

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
по курсу «Data Science»



ЦЕНТР
ДОПОЛНИТЕЛЬНОГО
ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Прогнозирование конечных свойств композиционных материалов

Бойко Татьяна Сергеевна



Этапы исследования

1

Разведочный анализ и предобработка данных

2

Подготовка данных для обучения и тестирования модели

3

Разработка и обучение модели, оценка результатов

4

Нейронная сеть для рекомендации «Соотношение матрица-наполнитель»

5

Создание приложения для расчета параметра «Соотношение матрица-наполнитель»

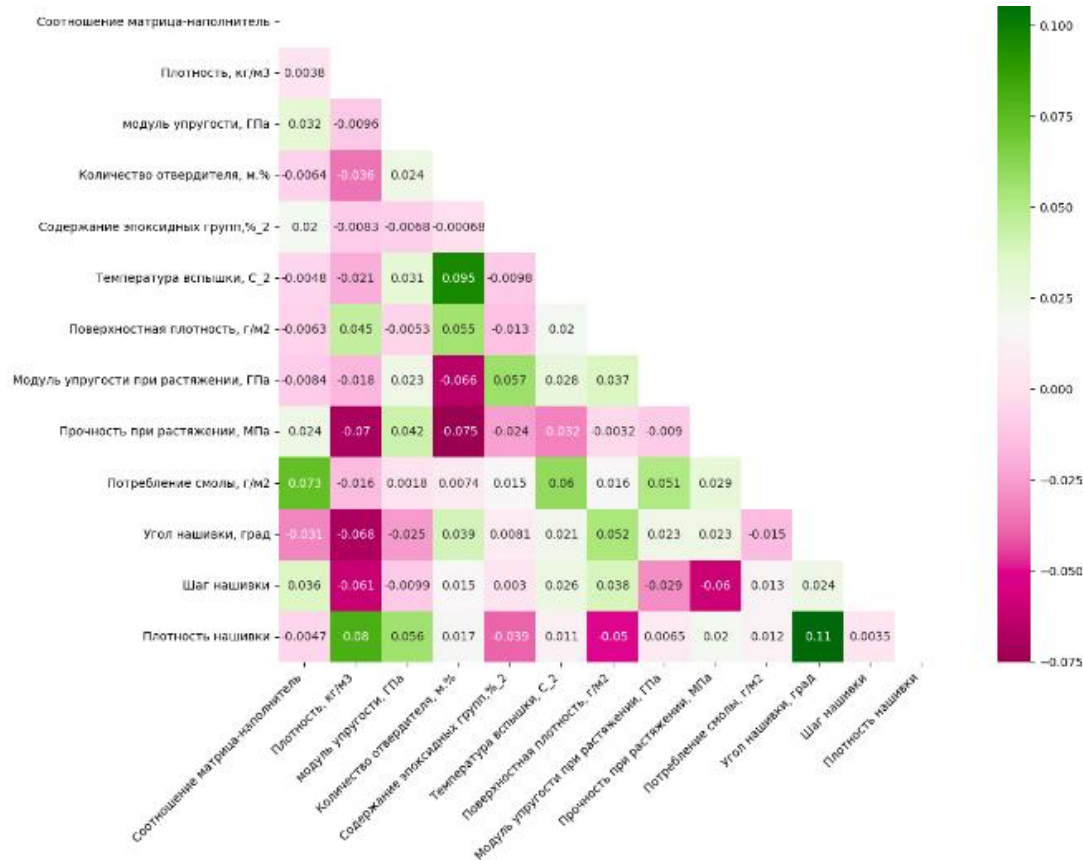
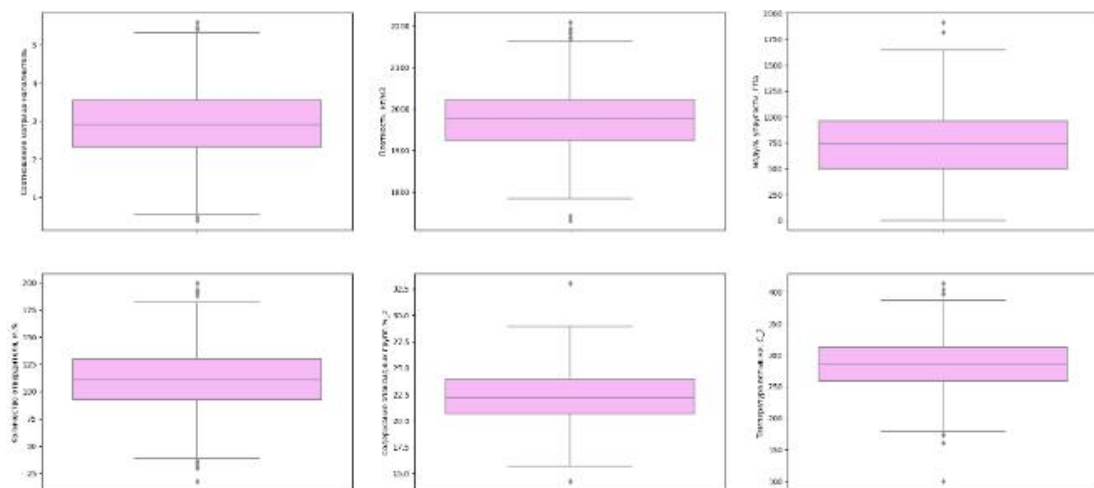
Цель работы: создать и обучить модель на данных о структуре и свойствах материалов, и затем использовать её для прогнозирования.

Актуальность: созданные прогнозные модели помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов возможными новыми характеристиками материалов, и цифровыми двойниками новых композитов.



Разведочный анализ, предобработка данных

| | count | mean | std | min | 25% | 50% | 75% | max |
|--------------------------------------|--------|-------------|------------|-------------|-------------|-------------|-------------|-------------|
| Соотношение матрица-наполнитель | 1023.0 | 2.930386 | 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 | 20.808034 | 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 | 883.225017 | 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 |





Подготовка данных и обучение модели

Используем метод межквартильного диапазона для определения и удаления выбросов в каждой переменной,

```
Q1=df.quantile(q=.25)
Q3=df.quantile(q=.75)
IQR=df.apply(stats.iqr)

df_clean = df[-((df < (Q1-1.5 * IQR)) | (df > (Q3+1.5 * IQR))).any(axis=1)]

df_clean.shape

(936, 13)
```

Выполним нормализацию данных

```
df_norm = df_clean.copy()

scaler_norm = MinMaxScaler()
scaler_norm.fit(df_norm)
df_norm = pd.DataFrame(data=scaler_norm.transform(df_norm), columns=df_norm.columns)
```

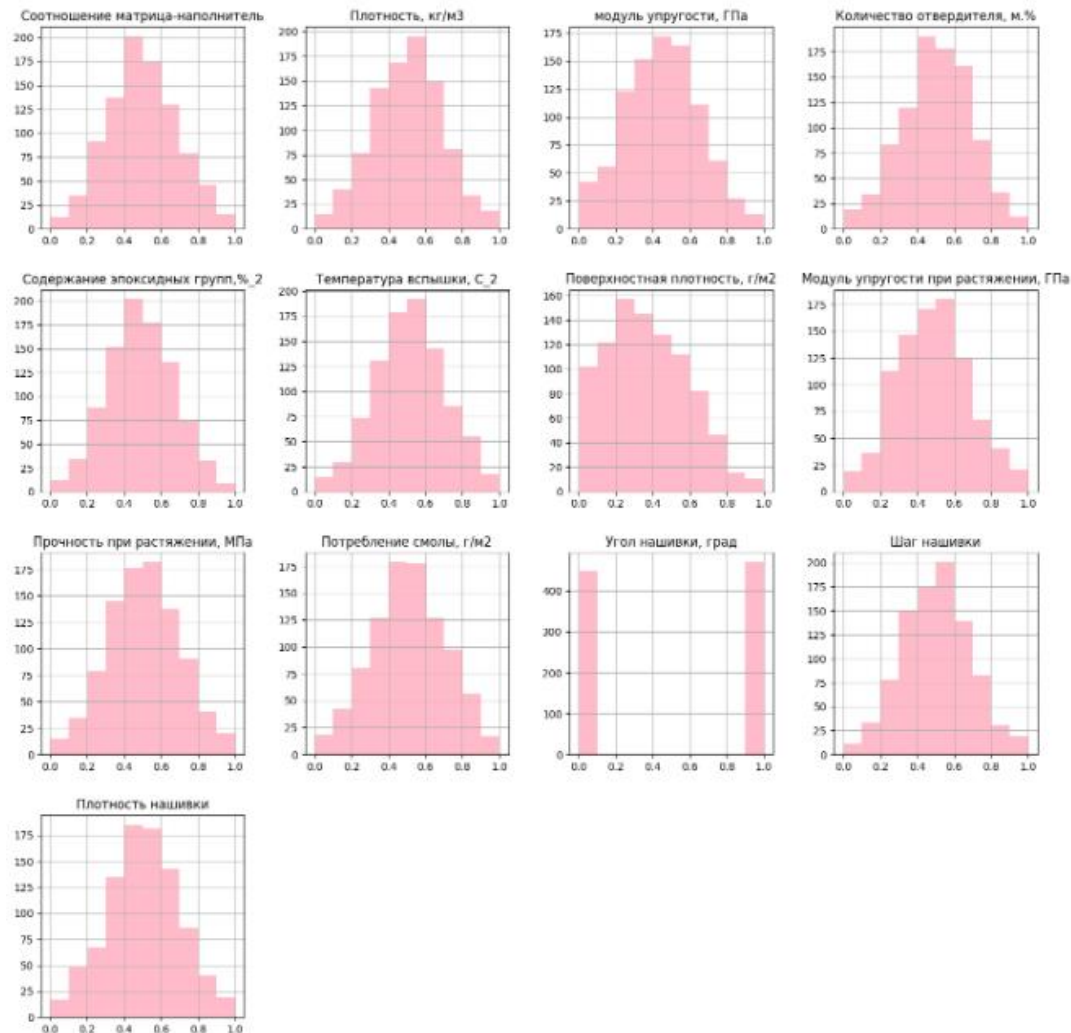
Выполним стандартизацию данных

```
df_std = df_norm.copy()

scaler_std = StandardScaler()
scaler_std.fit(df_std)
df_std = pd.DataFrame(data=scaler_std.transform(df_std), columns=df_std.columns)

df_std.head()
```

| | 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 | n |
|---|---------------------------------|------------------|-----------------------|-----------------------------|----------------------------------|--------------------------|-------------------------------|--------------------------------------|-------------------------------|-------------------------|--------------------|----|
| 0 | -1.195844 | 0.787334 | 0.008829 | -2.285328 | 0.647313 | -0.040478 | -0.974546 | -1.092818 | 1.194636 | 0.035721 | -1.023067 | -1 |
| 1 | -1.195844 | 0.787334 | 0.008829 | 0.667608 | -0.397000 | 0.349690 | -0.974546 | -1.092818 | 1.194636 | 0.035721 | -1.023067 | -0 |
| 2 | -0.174943 | 0.787334 | 0.052380 | 0.026933 | 0.028185 | -0.040478 | -0.974546 | -1.092818 | 1.194636 | 0.035721 | -1.023067 | -0 |
| 3 | -0.178754 | 0.364971 | 0.037113 | 0.026933 | 0.028185 | -0.040478 | -0.974546 | -1.092818 | 1.194636 | 0.035721 | -1.023067 | -0 |
| 4 | -0.400199 | -0.902116 | 0.217265 | 0.026933 | 0.028185 | -0.040478 | -0.974546 | -1.092818 | 1.194636 | 0.035721 | -1.023067 | -0 |





Поиск гиперпараметров по сетке с перекрестной проверкой, оценка результатов

```
lasso_params = {'alpha': [0.5, 1.5, 5, 10],  
               'max_iter': [400, 700, 1200, 2000]}  
GSCV_lasso_e = GridSearchCV(lasso, lasso_params, n_jobs=-1, cv=10)  
GSCV_lasso_e.fit(X_train_e, y_train_e)  
GSCV_lasso_e.best_params_  
print(f'Лучшая модель на кросс-валидации с параметрами {GSCV_lasso_e.best_params_} и результатом {GSCV_lasso_e.best_score_: .4f}')
```

Лучшая модель на кросс-валидации с параметрами {'alpha': 10, 'max_iter': 400} и результатом -0.0294

```
GSCV_lasso_s = GridSearchCV(lasso, lasso_params, n_jobs=-1, cv=10)  
GSCV_lasso_s.fit(X_train_s, y_train_s)  
GSCV_lasso_s.best_params_  
print(f'Лучшая модель на кросс-валидации с параметрами {GSCV_lasso_s.best_params_} и результатом {GSCV_lasso_s.best_score_: .4f}')
```

Лучшая модель на кросс-валидации с параметрами {'alpha': 10, 'max_iter': 400} и результатом -0.0354

```
best_model_lasso_e = GSCV_lasso_e.best_estimator_  
best_model_lasso_s = GSCV_lasso_s.best_estimator_
```

```
print(best_model_lasso_e)  
print(f'R2-score Lasso для Модуля упругости при растяжении, МПа: {best_model_lasso_e.score(X_test_e, y_test_e).round(3)}')
```

Lasso(alpha=10, max_iter=400)
R2-score Lasso для Модуля упругости при растяжении, МПа: -0.002

| | MSE train | MSE test | R2 train | R2 test | MAE train | MAE test | | MSE train | MSE test | R2 train | R2 test | MAE train | MAE test |
|----------------------------|-----------|----------|----------|---------|-----------|----------|--|-------------|-------------|----------|----------|-----------|----------|
| RandomForestRegressor | 6.9005 | 9.2088 | 0.2483 | -0.0145 | 2.1325 | 2.4773 | | 141389.7205 | 224732.5402 | 0.2756 | 0.0013 | 300.0945 | 384.4820 |
| RandomForestRegressor_best | 6.8874 | 9.1510 | 0.2498 | -0.0082 | 2.0923 | 2.4501 | | 150521.6349 | 226238.0916 | 0.2288 | -0.0054 | 305.7443 | 384.2891 |
| RandomForestRegressor_norm | 0.0290 | 0.0384 | 0.2502 | -0.0054 | 0.1357 | 0.1588 | | 0.0061 | 0.0403 | 0.8226 | -0.0193 | 0.0623 | 0.1625 |
| Lasso | 9.1087 | 9.0485 | 0.0078 | 0.0031 | 2.4409 | 2.4471 | | 191764.4221 | 224528.2006 | 0.0175 | 0.0022 | 348.1372 | 384.1102 |
| Lasso_best | 9.1642 | 9.0960 | 0.0017 | -0.0021 | 2.4453 | 2.4543 | | 191887.4816 | 223679.5537 | 0.0169 | 0.0060 | 348.3619 | 383.0815 |
| Lasso_norm | 0.0387 | 0.0383 | 0.0000 | -0.0009 | 0.1587 | 0.1591 | | 0.0343 | 0.0395 | 0.0000 | -26.9966 | 0.1469 | 0.1607 |
| DecisionTreeRegressor | 0.0000 | 18.7589 | 1.0000 | -1.0667 | 0.0000 | 3.5058 | | 0.0000 | 412531.9984 | 1.0000 | -0.8332 | 0.0000 | 505.6368 |
| DecisionTreeRegressor_best | 8.6243 | 9.7378 | 0.0606 | -0.0728 | 2.3382 | 2.5229 | | 184036.1737 | 229794.8330 | 0.0571 | -0.0212 | 332.8734 | 386.5286 |
| DecisionTreeRegressor_norm | 0.0363 | 0.0410 | 0.0606 | -0.0728 | 0.1517 | 0.1637 | | 0.0323 | 0.0403 | 0.0571 | -0.0212 | 0.1395 | 0.1620 |
| KNeighborsRegressor_best | 9.0767 | 9.2401 | 0.0113 | -0.0180 | 2.4322 | 2.4806 | | 190223.9471 | 225364.1789 | 0.0254 | -0.0015 | 346.2832 | 385.7366 |
| KNeighborsRegressor_norm | 0.0385 | 0.0382 | 0.0055 | 0.0010 | 0.1584 | 0.1591 | | 0.0326 | 0.0406 | 0.0488 | -0.0276 | 0.1435 | 0.1636 |
| SGDRegressor_best | 16.4760 | 15.3740 | -0.7947 | -0.6938 | 3.2687 | 3.2256 | | 216378.0788 | 263885.3112 | -0.1086 | -0.1727 | 368.6177 | 417.0691 |
| SGDRegressor_norm | 0.0387 | 0.0383 | -0.0002 | -0.0019 | 0.1588 | 0.1591 | | 0.0344 | 0.0396 | -0.0031 | -0.0013 | 0.1473 | 0.1611 |
| Ridge_best | 9.0739 | 9.0440 | 0.0116 | 0.0036 | 2.4419 | 2.4490 | | 191839.1980 | 223790.0397 | 0.0172 | 0.0055 | 348.3967 | 383.4161 |
| Ridge_norm | 0.0382 | 0.0382 | 0.0116 | 0.0011 | 0.1582 | 0.1593 | | 0.0338 | 0.0394 | 0.0136 | 0.0026 | 0.1460 | 0.1608 |

Все примененные модели не
справились с задачей,

результат

неудовлетворительный.

Скорее всего, проблема

связана с недостатком

вводных данных,

использованными подходами,

инструментами и методами, а

также необходимостью

дополнительных

исследований, включая

консультации экспертов.



Нейронная сеть для рекомендации «Соотношение матрица-наполнитель»

```
model2 = Sequential(x_train_n)

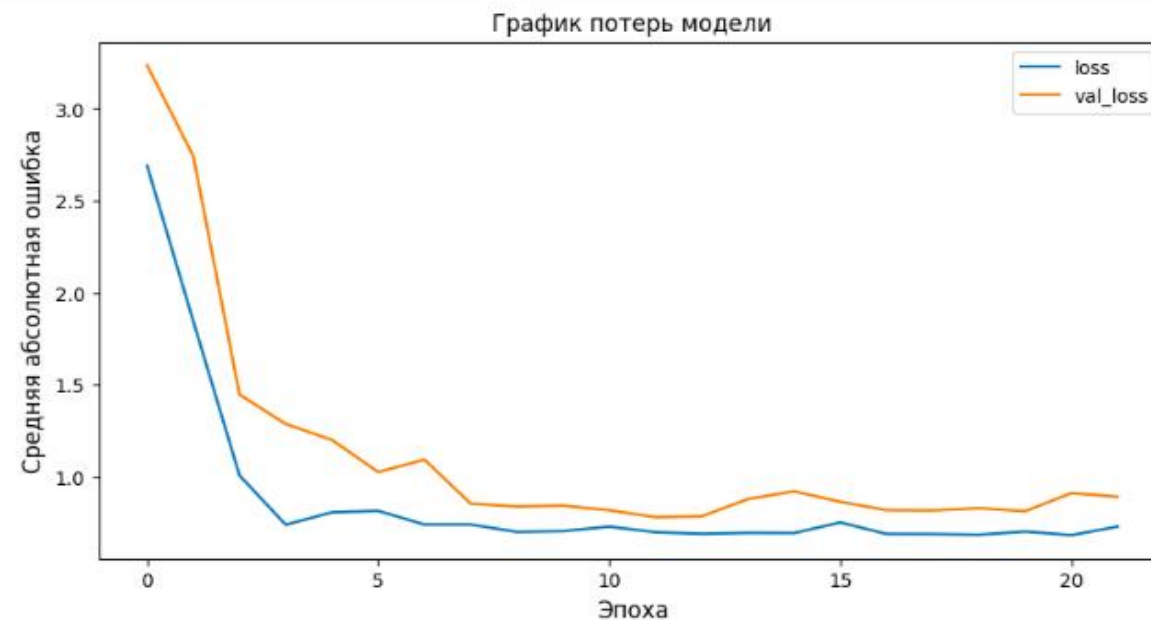
model2.add(Dense(128))
model2.add(BatchNormalization())
model2.add(LeakyReLU())
model2.add(Dense(128, activation='selu'))
model2.add(BatchNormalization())
model2.add(Dense(64, activation='selu'))
model2.add(BatchNormalization())
model2.add(Dense(32, activation='selu'))
model2.add(BatchNormalization())
model2.add(LeakyReLU())
model2.add(Dense(16, activation='selu'))
model2.add(BatchNormalization())
model2.add(Dense(1))
model2.add(Activation('selu'))

early_model2 = EarlyStopping(monitor='val_loss', min_delta=0, patience=10, verbose=1, mode='auto')

model2.compile(optimizer=tf.optimizers.SGD(learning_rate=0.02, momentum=0.5), loss='mean_absolute_error')

%%time
history2 = model2.fit(
    x_train,
    y_train,
    batch_size = 64,
    epochs=100,
    verbose=1,
    validation_split = 0.2,
    callbacks = [early_model2]
)
```

model_loss_plot(history2)



Средняя абсолютная ошибка (MAE) на обучающем наборе данных ниже, чем на валидационном.

Разработка приложения

```
import flask
from flask import render_template
import tensorflow as tf
from tensorflow import keras
import numpy as np

app = flask.Flask(__name__, template_folder = 'templates')

def prediction(param1, param2, param3, param4, param5, param6, param7, param8, param9, param10, param11, param12):
    model = tf.keras.models.load_model('../models/model2/')
    prediction = model.predict([param1, param2, param3, param4, param5, param6, param7, param8, param9, param10, param11, param12])
    return prediction[0][0]

@app.route('/', methods=['POST', 'GET'])

def main():
    params_list = []
    result = ''
    if flask.request.method == 'GET':
        return render_template('main.html')

    if flask.request.method == 'POST':
        for i in range(1,13,1):
            param = flask.request.form.get(f'param{i}')
            params_list.append(float(param))

        result = prediction(*params_list)
        return render_template('main.html', result=result)

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

← → ↺ 127.0.0.1:5000

Прогнозное значение параметра "Соотношение матрица-наполнитель"

Введите значения для расчета:

| | |
|---------------------------------------|--------------------------------|
| Плотность [кг/м3] | <input type="text" value="2"/> |
| Модуль упругости [ГПа] | <input type="text" value="3"/> |
| Количество отвердителя [м.%] | <input type="text" value="4"/> |
| Содержание эпоксидных групп [%_2] | <input type="text" value="1"/> |
| Температура вспышки [С_2] | <input type="text" value="6"/> |
| Поверхностная плотность [г/м2] | <input type="text" value="4"/> |
| Модуль упругости при растяжении [ГПа] | <input type="text" value="9"/> |
| Прочность при растяжении [МПа] | <input type="text" value="3"/> |
| Потребление смолы [г/м2] | <input type="text" value="5"/> |
| Угол нашивки [град] | <input type="text" value="2"/> |
| Шаг нашивки | <input type="text" value="3"/> |
| Плотность нашивки | <input type="text" value="4"/> |

Результат:

5.254844

На выходе пользователь получает результат прогноза для значения параметра «Соотношение матрица – наполнитель».



ЦЕНТР
ДОПОЛНИТЕЛЬНОГО
ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана



do.bmstu.ru