ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по курсу
«Data Science Pro»

Слушатель Радьков Дмитрий Васильевич

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Загрузка данных

Данные из двух файлов объединяем в один DataFrame. Используем INNER и индекс. Удаляем безымянные колонки.

```
# объеденяем
df = df_xbp.merge(df_xnup, left_index = True, right_index = True, how = 'inner')
# df.sample(4)
```

```
df.drop(df.filter(regex='^Unnamed:'), axis=1, inplace=True)
print(df.shape)
```

Смотрим типы данных, пропуски и уникальные значения, изучаем описательную статистику

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0000	2.9304	0.9132	0.3894	2.3179	2.9069	3.5527	5.5917
Плотность, кг/м3	1023.0000	1975.7349	73.7292	1731.7646	1924.1555	1977.6217	2021.3744	2207.7735
модуль упругости, ГПа	1023.0000	739.9232	330.2316	2.4369	500.0475	739.6643	961.8125	1911.5365
Количество отвердителя, м.%	1023.0000	110.5708	28.2959	17.7403	92.4435	110.5648	129.7304	198.9532
Содержание эпоксидных групп,%_2	1023.0000	22.2444	2.4063	14.2550	20.6080	22.2307	23.9619	33.0000
Температура вспышки, С_2	1023.0000	285.8822	40.9433	100.0000	259.0665	285.8968	313.0021	413.2734
Поверхностная плотность, г/м2	1023.0000	482.7318	281.3147	0.6037	266.8166	451.8644	693.2250	1399.5424
Модуль упругости при растяжении, ГПа	1023.0000	73.3286	3.1190	64.0541	71.2450	73.2688	75.3566	82.6821
Прочность при растяжении, МПа	1023.0000	2466.9228	485.6280	1036.8566	2135.8504	2459.5245	2767.1931	3848.4367
Потребление смолы, г/м2	1023.0000	218.4231	59.7359	33.8030	179.6275	219.1989	257.4817	414.5906
Угол нашивки, град	1023.0000	44.2522	45.0158	0.0000	0.0000	0.0000	90.0000	90.0000
Шаг нашивки	1023.0000	6.8992	2.5635	0.0000	5.0800	6.9161	8.5863	14.4405
Плотность нашивки	1023.0000	57.1539	12.3510	0.0000	49.7992	57.3419	64.9450	103.9889

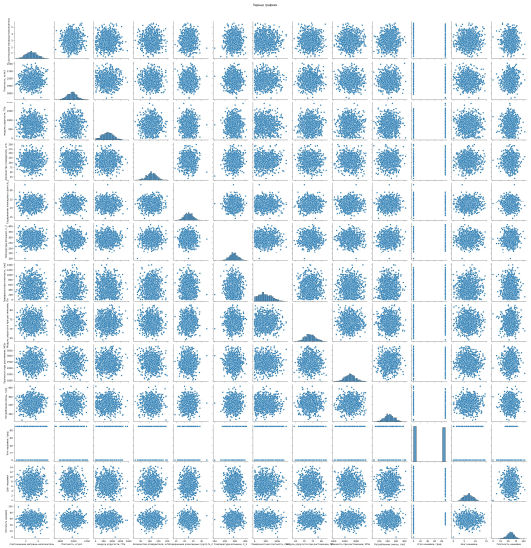
	DataType	MissingValues	UniqueValues	MissingValuesRatio
Соотношение матрица-наполнитель	float64	0	1014	0.0000
Плотность, кг/м3	float64	0	1013	0.0000
модуль упругости, ГПа	float64	0	1020	0.0000
Количество отвердителя, м.%	float64	0	1005	0.0000
Содержание эпоксидных групп,%_2	float64	0	1004	0.0000
Температура вспышки, С_2	float64	0	1003	0.0000
Поверхностная плотность, г/м2	float64	0	1004	0.0000
Модуль упругости при растяжении, ГПа	float64	0	1004	0.0000
Прочность при растяжении, МПа	float64	0	1004	0.0000
Потребление смолы, г/м2	float64	0	1003	0.0000
Угол нашивки, град	int64	0	2	0.0000
Шаг нашивки	float64	0	989	0.0000
Плотность нашивки	float64	0	988	0.0000

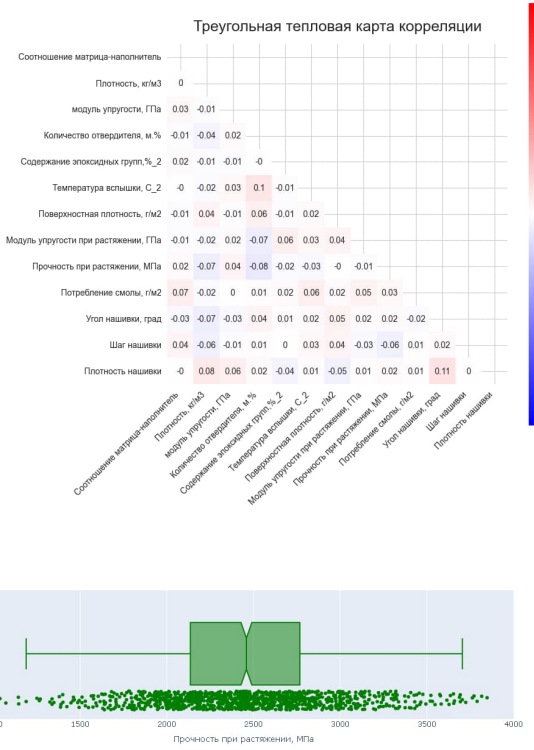
```
# Посмотрим на значения признака 'Угол нашивки, град'
df['Угол нашивки, град'].value_counts()
# будем использовать как числовой для начала
```

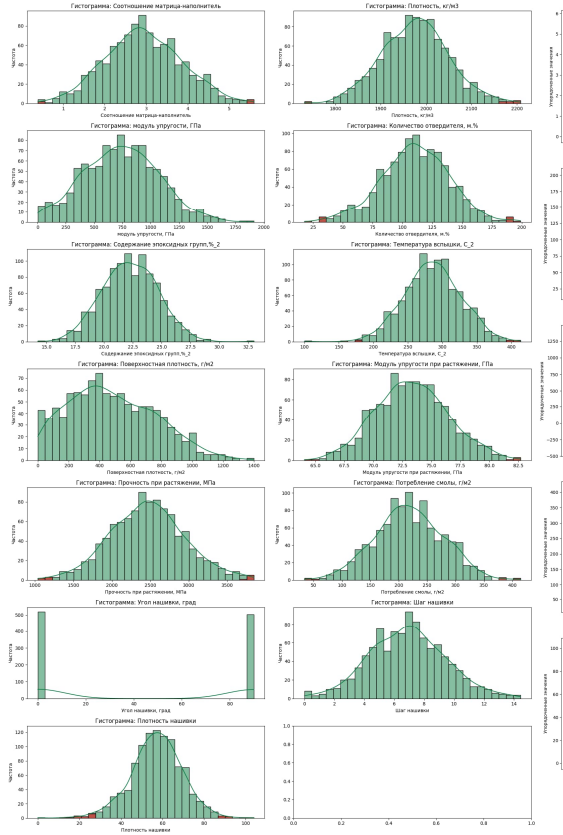
```
Угол нашивки, град
0     520
90    503
Name: count, dtype: int64
```

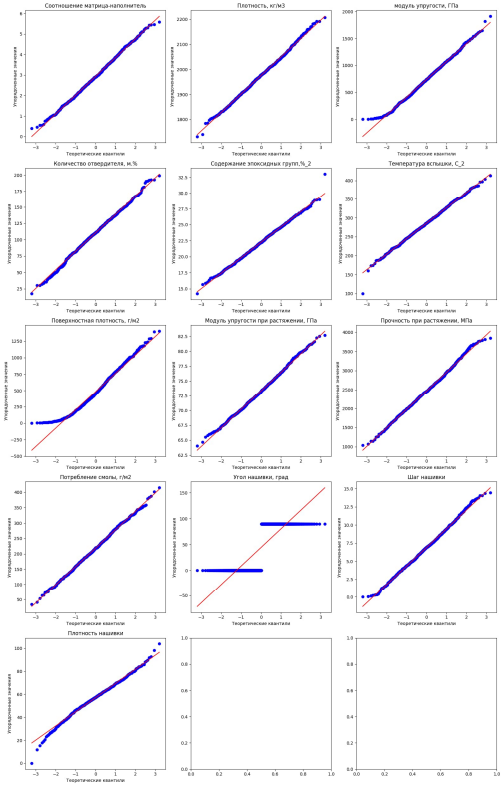
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Визуализация исходных данных





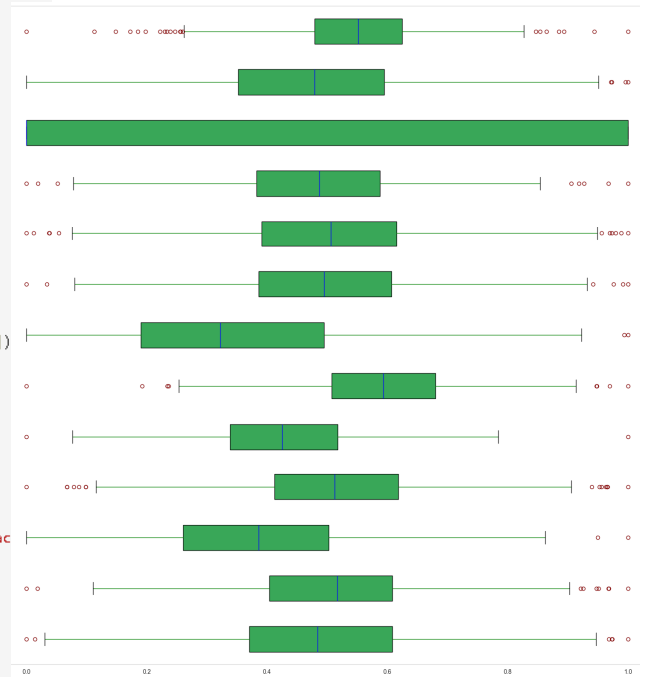




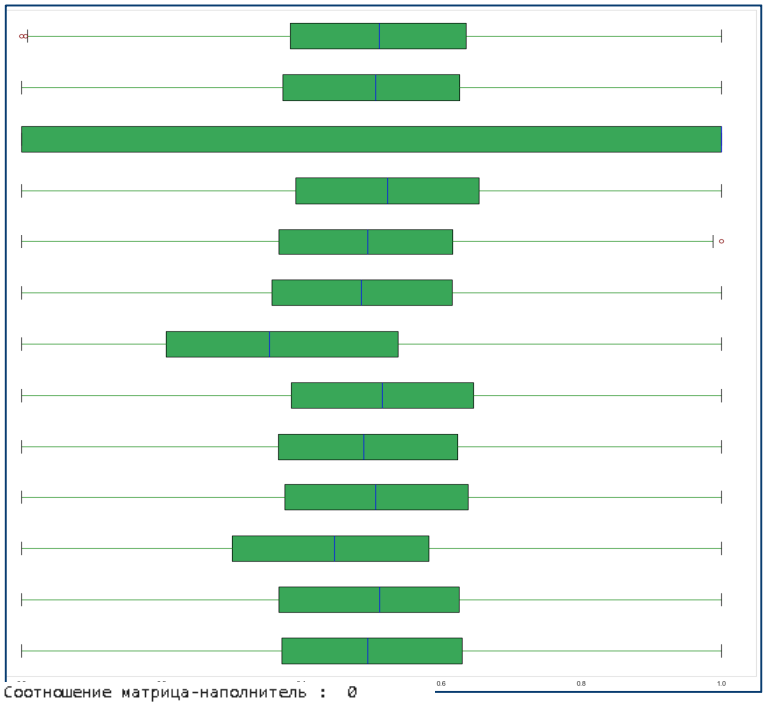
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Обработка выбросов

```python
def show_outs(df):
    sum_iq = 0
    cnt_iq = []
    for column in df:
        d = df.loc[:, [column]]
        q1 = np.quantile(df[column], 0.25)
        q3 = np.quantile(df[column], 0.75)
        iqr = q3 - q1
        lower = q1 - 1.5 * iqr
        upper = q3 + 1.5 * iqr
        d['iq'] = (df[column] < lower) | (df[column] > upper)
        sum_iq += d['iq'].sum()
        cnt_iq.append(d['iq'].sum())
        print(column, ': ', d['iq'].sum())
    # print('Всего выбросов:', sum_iq)
    # print('Всего выбросов:', end=' ')
    print(colored(f"Всего выбросов: {sum_iq}  ",'black', 'on_red', attrs=["bold"])

def show_outs_gr(df):
    scaler = MinMaxScaler()
    scaler.fit(df)
    plt.figure(figsize = (20, 20))
    plt.boxplot(pd.DataFrame(scaler.transform(df)),
                labels = df.columns, patch_artist = True,
                meanline = True, vert = False,
                boxprops = dict(facecolor = colors_from_palette[17], color = 'blac
                medianprops = dict(color = 'b'),
                whiskerprops = dict(color = "g"),
                capprops = dict(color = "black"),
                flierprops = dict(color = "y", markeredgecolor = "maroon"))

    plt.grid ( False )
    plt.show()
```



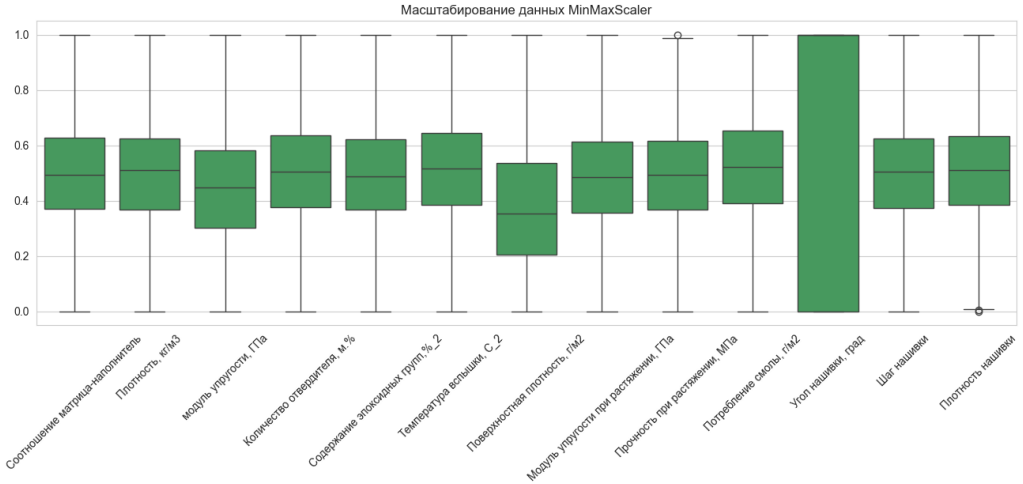
```
Соотношение матрица-наполнитель :  6
Плотность, кг/м3 :  9
модуль упругости, ГПа :  2
Количество отвердителя, м.% :  14
Содержание эпоксидных групп,%_2 :  2
Температура вспышки, С_2 :  8
Поверхностная плотность, г/м2 :  2
Модуль упругости при растяжении, ГПа :  6
Прочность при растяжении, МПа :  11
Потребление смолы, г/м2 :  8
Угол нашивки, град :  0
Шаг нашивки :  4
Плотность нашивки :  21
Всего выбросов: 93
```

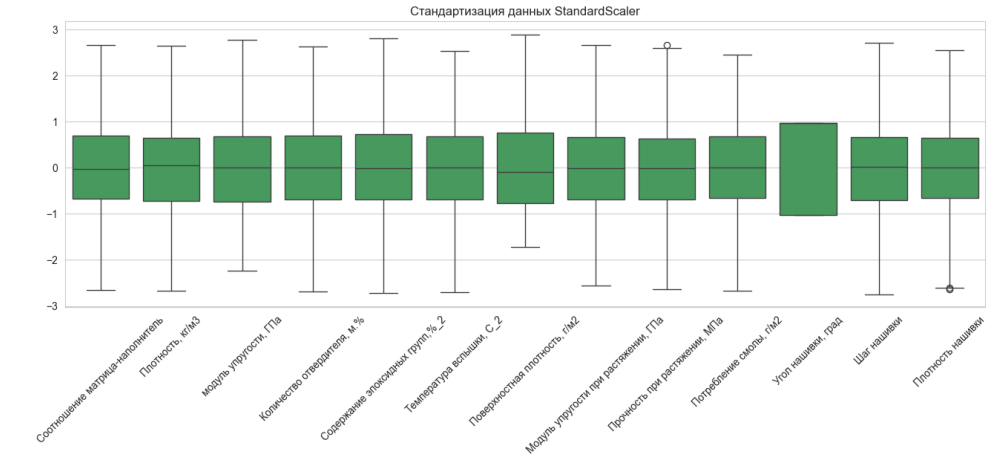


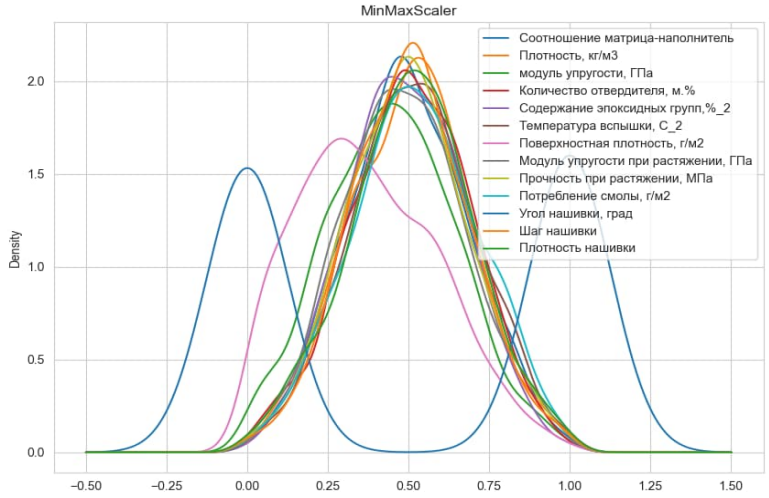
```
Соотношение матрица-наполнитель :  0
Плотность, кг/м3 :  0
модуль упругости, ГПа :  0
Количество отвердителя, м.% :  0
Содержание эпоксидных групп,%_2 :  0
Температура вспышки, С_2 :  0
Поверхностная плотность, г/м2 :  0
Модуль упругости при растяжении, ГПа :  0
Прочность при растяжении, МПа :  1
Потребление смолы, г/м2 :  0
Угол нашивки, град :  0
Шаг нашивки :  0
Плотность нашивки :  2
Всего выбросов: 3
```

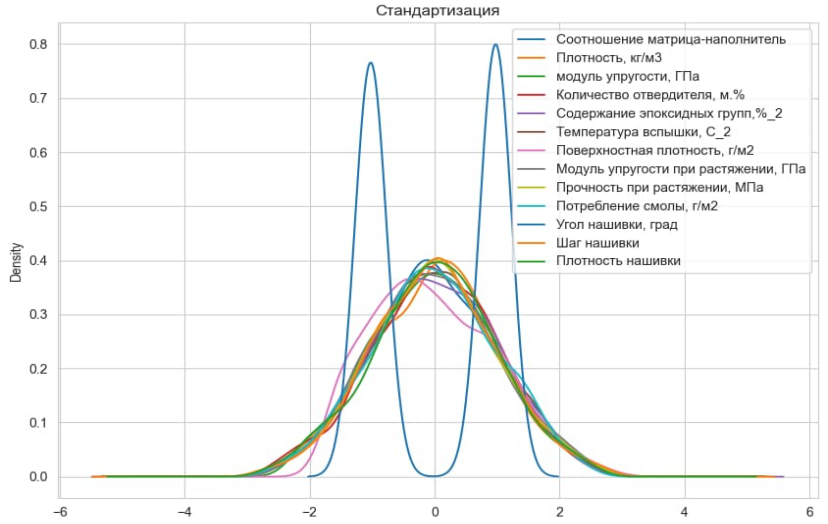
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Предобработка данных









ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Разработка моделей

Целевые переменные:

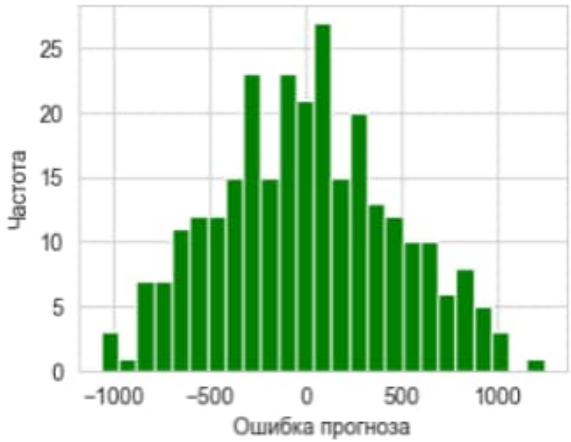
1. Прочность при растяжении, МПа.

2. Модуль упругости при растяжении, ГПа;

Исследуем модели:

1. Метод лассо регрессии
2. Метод опорных векторов
3. Метод случайного леса
4. Метод градиентного бустинга
5. Метод К ближайших соседей
6. Метод решающих деревьев

```
pipe = Pipeline(steps = [('scaler', StandardScaler()),
                         ('svr', SVR())
                        ],
                verbose = False)

param_grid = {
    'svr__C': [0.1, 1, 10, 100, 1000],
    'svr__epsilon': [0.01, 1, 10],
    'svr__kernel': ['rbf', 'linear']
}

grid = GridSearchCV(pipe, param_grid, cv = 10, verbose =1,
                    # scoring = 'neg_mean_absolute_error',
                    )
grid.fit(x_train, np.ravel(y_train))

y_pred = grid.predict(x_test)

r2 = r2_score(y_test, y_pred)
r2_cv = grid.best_score_
mae = mean_absolute_error(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test, y_pred)
rmse =  root_mean_squared_error(y_test, y_pred)
print(f"Лучшие параметры: {grid.best_params_}")
print(f"Качество на кросс валидации R2: {r2_cv:.3f}")
print(f"Качество на тестовой выборке R2: {grid.score(x_test, y_test):.3f}")
print(f"MAE: {mae:.3f}")
print(f"MAPE: {mape:.3f}")
print(f"RMSE: {rmse:.3f}")

m_res['SVR'] = [r2_cv, r2, mae, mape, rmse,str(grid.best_params_)]
show_gist(y_test, y_pred)
```



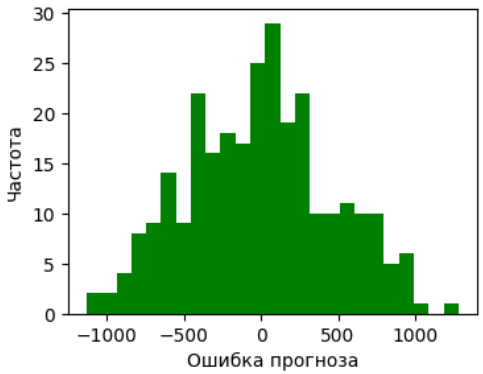
r2_cv
r2
MAE
MAPE
RMSE

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Разработка моделей

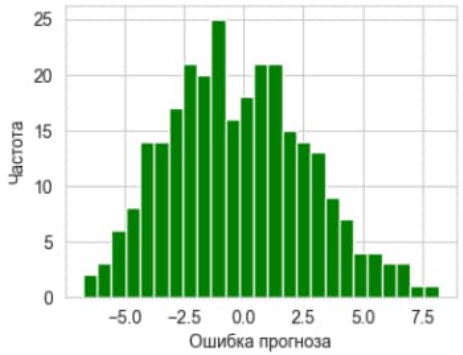
Прочность при растяжении, МПа.

	r2_cv	r2	MAE	MAPE	RMSE
Lasso	-0.0072	-0.0004	372.1623	0.1619	461.4009
SVR	-0.0070	0.0010	372.8447	0.1614	461.0750
RFR	-0.0340	0.0198	372.4440	0.1616	456.7380
gbr	-0.1436	-0.0332	382.7108	0.1648	468.9204
KNN	-0.0086	0.0033	372.7264	0.1625	460.5492
DTR_cv	0.0050	-0.0025	374.3168	0.1626	461.8998
DTR_rcv	0.0124	-0.0093	374.3611	0.1627	463.4595



Модуль упругости при растяжении, ГПа

	r2_cv	r2	MAE	MAPE	RMSE
Lasso	-0.0276	-0.0044	2.3853	0.0327	2.9225
SVR	-0.0244	-0.0119	2.3951	0.0329	2.9335
RFR	-0.0778	-0.0125	2.4182	0.0331	2.9342
gbr	-0.1422	-0.1382	2.5299	0.0347	3.1111
KNN	-0.0284	-0.0095	2.4008	0.0329	2.9300
DTR_cv	-0.0160	-0.0057	2.4059	0.0330	2.9245
DTR_rcv	-0.0127	0.0021	2.3860	0.0327	2.9131



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Нейронная сеть

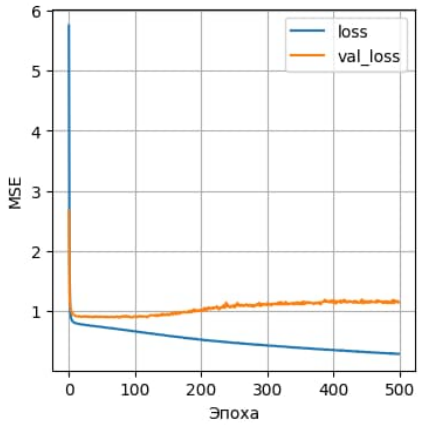
Целевая переменная «Соотношение матрица-наполнитель»

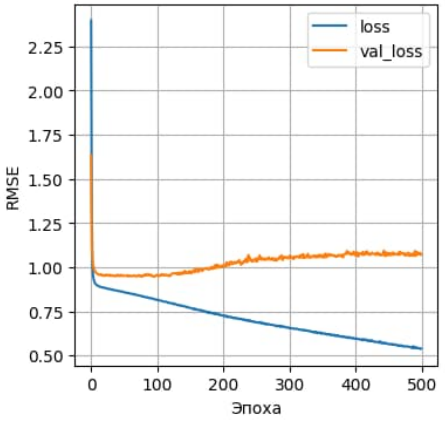
```
# слой нормализации

normalizer = tf.keras.layers.Normalization(input_shape=[12,], axis=-1)
normalizer.adapt(np.array(x_train))
```

```
plot_loss(history.history)

score = model.evaluate(x_test, y_test, batch_size=32)
mse = score[0]
rmse = score[1]
r2 = r2_score(y_test, model.predict(x_test))
print(f'mse: {mse:.6f}')
print(f'rmse: {rmse:.6f}')
print(f'R2: {r2:.6f}')

m_res['net_1'] = [r2, mse, rmse, model]
```

Сеть 1

```
model = Sequential([normalizer,
                    layers.Dense(8*2, activation='tanh'),
                    layers.Dense(16*2, activation='tanh'),
                    layers.Dense(24*2, activation='tanh'),
                    layers.Dense(1)
                    ])
model.compile(optimizer = tf.keras.optimizers.SGD(0.005),
              loss = 'mean_squared_error',
              metrics = [tf.keras.metrics.RootMeanSquaredError()])
```





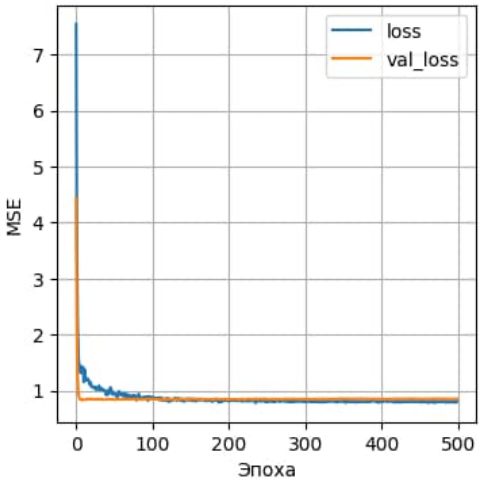
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

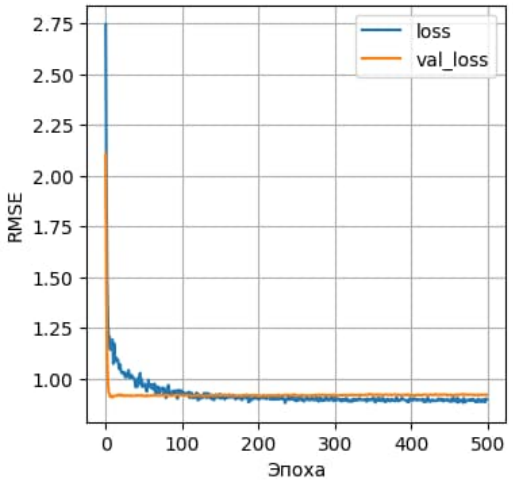
Нейронная сеть

Целевая переменная «Соотношение матрица-наполнитель»

Сеть 2

```
# добавим Dropout
model = Sequential([normalizer,
                    layers.Dense(8*2, activation='tanh'),
                    layers.Dropout(0.2),
                    layers.Dense(16*2, activation='tanh'),
                    layers.Dropout(0.3),
                    layers.Dense(24*2, activation='tanh'),
                    layers.Dropout(0.5),
                    layers.Dense(1)
                    ])

model.compile(optimizer = tf.keras.optimizers.SGD(0.005),
              loss = 'mean_squared_error',
              metrics = [tf.keras.metrics.RootMeanSquaredError()])
```



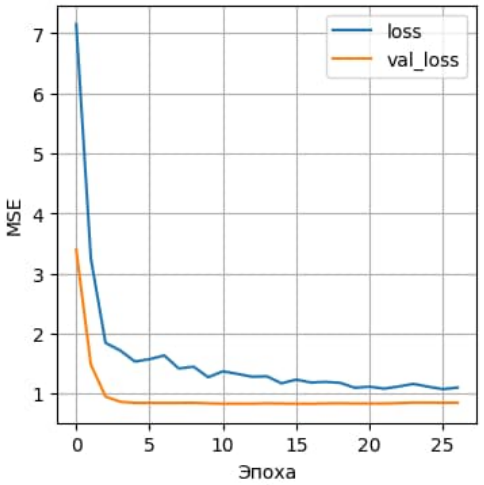


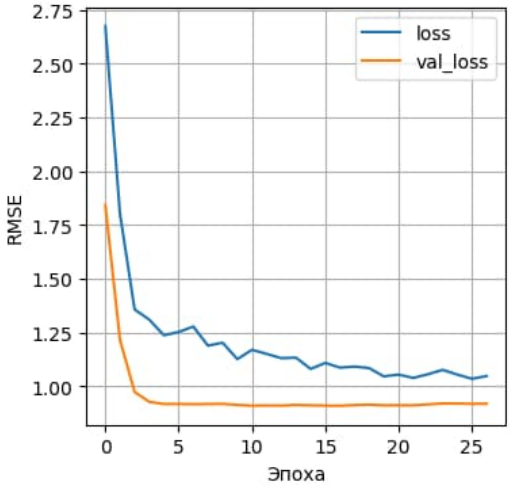
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Нейронная сеть

Сеть 3

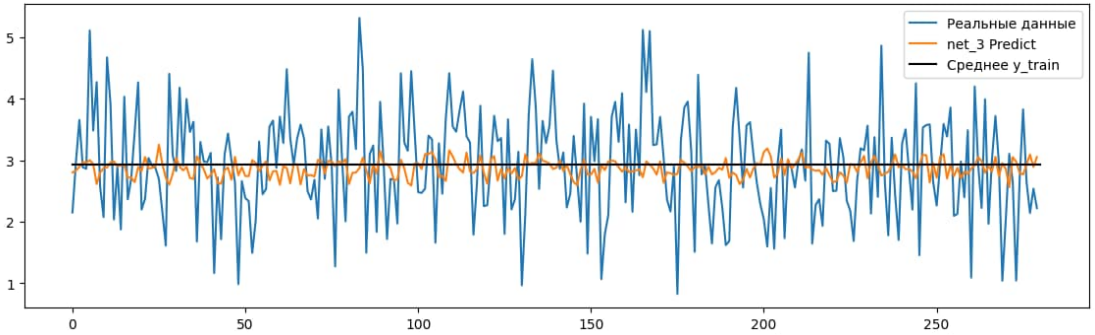
```
# добавим Dropout
model = Sequential([normalizer,
                    layers.Dense(8*2, activation='tanh'),
                    layers.Dropout(0.2),
                    layers.Dense(16*2, activation='tanh'),
                    layers.Dropout(0.3),
                    layers.Dense(24*2, activation='tanh'),
                    layers.Dropout(0.5),
                    layers.Dense(1)
                    ])

model.compile(optimizer = tf.keras.optimizers.SGD(0.005),
              loss = 'mean_squared_error',
              metrics = [tf.keras.metrics.RootMeanSquaredError()])

early_stopping_callback = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
```






```
metric_cols = ['r2', 'MSE','RMSE']
res_df = pd.DataFrame(m_res, index= metric_cols + ['model_obj'] ).T
res_df[metric_cols]
```

	r2	MSE	RMSE
net_1	-0.9071	1.4118	1.1882
net_2	-0.0312	0.7634	0.8737
net_3	-0.0278	0.7609	0.8723




```
best_net_predict = res_df.loc['net_3', 'model_obj'].predict(x_test)
```

ЦЕНТР
ДОПОЛНИТЕЛЬНОГО
ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

do.bmstu.ru