	2.44	8.95	2.99	0.03
DecisionTreeRegressor	0.011s	0.016s	-1.26	3.61	19.74	4.44	0.05
RandomForestRegressor	1.962s	0.016s	0.00	2.40	8.71	2.95	0.03
GradientBoostingRegressor	0.691s	0.000s	-0.11	2.49	9.64	3.10	0.03
SGDRegressor	0.050s	0.000s	-0.05	2.47	9.13	3.02	0.03

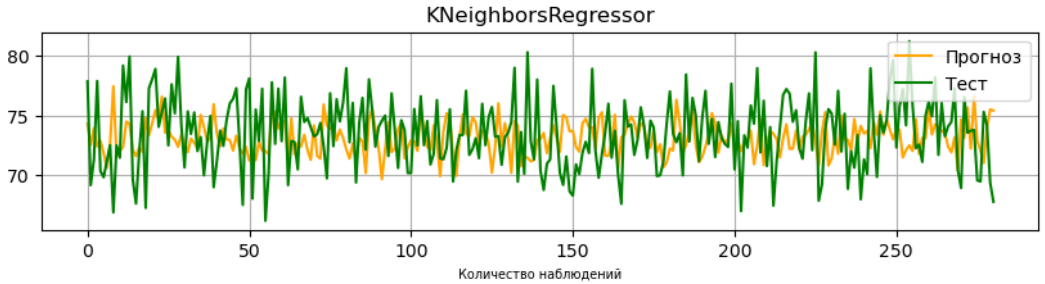
Результаты кросс-валидации

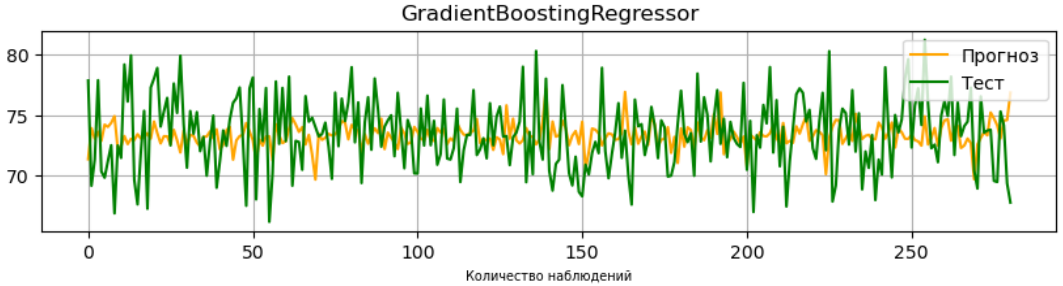
	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.04	2.50	9.69	3.10	0.03
KNeighborsRegressor	-0.27	2.75	11.75	3.42	0.04
SVR	-0.05	2.51	9.74	3.11	0.03
DecisionTreeRegressor	-1.21	3.56	20.51	4.51	0.05
RandomForestRegressor	-0.08	2.55	10.09	3.20	0.03
GradientBoostingRegressor	-0.14	2.59	10.58	3.24	0.04
SGDRegressor	-0.09	2.53	10.14	3.17	0.03

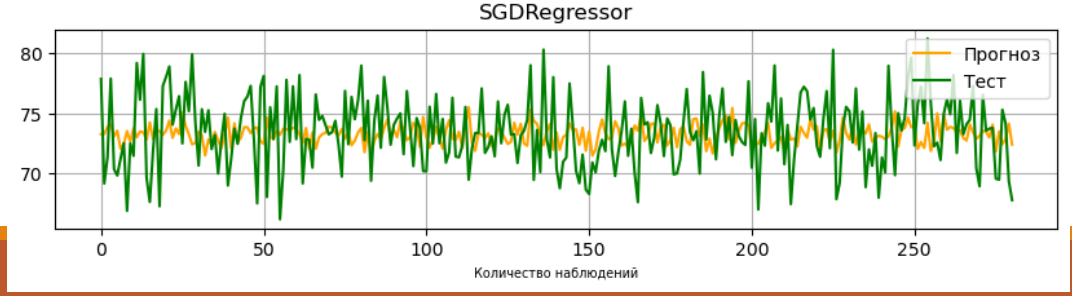
CPU times: total: 2min
Wall time: 2min 7s

Графики обучения моделей для параметра «Модуль упругости при растяжении, ГПа» с удалением выбросов по методу межквартильных расстояний (IQR)







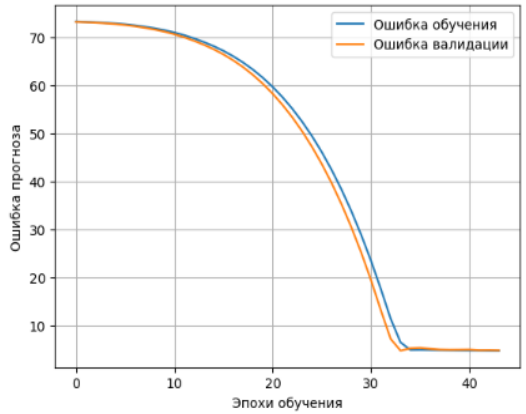


Применение нейронной сети с удалением выбросов по методу межквартильных расстояний (IQR)

```
# Применение Нейронной сети с выбросами iqr
y = dataset_clean_iqr['Модуль упругости при растяжении, ГПа']
x = dataset_clean_iqr.drop(columns=['Модуль упругости при растяжении, ГПа'])

X_train, X_test, y_train, y_test = train_test_split(x, y,
                                                    test_size=0.3,
                                                    random_state=42)

normalizer = Normalization(input_shape=[12,], axis=None)
normalizer.adapt(np.array(X_train))
```

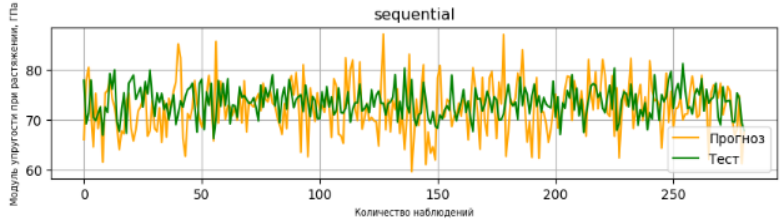
```
y_test значение 359    77.825677
Name: Модуль упругости при растяжении, ГПа, dtype: float64
Предсказательное значение:[[66.05327]]
```

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential	3.458s	0.153s	-3.19	4.77	36.54	6.04	0.06



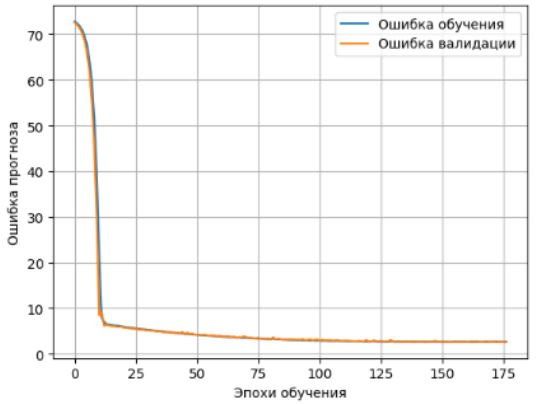

```
model_1 = Sequential([
    normalizer,
    Dense(8, activation='relu'),
    Dense(6, activation='relu'),
    Dense(3, activation='relu'),
    Dense(1)
  ])
model_1.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization (Normalizati  (None, 12)                3
 on)

 dense (Dense)               (None, 8)                 104

 dense_1 (Dense)             (None, 6)                 54

 dense_2 (Dense)             (None, 3)                 21

 dense_3 (Dense)             (None, 1)                 4

=================================================================
Total params: 186 (748.00 Byte)
Trainable params: 183 (732.00 Byte)
Non-trainable params: 3 (16.00 Byte)
_________________________________________________________________
```



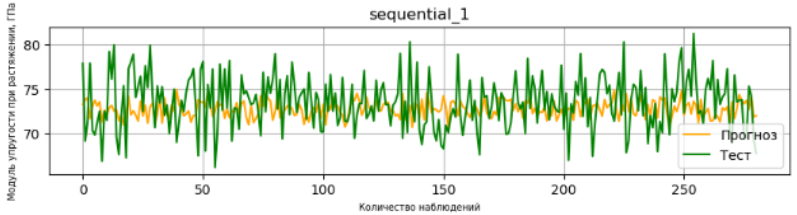

```
y_test значение 359    77.825677
Name: Модуль упругости при растяжении, ГПа, dtype: float64
Предсказательное значение:[[73.26266]]
```

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential	3.458s	0.153s	-3.19	4.77	36.54	6.04	0.06
sequential_1	11.402s	0.153s	-0.19	2.63	10.41	3.23	0.04




```
model_2 = Sequential([
    normalizer,
    Dense(3, activation='relu'),
    Dense(3, activation='linear'),
    Dense(3, activation='linear'),
    Dense(3, activation='linear'),
    Dense(3, activation='linear'),
    Dense(3, activation='linear'),
    Dense(1)
  ])
model_2.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization (Normalizati  (None, 12)                3
 on)

 dense_4 (Dense)             (None, 3)                 39

 dense_5 (Dense)             (None, 3)                 12

 dense_6 (Dense)             (None, 3)                 12

 dense_7 (Dense)             (None, 3)                 12

 dense_8 (Dense)             (None, 3)                 12

 dense_9 (Dense)             (None, 3)                 12

 dense_10 (Dense)            (None, 1)                 4

=================================================================
Total params: 106 (428.00 Byte)
Trainable params: 103 (412.00 Byte)
Non-trainable params: 3 (16.00 Byte)
_________________________________________________________________
```



Разработка и обучение модели для показателя «Прочность при растяжении, МПа».

Обучение модели без удаления выбросов

Модели для прогноза показателя "Прочность при растяжении, МПа" и результаты обучения

Размер обучающей выборки: 716
Размер тестовой выборки: 307

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.003s	0.002s	0.00	379.79	222990.58	472.22	0.17
KNeighborsRegressor	0.000s	0.000s	-0.19	404.98	265667.50	515.43	0.18
SVR	0.031s	0.007s	-0.00	379.60	224399.41	473.71	0.17
DecisionTreeRegressor	0.031s	0.000s	-0.93	515.15	432718.64	657.81	0.22
RandomForestRegressor	2.241s	0.000s	-0.01	378.11	225407.65	474.77	0.17
GradientBoostingRegressor	0.745s	0.002s	-0.07	394.54	239473.91	489.36	0.17
SGDRegressor	0.016s	0.000s	-0.08	392.61	240967.79	490.88	0.17

Результаты кросс-валидации

	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.03	388.92	245120.25	494.34	0.17
KNeighborsRegressor	-0.21	426.88	288409.72	535.62	0.19
SVR	-0.01	384.82	241019.60	490.41	0.17
DecisionTreeRegressor	-1.14	572.07	520415.05	713.63	0.24
RandomForestRegressor	-0.03	392.50	252834.30	499.81	0.17
GradientBoostingRegressor	-0.10	406.80	263665.68	512.17	0.18
SGDRegressor	-0.07	396.93	254849.57	503.89	0.17

CPU times: total: 2min 15s
Wall time: 2min 22s

Обучение модели с удалением выбросов по методу стандартных отклонений (Z-score)

Модели для прогноза показателя "Прочность при растяжении, МПа" и результаты обучения

Размер обучающей выборки: 699
Размер тестовой выборки: 300

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.000s	-0.01	380.40	227787.17	477.27	0.17
KNeighborsRegressor	0.000s	0.000s	-0.27	423.31	288330.43	535.10	0.19
SVR	0.031s	0.016s	-0.00	375.44	225299.04	474.66	0.17
DecisionTreeRegressor	0.038s	0.000s	-0.92	523.08	433110.65	658.11	0.23
RandomForestRegressor	2.162s	0.000s	-0.04	390.18	234911.56	484.68	0.17
GradientBoostingRegressor	0.712s	0.000s	-0.09	392.72	245790.53	495.77	0.17
SGDRegressor	0.031s	0.000s	-0.01	379.60	228514.44	478.03	0.17

Результаты кросс-валидации

	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.03	389.74	242072.85	490.01	0.17
KNeighborsRegressor	-0.20	417.82	283102.33	529.71	0.18
SVR	-0.01	384.50	238186.74	485.75	0.17
DecisionTreeRegressor	-1.06	554.30	468836.33	693.17	0.24
RandomForestRegressor	-0.06	399.92	250864.45	497.26	0.17
GradientBoostingRegressor	-0.09	400.33	257358.39	504.02	0.17
SGDRegressor	-0.06	396.24	253310.16	495.94	0.17

CPU times: total: 2min 11s
Wall time: 2min 19s

Обучение модели с удалением выбросов по методу межквартильных расстояний (IQR)

Модели для прогноза показателя "Прочность при растяжении, МПа" и результаты обучения

Размер обучающей выборки: 655
Размер тестовой выборки: 281

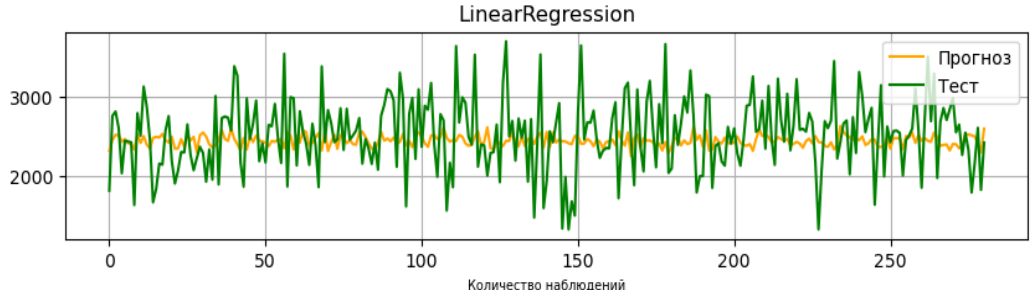
	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.008s	-0.05	363.14	213699.15	462.28	0.15
KNeighborsRegressor	0.016s	0.000s	-0.21	387.41	247988.21	497.98	0.16
SVR	0.016s	0.016s	-0.04	361.46	211321.11	459.70	0.15
DecisionTreeRegressor	0.038s	0.000s	-1.37	554.87	483205.11	695.13	0.23
RandomForestRegressor	1.902s	0.018s	-0.10	374.40	224713.09	474.04	0.15
GradientBoostingRegressor	0.696s	0.000s	-0.13	381.93	229672.12	479.24	0.16
SGDRegressor	0.007s	0.000s	-0.09	369.19	221973.99	471.14	0.15

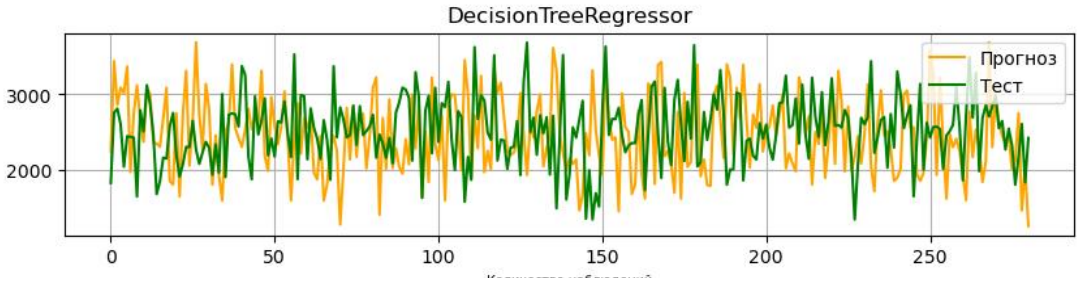
Результаты кросс-валидации

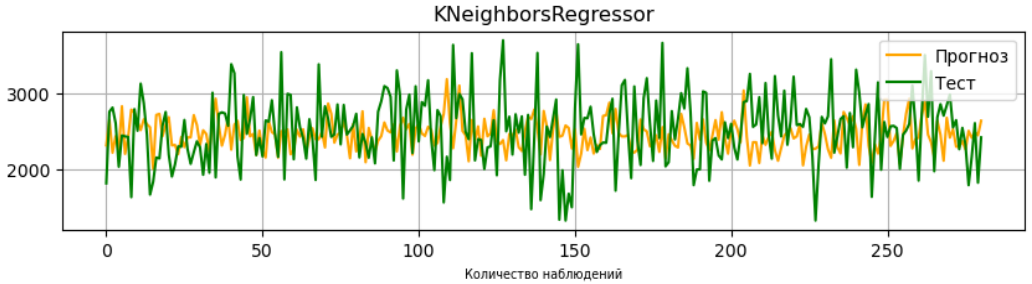
	R2	MAE	MSE	RMSE	MAPE
LinearRegression	-0.02	374.93	219961.89	467.74	0.16
KNeighborsRegressor	-0.22	410.77	261977.56	510.81	0.18
SVR	-0.01	372.65	217943.05	465.94	0.16
DecisionTreeRegressor	-1.08	523.09	422603.88	658.96	0.23
RandomForestRegressor	-0.05	383.07	229260.14	477.17	0.17
GradientBoostingRegressor	-0.11	391.14	240134.74	488.43	0.17
SGDRegressor	-0.07	384.83	226419.70	474.80	0.17

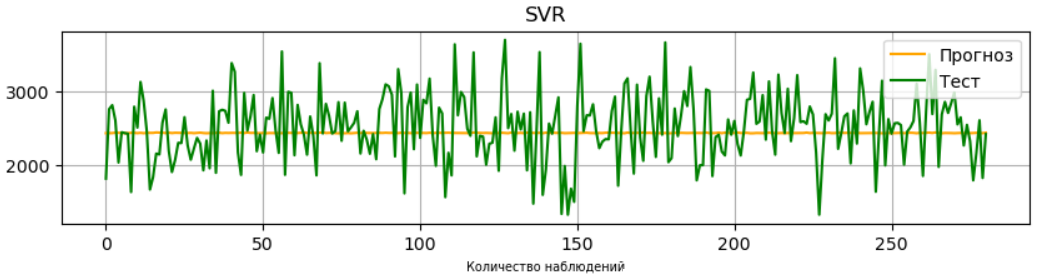
CPU times: total: 1min 58s
Wall time: 2min 2s

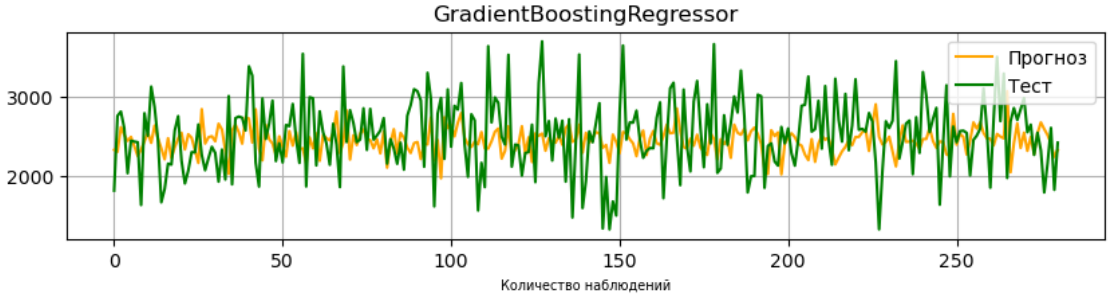
Графики обучения моделей для параметра «Прочность при растяжении, МПа» с удалением выбросов по методу межквартильных расстояний (IQR)

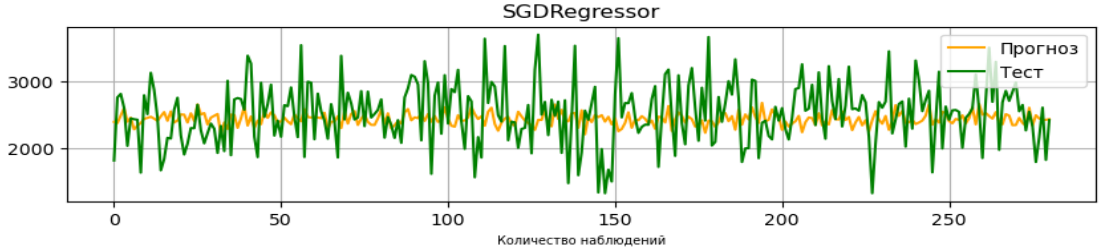












Поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой

```
# гиперпараметры
parameters = {
        'LinearRegression()':
               {'fit_intercept': [True, False],
                'n_jobs': [None, 1, 2],
                },
        'KNeighborsRegressor()':
               {'n_neighbors': range(44, 50),
                'weights': ['uniform', 'distance'],
                },
        'SVR()':
               {'kernel':['linear', 'poly', 'rbf'],
                'C': [0.01, 0.1, 0.9],
                },
    'DecisionTreeRegressor()':
               {'criterion':['squared_error', 'friedman_mse', 'absolute_error'],
                'splitter': ['best','random'],
                'max_depth':[2, 3, None],
                'max_features':[7, 8, 'sqrt', 'log2', None],
                'ccp_alpha': [0.0, 0.001, 0.01, 0.1],
               },
      'RandomForestRegressor()':
               {'n_estimators':[50, 90, 100, 150, 200],
                'max_depth':[None,2,5,10,12],
                'max_features':[3, 7, 9, 12, 21, 'sqrt', 'log2', None],
               },
       'GradientBoostingRegressor()':
               {'n_estimators':[30, 35, 40],
                'max_depth':[3, 7, 9],
                },
       'SGDRegressor()':
               {'penalty':['l2', 'l1'],
                'learning_rate':['constant', 'optimal'],
                },
        }
```

```
# Гиперпараметрическая оптимизации модели методом поиска по сетке.
def grid_search(x, y, regressor, best_models):
    name = y.name
    print(f'\033[1mГиперпараметрическая оптимизации модели методом поиска по сетке показателя "{name}"\n')
    X_train, X_test, y_train, y_test = train_test_split(x, y,
                                                test_size=0.3,
                                                random_state=42)
    print(f'\033[0mРазмер обучающей выборки: {X_train.shape[0]}')
    print(f'\033[0mРазмер тестовой выборки: {X_test.shape[0]}\n')
    grid_result= pd.DataFrame()
    params = {}

    scaler = MinMaxScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)

    for param_name, param in parameters.items():
        if param_name == str(regressor):
            for score_name, score in score_list.items():
                kfold = KFold(n_splits=10, random_state=42, shuffle=True)
                grid = GridSearchCV(regressor, param_grid=param, scoring=score, cv=kfold, n_jobs=-1)
                result = grid.fit(X_train, y_train)
                params[score_name] = result.best_estimator_
                grid_result.loc[param_name,['Best parameters']] = ';  '.join(f"{key}:{value}" for key,value in params.items())
                grid_result.loc[param_name, score_name]= '{:.2f}'.format(abs(result.best_score_) if score_name !="R2" else result
                best_models[param_name]= [val for key, val in params.items() if key =='MAE']
    print(best_models)
    display(grid_result.style.format(precision=10))
```

```
Гиперпараметрическая оптимизации модели методом поиска по сетке показателя "Прочность при растяжении, МПа"

Размер обучающей выборки: 655
Размер тестовой выборки: 281

{'LinearRegression()': [LinearRegression()]}
```

	Best parameters	R2	MAE	MSE	RMSE	MAPE
LinearRegression()	R2:LinearRegression(); MAE:LinearRegression(); MSE:LinearRegression(); RMSE:LinearRegression(); MAPE:LinearRegression()	-0.02	374.93	219961.89	467.74	0.16

	Best parameters	R2	MAE	MSE	RMSE	MAPE
KNeighborsRegressor()	R2:KNeighborsRegressor(n_neighbors=49); MAE:KNeighborsRegressor(n_neighbors=49); MSE:KNeighborsRegressor(n_neighbors=49); RMSE:KNeighborsRegressor(n_neighbors=49); MAPE:KNeighborsRegressor(n_neighbors=49)	-0.03	375.73	221140.35	469.47	0.16

```
CPU times: total: 422 ms
Wall time: 933 ms
```

SVR()	R2:SVR(C=0.9, kernel='poly'); MAE:SVR(C=0.9, kernel='poly'); MSE:SVR(C=0.9); RMSE:SVR(C=0.9, kernel='poly'); MAPE:SVR(C=0.9, kernel='poly')	-0.01	372.51	217949.40	466.86	0.16

	Best parameters	R2	MAE	MSE	RMSE	MAPE
DecisionTreeRegressor()	R2:DecisionTreeRegressor(ccp_alpha=0.1, max_depth=2, max_features='sqrt'); MAE:DecisionTreeRegressor(criterion='friedman_mse', max_depth=2, max_features='sqrt'); MSE:DecisionTreeRegressor(criterion='absolute_error', max_depth=2, max_features=7, splitter='random'); RMSE:DecisionTreeRegressor(max_depth=2, splitter='random'); MAPE:DecisionTreeRegressor(ccp_alpha=0.01, criterion='absolute_error', max_depth=3, max_features=7, splitter='random')	-0.00	371.07	214843.98	461.39	0.16

	Best parameters	R2	MAE	MSE	RMSE	MAPE
RandomForestRegressor()	R2:RandomForestRegressor(max_depth=2, max_features=3); MAE:RandomForestRegressor(max_depth=2, max_features=3, n_estimators=50); MSE:RandomForestRegressor(max_depth=2, max_features=3); RMSE:RandomForestRegressor(max_depth=2, max_features='sqrt', n_estimators=150); MAPE:RandomForestRegressor(max_depth=5, max_features=7, n_estimators=50)	-0.00	371.66	215797.58	464.00	0.16

```
CPU times: total: 20.9 s
Wall time: 41min 31s
```

	Best parameters	R2	MAE	MSE	RMSE	MAPE
GradientBoostingRegressor()	R2:GradientBoostingRegressor(n_estimators=30); MAE:GradientBoostingRegressor(n_estimators=30); MSE:GradientBoostingRegressor(n_estimators=30); RMSE:GradientBoostingRegressor(n_estimators=30); MAPE:GradientBoostingRegressor(n_estimators=30)	-0.04	377.41	224967.40	473.12	0.16

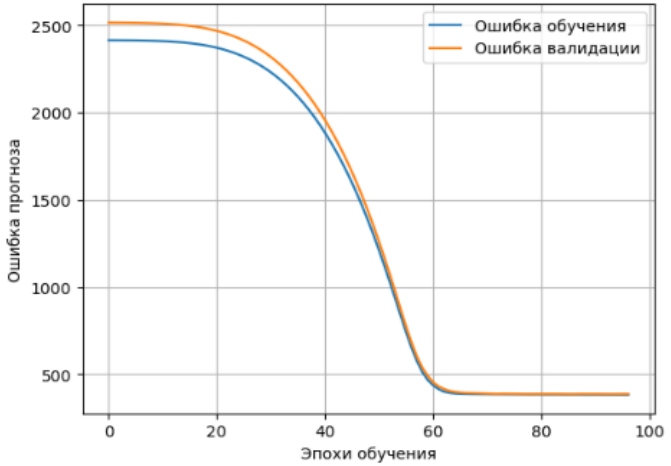
	Best parameters	R2	MAE	MSE	RMSE	MAPE
SGDRegressor()	R2:SGDRegressor(learning_rate='constant'); MAE:SGDRegressor(learning_rate='constant', penalty='l1'); MSE:SGDRegressor(learning_rate='constant'); RMSE:SGDRegressor(learning_rate='constant'); MAPE:SGDRegressor(learning_rate='constant', penalty='l1')	-0.04	375.96	225967.73	472.72	0.16

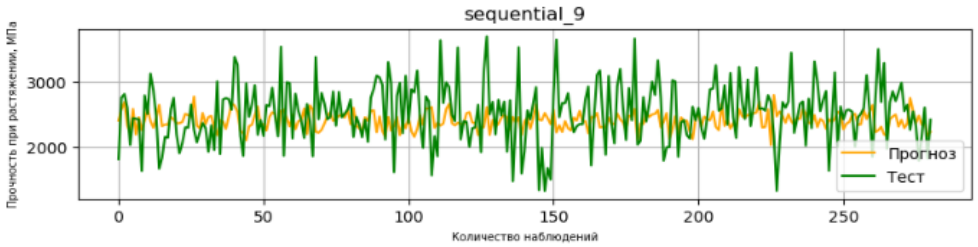
```
CPU times: total: 266 ms
Wall time: 432 ms
```

Применение нейронной сети с удалением выбросов по методу межквартильных расстояний (IQR)

```
y_test значение 359    1823.256512
Name: Прочность при растяжении, МПа, dtype: float64
Предсказательное значение:[[2409.5596]]
```

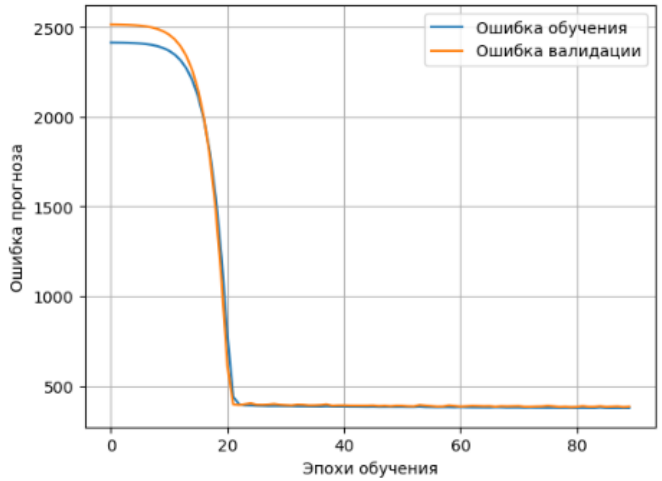
	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_9	6.185s	0.131s	-0.14	376.02	231706.12	481.36	0.15

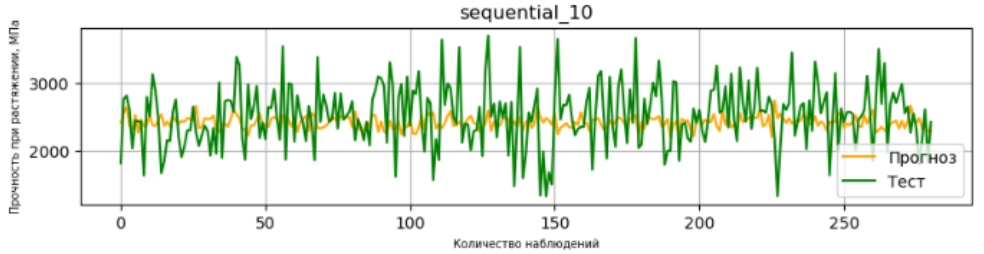





```
y_test значение 359    1823.256512
Name: Прочность при растяжении, МПа, dtype: float64
Предсказательное значение:[[2415.3328]]
```

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_9	6.185s	0.131s	-0.14	376.02	231706.12	481.36	0.15
sequential_10	6.263s	0.148s	-0.09	370.05	221833.85	470.99	0.15





Общие результаты обучения моделей

для показателя «Модуль упругости при растяжении»

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.002s	0.000s	-0.01	2.41	8.76	2.96	0.03
KNeighborsRegressor	0.009s	0.006s	-0.21	2.64	10.53	3.24	0.04
SVR	0.023s	0.017s	-0.03	2.44	8.95	2.99	0.03
DecisionTreeRegressor	0.035s	0.002s	-1.10	3.51	18.28	4.28	0.05
RandomForestRegressor	1.934s	0.012s	-0.02	2.41	8.91	2.99	0.03
GradientBoostingRegressor	0.698s	0.002s	-0.11	2.49	9.63	3.10	0.03
SGDRegressor	0.036s	0.000s	-0.04	2.46	9.06	3.01	0.03
sequential	3.458s	0.153s	-3.19	4.77	36.54	6.04	0.06
sequential_1	11.402s	0.153s	-0.19	2.63	10.41	3.23	0.04

для показателя «Прочность при растяжении»

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
LinearRegression	0.000s	0.008s	-0.05	363.14	213699.15	462.28	0.15
KNeighborsRegressor	0.016s	0.000s	-0.21	387.41	247988.21	497.98	0.16
SVR	0.016s	0.016s	-0.04	361.46	211321.11	459.70	0.15
DecisionTreeRegressor	0.038s	0.000s	-1.37	554.87	483205.11	695.13	0.23
RandomForestRegressor	1.902s	0.018s	-0.10	374.40	224713.09	474.04	0.15
GradientBoostingRegressor	0.696s	0.000s	-0.13	381.93	229672.12	479.24	0.16
SGDRegressor	0.007s	0.000s	-0.09	369.19	221973.99	471.14	0.15
sequential_9	6.185s	0.131s	-0.14	376.02	231706.12	481.36	0.15
sequential_10	6.263s	0.148s	-0.09	370.05	221833.85	470.99	0.15

Нейронная сеть

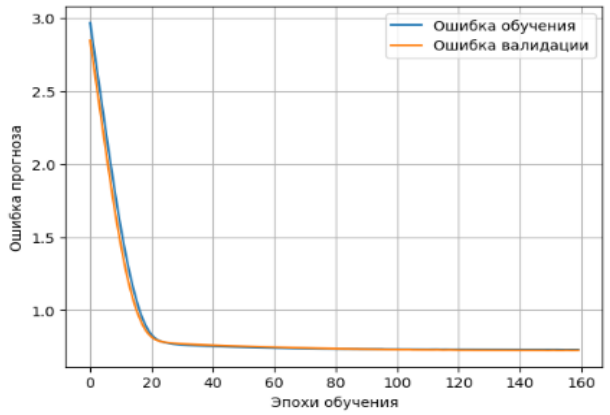
Данные для построения нейронной сети были очищены от выбросов по методу межквартильных расстояний (IQR) и разделены на обучающую и тестовую выборки.

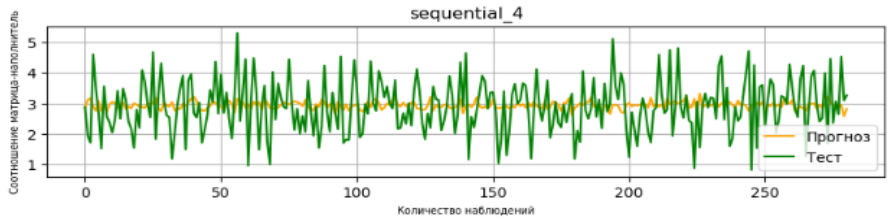
```
model_5 = Sequential([
    normalizer,
    Dense(units=1)
])

model_5.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_6 (Normaliza  (None, 12)                3
 tion)

 dense_22 (Dense)            (None, 1)                 13

=================================================================
Total params: 16 (68.00 Byte)
Trainable params: 13 (52.00 Byte)
Non-trainable params: 3 (16.00 Byte)
_________________________________________________________________
```

```
y_test значение 359    2.871562
Name: Соотношение матрица-наполнитель, dtype: float64
Предсказательное значение:[[2.8620236]]
```

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.226s	0.100s	-0.03	0.71	0.78	0.89	0.30



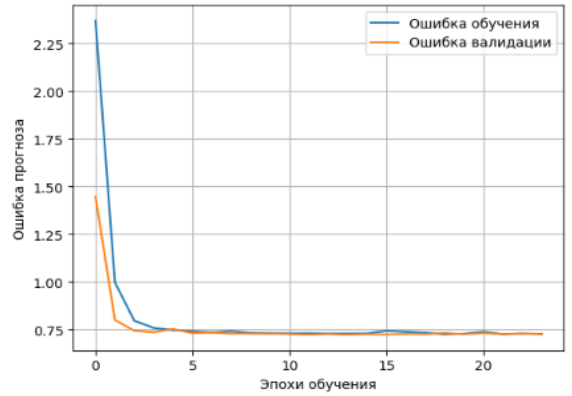


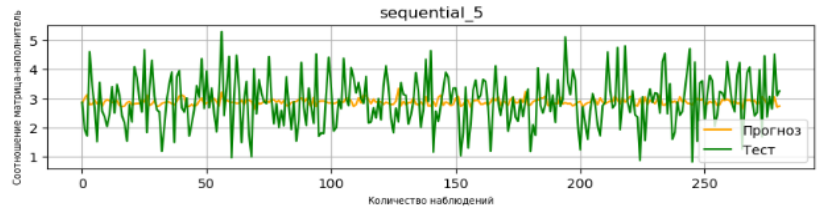

```
model_6 = Sequential([
    normalizer,
    Dense(64, activation='relu'),
    Dense(64, activation='relu'),
    Dense(1)
  ])
model_6.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_6 (Normaliza  (None, 12)                3
 tion)

 dense_23 (Dense)            (None, 64)                832

 dense_24 (Dense)            (None, 64)                4160

 dense_25 (Dense)            (None, 1)                 65

=================================================================
Total params: 5060 (19.77 KB)
Trainable params: 5057 (19.75 KB)
Non-trainable params: 3 (16.00 Byte)
_________________________________________________________________
```

```
y_test значение 359    2.871562
Name: Соотношение матрица-наполнитель, dtype: float64
Предсказательное значение:[[2.8480115]]
```

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.226s	0.100s	-0.03	0.71	0.78	0.89	0.30
sequential_5	1.986s	0.132s	-0.02	0.70	0.78	0.88	0.30



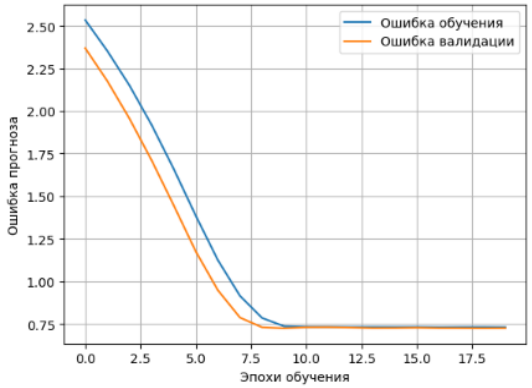


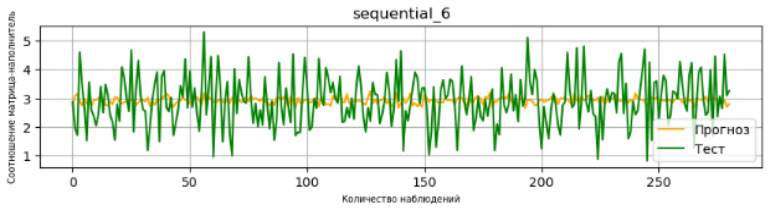

```
model_7 = Sequential([
    normalizer,
    Dense(8, activation='relu'),
    Dense(8, activation='relu'),
    Dense(1)
  ])
model_7.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_6 (Normaliza  (None, 12)                3
 tion)

 dense_26 (Dense)            (None, 8)                 104

 dense_27 (Dense)            (None, 8)                 72

 dense_28 (Dense)            (None, 1)                 9

=================================================================
Total params: 188 (756.00 Byte)
Trainable params: 185 (740.00 Byte)
Non-trainable params: 3 (16.00 Byte)
_________________________________________________________________
```

```
y_test значение 359    2.871562
Name: Соотношение матрица-наполнитель, dtype: float64
Предсказательное значение:[[2.7993858]]
```

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.226s	0.100s	-0.03	0.71	0.78	0.89	0.30
sequential_5	1.986s	0.132s	-0.02	0.70	0.78	0.88	0.30
sequential_6	1.792s	0.216s	-0.01	0.71	0.77	0.88	0.30



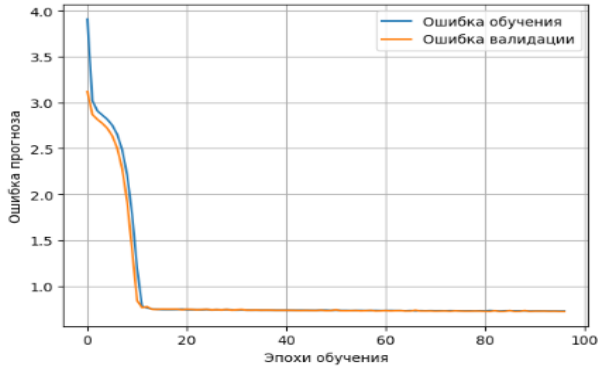


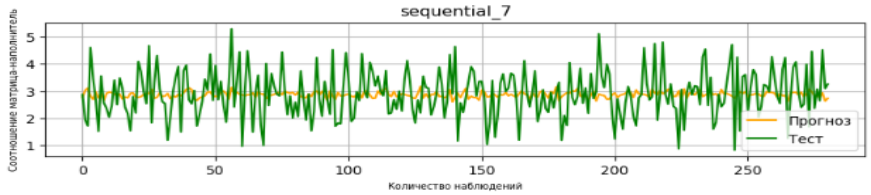
```
model_8 = Sequential([
    normalizer,
    Dense(64, activation='relu'),
    Dense(8, activation='relu'),
    Dense(8, activation='relu'),
    Dense(1)
  ])
model_8.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_6 (Normaliza  (None, 12)                3
 tion)

 dense_29 (Dense)            (None, 64)                832

 dense_30 (Dense)            (None, 8)                 520

 dense_31 (Dense)            (None, 8)                 72

 dense_32 (Dense)            (None, 1)                 9

=================================================================
Total params: 1436 (5.61 KB)
Trainable params: 1433 (5.60 KB)
Non-trainable params: 3 (16.00 Byte)
_________________________________________________________________
```

```
y_test значение 359    2.871562
Name: Соотношение матрица-наполнитель, dtype: float64
Предсказательное значение:[[2.7823238]]
```

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.226s	0.100s	-0.03	0.71	0.78	0.89	0.30
sequential_5	1.986s	0.132s	-0.02	0.70	0.78	0.88	0.30
sequential_6	1.792s	0.216s	-0.01	0.71	0.77	0.88	0.30
sequential_7	6.202s	0.122s	-0.02	0.71	0.77	0.88	0.30



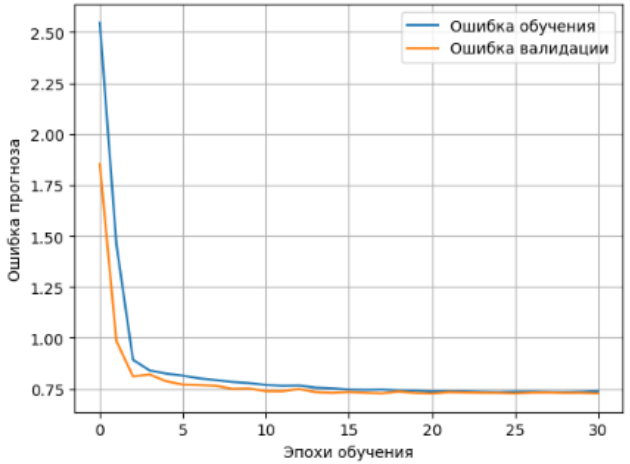


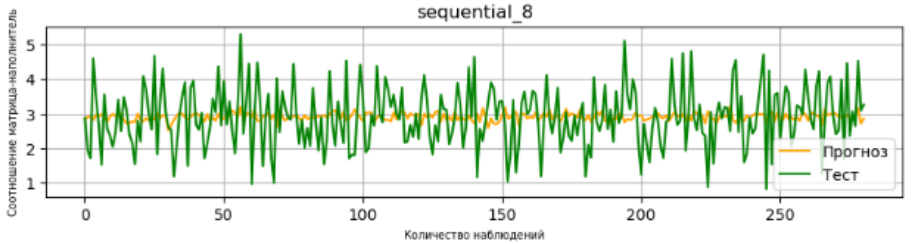

```
model_9 = Sequential([
    normalizer,
    Dense(8, activation='linear'),
    Dense(8, activation='linear'),
    Dense(1)
  ])
model_9.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_6 (Normaliza  (None, 12)                3
 tion)

 dense_33 (Dense)            (None, 8)                 104

 dense_34 (Dense)            (None, 8)                 72

 dense_35 (Dense)            (None, 1)                 9

=================================================================
Total params: 188 (756.00 Byte)
Trainable params: 185 (740.00 Byte)
Non-trainable params: 3 (16.00 Byte)
_________________________________________________________________
```

```
y_test значение 359    2.871562
Name: Соотношение матрица-наполнитель, dtype: float64
Предсказательное значение:[[2.8346715]]
```

	Training time	Predict time	R2	MAE	MSE	RMSE	MAPE
sequential_4	9.226s	0.100s	-0.03	0.71	0.78	0.89	0.30
sequential_5	1.986s	0.132s	-0.02	0.70	0.78	0.88	0.30
sequential_6	1.792s	0.216s	-0.01	0.71	0.77	0.88	0.30
sequential_7	6.202s	0.122s	-0.02	0.71	0.77	0.88	0.30
sequential_8	3.294s	0.116s	-0.02	0.71	0.78	0.88	0.30





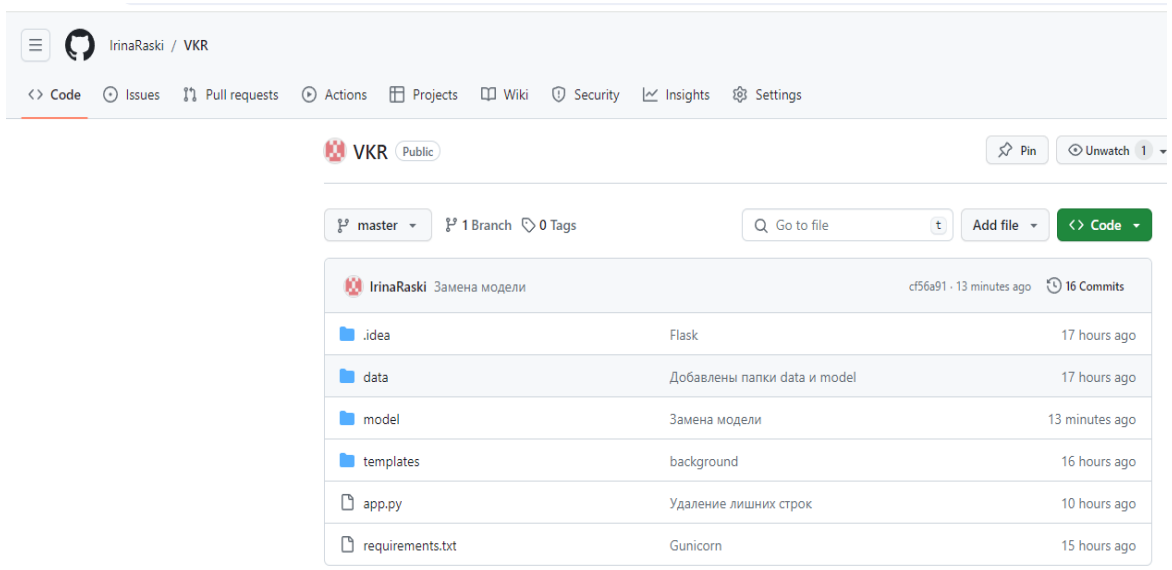
Разработка приложения


```python
import numpy as np
from flask import Flask, render_template, request
import pickle

app = Flask(__name__, template_folder='templates')
model = pickle.load(open(r'model/model.pkl', 'rb'))


 irinaraski
@app.route( rule: '/', methods=['get', 'post'])
def main():
    if request.method == 'GET':
        return render_template('main.html')

    if request.method == 'POST':
        prediction_text = [float(x) for x in request.form.values()]
        x_new = [np.array(prediction_text)]
        prediction = model.predict(x_new).flatten()
        result = f'{str(prediction)[1:-1]}'
        return render_template( template_name_or_list: 'main.html', result=result)


if __name__ == '__main__':
    app.run()
```

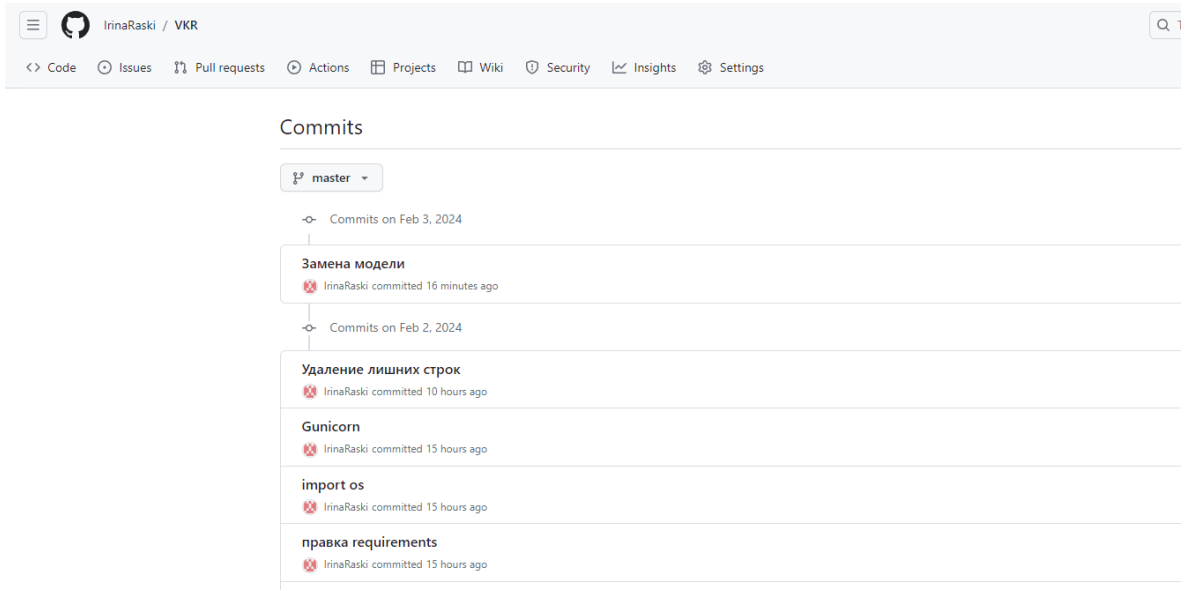
Создание удаленного репозитория

Репозиторий создан на github.com по адресу:

https://github.com/IrinaRaski/VKR



профиль на github.com



commit приложения

БЛАГОДАРЮ ЗА ВНИМАНИЕ !