ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Выпускная квалификационная работа по курсу “Data Science Pro”

«Прогнозирование конечных свойств новых материалов (композиционных материалов)»

**Маев Игорь Александрович**

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Цель работы

- предобработка, анализ данных,
- обучение моделей для обработки данных,
- демонстрация результатов работы с использованием методов и инструментов, изученных в рамках программы профессиональной переподготовки «Data Science Pro» в МГТУ им. Н.Э. Баумана

# Результаты работы

https://github.com/albaross86/GQP

https://gqp.onrender.com

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Ход работы

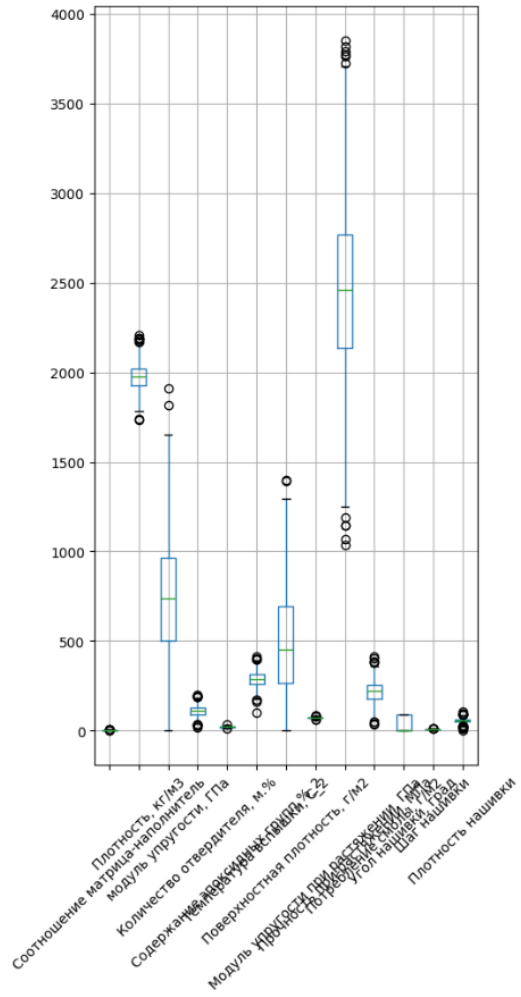
1. Изучение теоретических основ и методов решения поставленной задачи
2. Разведочный анализ данных, предобработка
3. обучение нескольких моделей для прогноза модуля упругости при растяжении и прочности при растяжении, поиск гиперпараметров модели с помощью поиска по сетке с перекрестной проверкой
4. написание нейронной сети для рекомендации соотношения матрица-наполнитель, оценка точности модели на тренировочном и тестовом датасете
5. разработка приложения с графическим интерфейсом для выдачи прогноза, полученного в задании 5; создание репозитория в GitHub, размещение в нём кода исследования с оформлением файла README

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# EDA, предобработка

- Количество входных переменных – 13
- Объём выборки (после объединения): 1023
- Количество пропусков: 0
- Количество уникальных значений у большинства переменных - более 1000
- Лишь одна переменная («Угол нашивки») имеет два уникальных значения – 0 и 90 градусов).
- Количество дубликатов: 0
- Выбросы в ряде параметров видны, но экстремальных ошибочных выбросов нет

```
#Вывод "ящиков с усами" для предварительного (визуального) анализа выбросов
boxplot = df.boxplot(rot=45, figsize=(5,10))
plt.show()
```
✓ 0.3s




```
df.describe()
```
✓ 0.0s Python

| | 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 | Шаг нашивки | Плотность нашивки |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 285.882151 | 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 |

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Методы

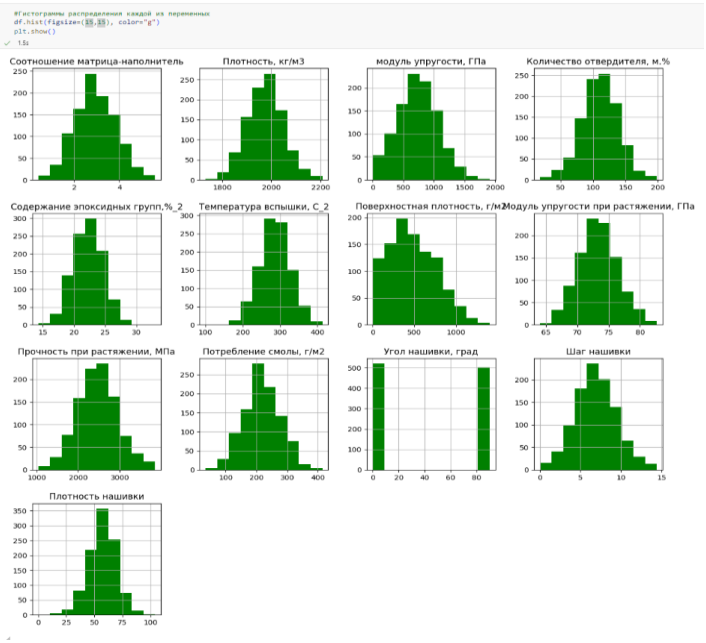
1. Линейная регрессия (Linear Regression)
2. «Случайный лес» (Random Forest)
3. Градиентный бустинг (Gradient Boosting)
4. Метод опорных векторов для задач регрессии (SVR)
5. К ближайших соседей (k Nearest Neighbors)

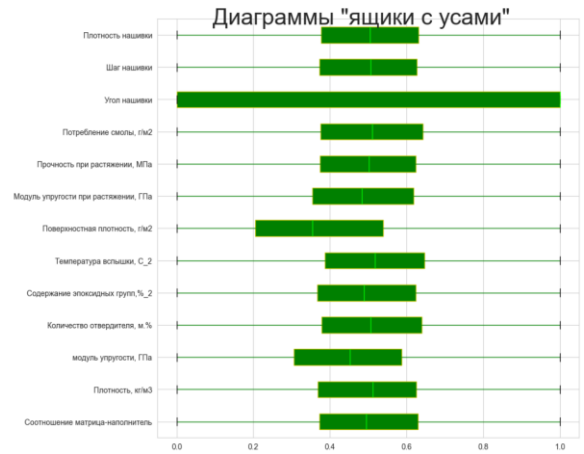
**ОБУЧЕНИЕ С УЧИТЕЛЕМ:**

**ЗАДАЧА РЕГРЕССИИ**

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

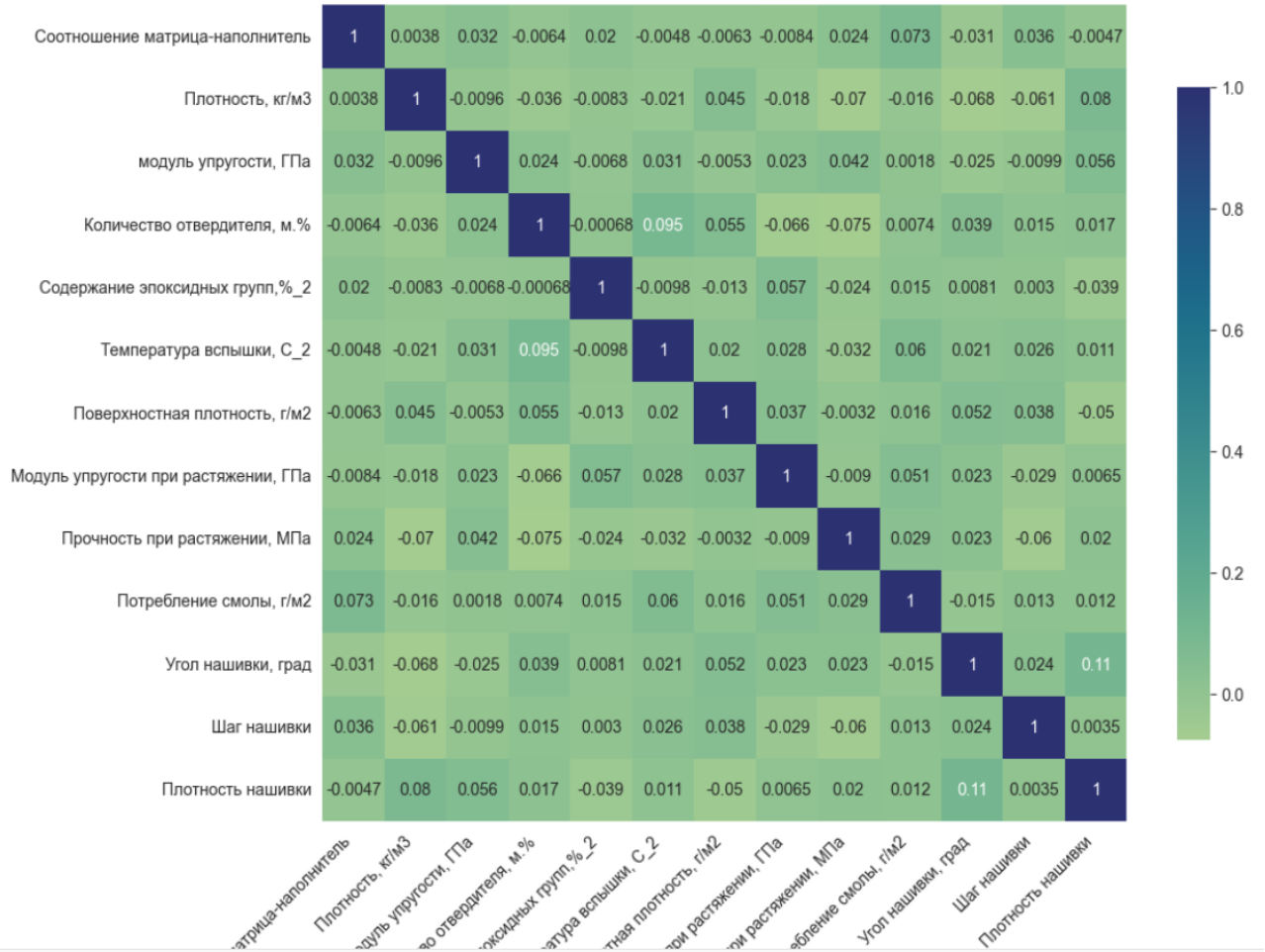
# EDA, предобработка




```
plt.figure(figsize=(11, 9))
ax = sns.heatmap(dataset_norm_df.corr(), annot=True, cmap='crest', cbar_kws={'shrink': 0.8})
ax.set(xlabel="", ylabel="")
plt.xticks(rotation=45, ha='right')
plt.show()
```

✓ 0.4s

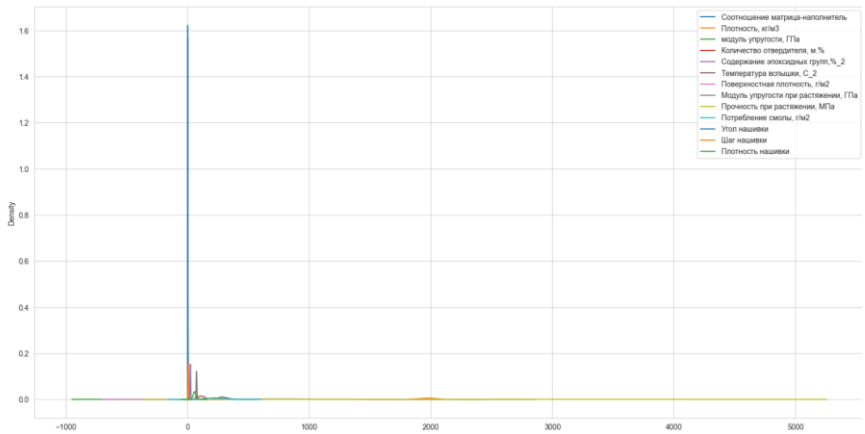


ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# EDA, предобработка

До

```
# Оценка плотности ядра до нормализации
def plt_kde(df):
    fig, ax = plt.subplots(figsize=(20, 10))
    df.plot(kind='kde', ax=ax)
plt_kde(df)
```

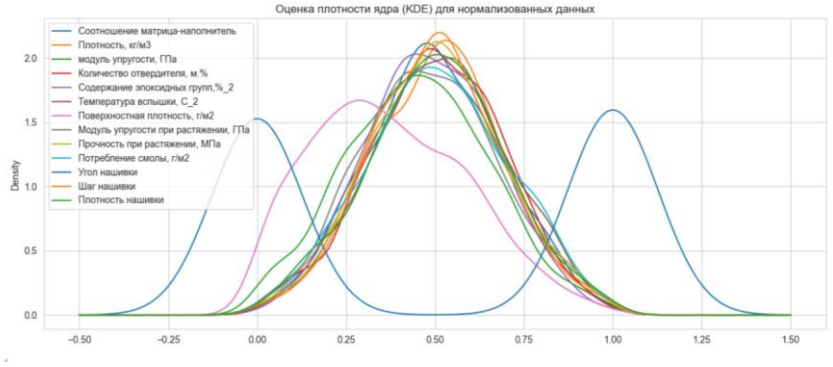



StandardScaler

```
#Выводим график оценки плотности ядра (KDE) для стандартизированных данных
fig, ax = plt.subplots(figsize = (15, 6))
df_std_2.plot(kind = 'kde', ax = ax)  #график плотности ядра для нормализованных данных
plt.title('Оценка плотности ядра (KDE) для нормализованных данных')
plt.show()
```

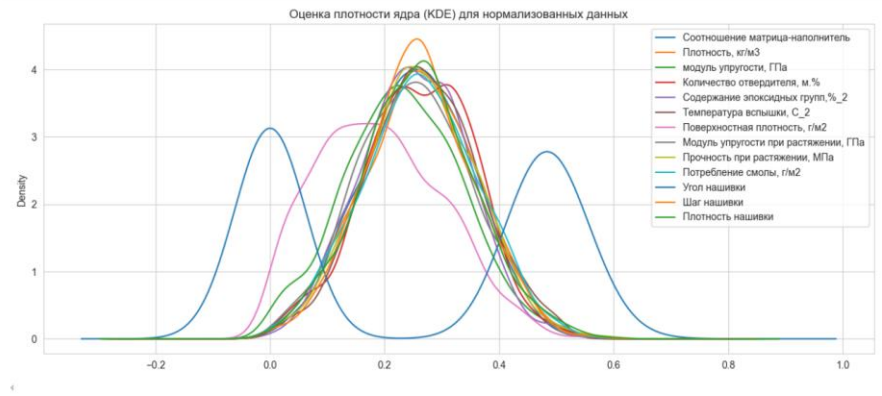

MinMax

```
#Выводим график оценки плотности ядра (KDE) для нормализованных данных
fig, ax = plt.subplots(figsize = (15, 6))
df_cleaned3_norm_df.plot(kind = 'kde', ax = ax)  #график плотности ядра для нормализованных данных
plt.title('Оценка плотности ядра (KDE) для нормализованных данных')
plt.show()
```




Normalizer

```
#Выводим график оценки плотности ядра (KDE) для нормализованных данных
fig, ax = plt.subplots(figsize = (15, 6))
df_norm_n.plot(kind = 'kde', ax = ax)  #график плотности ядра для нормализованных данных
plt.title('Оценка плотности ядра (KDE) для нормализованных данных')
plt.show()
```



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Обучение моделей

Составляем отчёт по избранным метрикам.

```
Report = {"Metrics":["MAE","MSE","R2 Score Test"],
          "Linear Regression S": [mae_lr, mse_lr, lr.score(x_test_1, y_test_1)],
          "Linear Regression N": [mae_lr2, mse_lr2, lr2.score(x_test_2, y_test_2)],
          "Random Forest S": [mae_rfr, mse_rfr, rfr.score(x_test_1, y_test_1)],
          "Random Forest N": [mae_rfr2, mse_rfr2, rfr2.score(x_test_2, y_test_2)],
          "Gradient Boosting S": [mae_gbr, mse_gbr, gbr.score(x_test_1, y_test_1)],
          "Gradient Boosting N": [mae_gbr2, mse_gbr2, gbr2.score(x_test_2, y_test_2)],
          "SVR S": [mae_svr, mse_svr, svr.score(x_test_1, y_test_1)],
          "SVR N": [mae_svr2, mse_svr2, svr2.score(x_test_2, y_test_2)],
          "KNN S": [mae_knn, mse_knn, knn.score(x_test_1, y_test_1)],
          "KNN N": [mae_knn2, mse_knn2, knn2.score(x_test_2, y_test_2)]

           }


dfres = pd.DataFrame(Report)
dfres
```

✓ 0.0s

| | Metrics | Linear Regression S | Linear Regression N | Random Forest S | Random Forest N | Gradient Boosting S | Gradient Boosting N | SVR S | SVR N | KNN S | KNN N |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | MAE | 0.823007 | 0.059330 | 0.852347 | 0.073481 | 0.850682 | 0.070220 | 0.914542 | 0.083163 | 0.890912 | 0.419233 |
| 1 | MSE | 1.047282 | 0.005668 | 1.132812 | 0.008882 | 1.097008 | 0.008244 | 1.287698 | 0.010846 | 1.263931 | 0.009742 |
| 2 | R2 Score Test | -0.005456 | 0.408192 | -0.087570 | 0.072682 | -0.053196 | 0.139285 | -0.236270 | -0.132376 | -0.213452 | -0.017154 |

```
dfres2 = dfres[["Linear Regression S", "Linear Regression N", "Random Forest S", "Random Forest N", "Gradient Boosting S", "Gradient Boosting N", "SVR S", "SVR N", "K
dfres2.index = dfres["Metrics"]


dfres3 = dfres2.style.background_gradient(cmap='Greens', axis=1)
dfres3
```

| Metrics | Linear Regression S | Linear Regression N | Random Forest S | Random Forest N | Gradient Boosting S | Gradient Boosting N | SVR S | SVR N | KNN S | KNN N |
|---|---|---|---|---|---|---|---|---|---|---|
| MAE | 0.823007 | 0.059330 | 0.852347 | 0.073481 | 0.850171 | 0.070138 | 0.914542 | 0.083163 | 0.890912 | 0.419233 |
| MSE | 1.047282 | 0.005668 | 1.132812 | 0.008882 | 1.096915 | 0.008248 | 1.287698 | 0.010846 | 1.263931 | 0.009742 |
| R2 Score Test | -0.005456 | 0.408192 | -0.087570 | 0.072682 | -0.053107 | 0.138898 | -0.236270 | -0.132376 | -0.213452 | -0.017154 |

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Поиск гиперпараметров модели по сетке с перекрестной проверкой

```
pipe2 = Pipeline([('preprocessing', StandardScaler()), ('regressor', SVR())])
param_grid2 = [
{'regressor': [SVR()], 'preprocessing': [StandardScaler(), MinMaxScaler(), None],
'regressor__gamma': [0.001, 0.01, 0.1, 1, 10, 100],
'regressor__C': [0.001, 0.01, 0.1, 1, 10, 100]},
{'regressor': [RandomForestRegressor(n_estimators=100)],
'preprocessing':  [StandardScaler(), MinMaxScaler(), None]},
{'regressor': [LinearRegression()], 'preprocessing':  [StandardScaler(), MinMaxScaler(), None]},
{'regressor': [GradientBoostingRegressor()], 'preprocessing':  [StandardScaler(), MinMaxScaler(), None]},
{'regressor': [KNeighborsRegressor()], 'preprocessing':  [StandardScaler(), MinMaxScaler(), None]},]
grid2 = GridSearchCV(pipe2, param_grid2, cv=10)
grid2.fit(x_train_1, np.ravel(y_train_1))
print("Наилучшие параметры:\n{}\n".format(grid2.best_params_))
print("Наилучшее значение правильности перекрестной проверки: {:.2f}".format(grid2.best_score_))
print("Правильность на тестовом наборе: {:.2f}".format(grid2.score(x_test_1, y_test_1)))
```

```
Наилучшие параметры:
{'preprocessing': StandardScaler(), 'regressor': SVR(), 'regressor__C': 1, 'regressor__gamma': 1}

Наилучшее значение правильности перекрестной проверки: -0.01
Правильность на тестовом наборе: 0.00
```

```
print("Наилучшая модель:\n{}".format(grid2.best_estimator_))
```

```
Наилучшая модель:
Pipeline(steps=[('preprocessing', StandardScaler()),
                ('regressor', SVR(C=1, gamma=1))])
```

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Написание нейросети

Нейросеть для рекомендации соотношения "матрица - наполнитель"

```
#@title Нейронная сеть, которая рекомендует соотношение матрица-наполнитель
import tensorflow as tf
from tensorflow import keras

from keras import Sequential
from keras.models import Model
from keras.layers import Input, Dense
from keras import utils
from keras.layers import BatchNormalization



from PIL import Image
import matplotlib.pyplot as plt
%matplotlib inline

print(f'tensorflow ver: {tf.__version__}')
```

✓ 0.0s

```
tensorflow ver: 2.18.0
```

```
min_max_scaler = MinMaxScaler()
standard_scaler = StandardScaler()

X = np.array(df.drop('Соотношение матрица-наполнитель', axis = 1))
y = np.array(df['Соотношение матрица-наполнитель'])

y = y.reshape(-1, 1)

X_scaled = min_max_scaler.fit_transform(X)
y_scaled = min_max_scaler.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled,
                                                    test_size = 0.3,
                                                    shuffle = True)
```

✓ 0.0s

```
#@title Функция, собирающая НС
def construct_model():
    return tf.keras.Sequential([
    keras.layers.Input(shape=(12,)),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dense(units=64, activation='relu'),
    keras.layers.Dense(units=48, activation='relu'),
    keras.layers.Dense(units=1)
    ])
```

✓ 0.0s

```
def compile_model(model):
    model.compile(optimizer=keras.optimizers.Adam(),
                  loss=keras.losses.MeanAbsolutePercentageError(),
                  metrics=[tf.keras.metrics.RootMeanSquaredError()])
    return model
```

7]

```
#@title Функция для графика ошибки
def plot_loss(history):
    fig, axes = plt.subplots(figsize=(15, 5))
    axes.plot(history['root_mean_squared_error'], label='loss')
    axes.plot(history['val_root_mean_squared_error'], label='val_loss')
    axes.set_xlabel('Эпоха')
    axes.set_ylabel('RMSE')
    axes.legend()
    axes.grid(True)
    plt.show()
```

8]

```
#@title Сборка НС
model = construct_model()
```

9]

```
#@title Компиляция НС
model = compile_model(model)
```

0]

```
model.summary()
```

1]

**Model: "sequential_4"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_14 (Dense) | (None, 32) | 416 |
| dense_15 (Dense) | (None, 64) | 2,112 |
| dense_16 (Dense) | (None, 48) | 3,120 |
| dense_17 (Dense) | (None, 1) | 49 |

**Total params:** 5,697 (22.25 KB)

**Trainable params:** 5,697 (22.25 KB)

**Non-trainable params:** 0 (0.00 B)

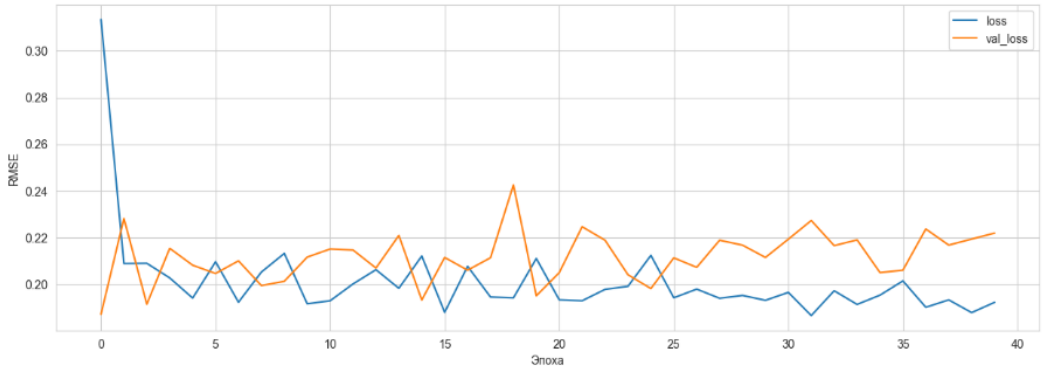
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Результаты работы нейросети

```
Epoch 11/40
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 35.0801 - root_mean_squared_error: 0.1945 - val_loss: 53.3361 - val_root_mean_squared_error: 0.2151
Epoch 12/40
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 38.5762 - root_mean_squared_error: 0.2101 - val_loss: 53.1198 - val_root_mean_squared_error: 0.2147
Epoch 13/40
...
Epoch 40/40
16/16 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 31.2073 - root_mean_squared_error: 0.1917 - val_loss: 52.7529 - val_root_mean_squared_error: 0.2219
CPU times: total: 1.59 s
Wall time: 4.34 s
```

*Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...*

```
plot_loss(history.history)
```




```
# Проверяем точность нейросети на тестовых данных
model.evaluate(X_test, y_test)
```

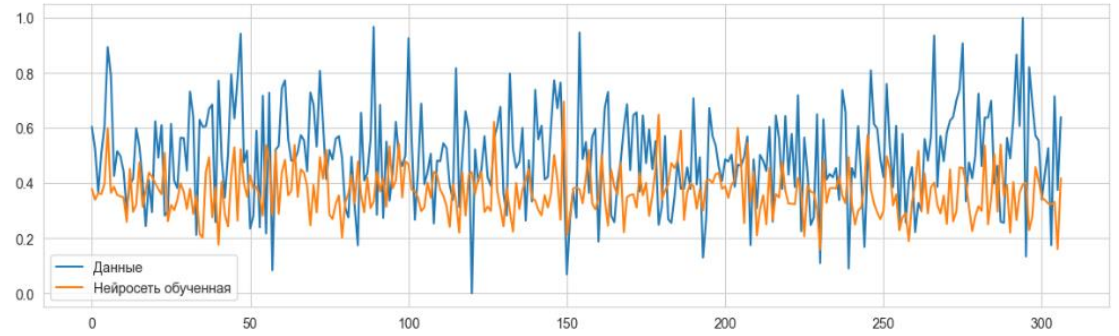
```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 1504045.0000 - root_mean_squared_error: 0.2283

[1432195.25, 0.23443034291267395]
```

```
# Предсказание
y_pred = model.predict(X_test)
```

```
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

```
# Визуализация
fig, ax = plt.subplots(figsize=(14, 4))
ax.plot(y_test, label='Данные')
ax.plot(y_pred, label='Нейросеть обученная')
ax.legend()
plt.show()
```



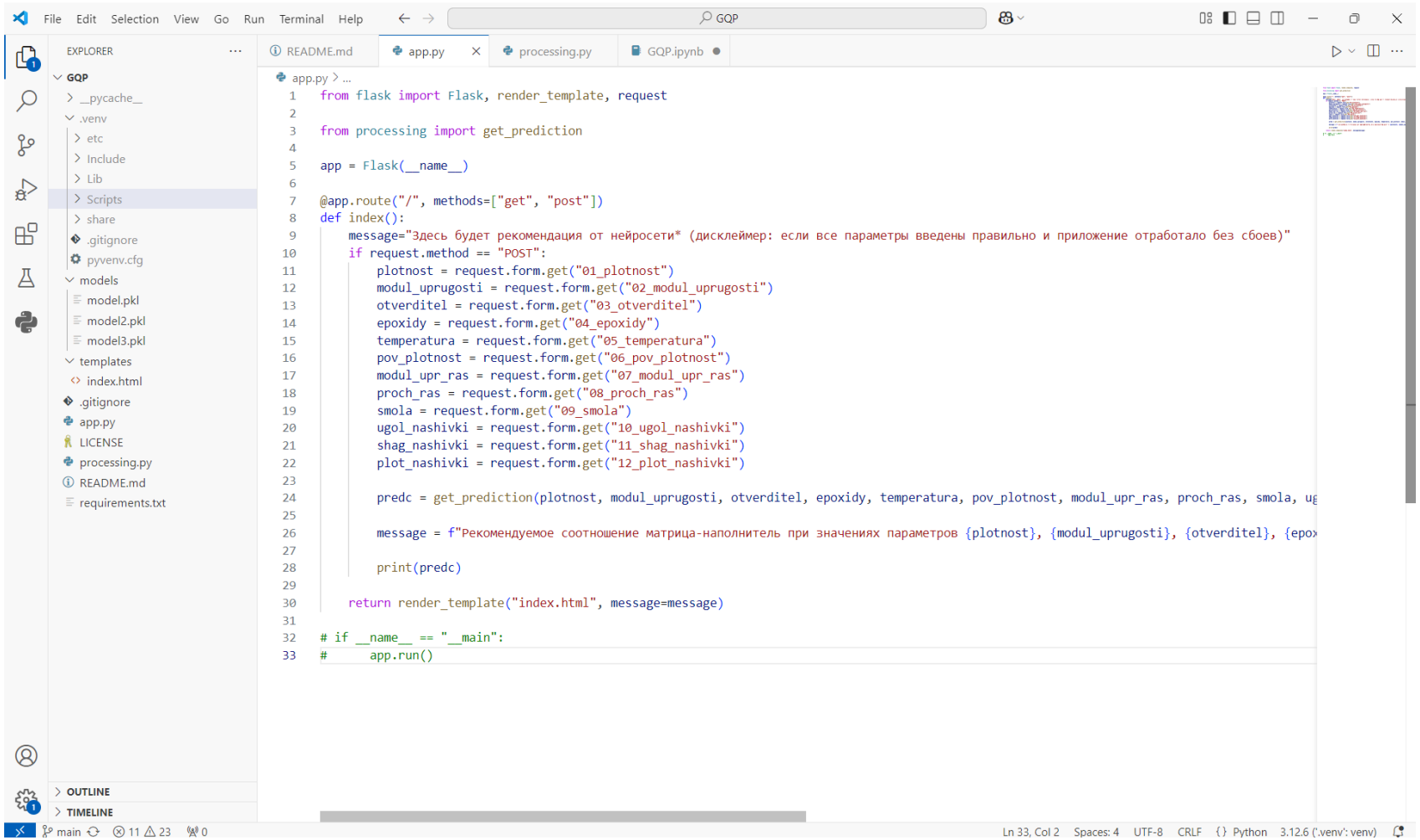

```
#@title Сохраняем модель
pickle.dump(model, open('model2.pkl', 'wb'))

!ls
```

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка приложения

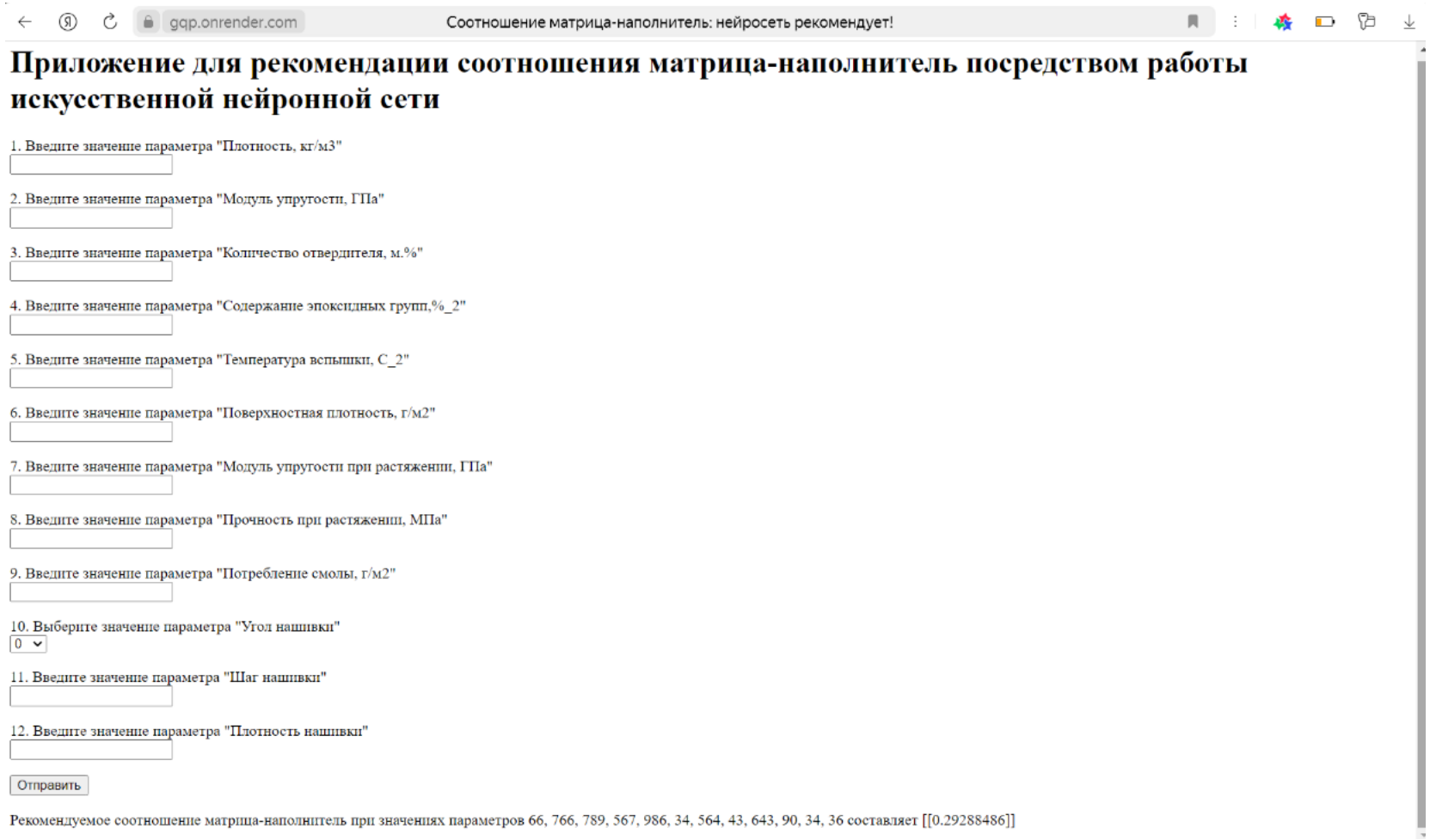
https://github.com/albaross86/GQP



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

# Разработка приложения

https://gqp.onrender.com



ЦЕНТР
ДОПОЛНИТЕЛЬНОГО
ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

do.bmstu.ru

**СПАСИБО ЗА ВНИМАНИЕ!**